Inhaltsverzeichnis

1. Einführung

Die Kommunikation zwischen Geschäftspartnern und der Austausch von Informationen zwischen eben diesen stellt eine große Herausforderung für alle Beteiligten dar, die häufig mit hohen Kosten und einer mitunter enormen Komplexität verbunden sind.

Gerade systemübergreifende Geschäftsprozesse – auch über Unternehmensgrenzen hinweg – erfordern, dass alle beteiligten Systeme die prozessrelevanten Daten miteinander in geeigneter Form austauschen können und insbesondere bezüglich der Daten ein gleiches Verständnis haben.

In der Folgezeit stieg damit auch die Bedeutung der Extensible Markup Language (XML) im Umfeld der Sozialversicherung.

Die Vorteile von XML sind nicht zuletzt darin zu sehen, dass XML als offener Internetstandard eine standardisierte, textbasierte Meta-Auszeichnungssprache darstellt, die es ermöglicht, Daten bzw. Dokumente bezüglich Inhalt und Darstellungsform derart zu beschreiben und zu strukturieren, dass sie – vor allem auch über das Internet – zwischen einer Vielzahl von Anwendungen in verschiedensten Hard- und Softwareumgebungen hersteller- und branchenneutral automatisiert ausgetauscht und weiterverarbeitet werden können.

Durch die Schaffung einer einheitlichen XML-Richtlinie sollen sämtliche XML-Aktivitäten im Umfeld der Sozialversicherung zentralisiert und gebündelt werden. Durch deren modularisierten Aufbau sollen so zukünftig einheitliche und integrierte XML-Schnittstellen entwickelt werden können. Dies kann mittel- und langfristig zu Effizienzsteigerungen und Kosteneinsparungen führen. Insbesondere lassen sich durch dieses Vorgehen die Implementierungsaufwände drastisch reduzieren.

Durch die XML-Richtlinie wird für den Datenaustausch auf Basis von XML ein standardisiertes und einheitliches Rahmenwerk geschaffen, mit dessen Hilfe alle zukünftigen XML-Schnittstellenimplementierungen vollständig beschrieben werden können. Hierbei werden nicht nur die Sprachelemente und konkreten Entwurfsprinzipien vorgeschrieben, sondern auch die Grundstrukturen verfahrensneutral festgelegt.

2. Allgemeine Grundlagen

2.1 Verwendung von Schlüsselworten

Für die genaue Unterscheidung zwischen der Verbindlichkeit und Aussagekraft von Inhalten und Vorgaben werden die dem [RFC 2119] entsprechenden Großbuchstaben in deutscher Sprache verwendet:

- **MUSS** bedeutet, dass es sich um eine absolut gültige Festlegung bzw. Anforderung handelt.
- **DARF NICHT** bezeichnet den absolut gültigen Ausschluss einer Festlegung bzw. Anforderung.
- **SOLL** beschreibt eine Empfehlung. Abweichungen zu diesen Festlegungen sind in begründeten Fällen möglich.
- **SOLL NICHT** kennzeichnet die Empfehlung, eine Eigenschaft auszuschließen. Abweichungen sind in begründeten Fällen möglich.
- **KANN** bedeutet, dass die Eigenschaften fakultativ oder optional sind und damit keinen allgemeingültigen Empfehlungscharakter besitzen.

2.2 Begriffsdefinition

Dieses Kapitel dient der Definition und Festlegung allgemeiner Begriffe.

Definition:	**XML-Regelwerk** – Allgemeine Entwurfsprinzipien und Vorgaben zur Implementierung einer auf der XML-Richtlinie basierenden XML-Schnittstellenimplementierung. (vgl. Kapitel 4 – Regeln) **XML-Richtlinie** – Die XML-Richtlinie enthält neben dem XML-Regelwerk auch noch Vorschriften zur Versionierung von Schema-Dateien sowie Regeln zur Vergabe von Schemaversionsnummern siehe Kapitel 4.9 und 5.2. Die Dateinamenskonventionen sind ebenfalls Bestandteil der XML-Richtlinie siehe hierzu Kapitel 5.1. **XML-Schnittstelle** – Ein offenes und standardisiertes Datenformat, das einen automatisierten elektronischen und bidirektionalen XML-Datenaustausch zwischen allen Kommunikationspartnern im SV- Umfeld ermöglicht, erfordert die Definition von Semantik, Struktur und von Datentypen in Form von XML-Schemata für die Implementierung einer auf der XML-Richtlinie basierenden XML-Schnittstelle. Im Gegensatz zu der XML-Richtlinie, die sich auf einem höheren Abstraktionsniveau befindet, stellt eine XML-Schnittstelle eine konkrete Anwendung, d. h. Umsetzung und Ausprägung, der XML-Richtlinie dar.

2.3 XML Basis-Schema

Es steht ein GI4X Basis-Schema mit verfahrensübergreifenden Datentypdefinitionen zur Verfügung. In diesem Schema sind die „SimpleTypes“ definiert, die in mehreren Verfahren im SV-Umfeld eingesetzt werden. Diese definierten „SimpleTypes“ können z. B. zur Erstellung von verfahrensspezifischen „ComplexTypes“ genutzt werden. Das GI4X Basis-Schema ist anzupassen, wenn weitere „SimpleTypes“ verfahrensübergreifend zu definieren sind oder „SimpleTypes“ obsolet werden.

3. Anforderungen und Zielsetzung

Der Nutzung von XML für den strukturierten Austausch von Daten im SV-Umfeld kommt eine immer größere Bedeutung zu.

Aus konkreten Implementierungen können die Vorgaben und Festlegungen der XML-Richtlinie anhand von Praxiserfahrungen im Zeitablauf evaluiert und somit gehärtet bzw. falsifiziert oder aber auch ergänzt werden. Über einen solchen dynamischen Prozess kann die Qualität der XML-Richtline stetig überprüft und im Zeitablauf verbessert werden.

Originäres Ziel MUSS sein, sämtliche XML-Aktivitäten im Umfeld der Sozialversicherung zu vereinheitlichen und in Form eines Rahmenwerkes verbindlich für die Implementierung von Schnittstellen festzuschreiben.

4. Regeln

4.1 Allgemein

4.1.1 Umfang XML-Sprachelemente

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-AU-1	[DARF NICHT]	Alle Sprachelemente von XML-Schemata, die nachfolgend nicht explizit aufgeführt sind, werden zunächst solange explizit ausgeschlossen, bis begründete Tatbestände für deren Verwendung und damit nachträglicher expliziten Berücksichtigung sprechen.	

4.1.2 Dateinamenkonvention

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-AD-1	[MUSS]	Der Dateiname muss wie folgt aufgebaut sein: [VK]-[QN]-[VN]-[LN].[SUF].	Zusatzinformationen zu der Regel sind im Kapitel 5.1 aufgeführt.

4.1.3 XML-Header

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-AH-1	[MUSS]	Es werden ausschließlich XML in der Version 1.0 und 1.1 sowie XML-Schemata in der Version 1.0 unterstützt.	
RX-AH-2	[MUSS]	Der Wert des „elementFormDefault“ Attributes muss „qualified“ sein.	
RX-AH-3	[MUSS]	Der Wert des „attributeFormDefault“ Attributes muss „unqualified“ sein. Eine Ausnahme stellen in einem anderen Schema wieder verwendbare, globale Attribute dar, die den Zielnamensraum des Schemas, in dem sie definiert sind, besitzen MÜSSEN.	

4.1.4 XML-Namensraum

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-AN-1	[MUSS]	Jeder Namensraum **MUSS** eindeutig innerhalb der Namensraumhierarchie von GI4X sein.	
RX-AN-2	[MUSS]	Der Namensraum MUSS wie folgt aufgebaut sein: Eine URI gemäß [RFC 2396] gefolgt von der Verfahrenskennung und der Versionsnummer: URI(beliebig)/[VK]-[QN]/[VN‘] (für verfahrensspezifische Schemata und verfahrensübergreifende Schemata) bzw. URI(beliebig)/[VK]-basis/[VN‘] (nur für verfahrensübergreifende Schemata) Die Namensraumversion [VN’] ist zweiteilig, besteht maximal aus 7 Stellen und besitzt den allgemeinen Aufbau [HVNR].[NVNR].	Die Bestandteile der Namensraumversion sind im Folgenden Kapitel erläutert: 5.2.2. Die Änderungsgründe für die Erhöhung von HVNR und NVNR gemäß der Schemaversion sind in den folgenden Kapitel aufgeführt: 5.2.3.2.2, 5.2.3.2.3 Des Weiteren sind Informationen bzgl.

		Ausnahme bildet das GI4X-Basisschema. Dieses ist wie folgt zu benennen: GI4X:/xml-schema/GI4X-basis/[VN'] Eine Ausnahme bilden sogenannte Brücken-Schemata. Der Namensraum eines Brückenschemas MUSS derselbe sein wie der des Schemas, das die in einem Brückenschema eingeschränkten Typen definiert.	des Aufbaus beginnend mit VK im Kapitel 5.1 aufzufinden. Informationen zum Brückenschemata sind dem Folgenden Kapitel zu entnehmen: 5.1
RX-AN-3	[MUSS]	Im jeweiligen Instanzdokument **MÜSSEN** die verwendeten Namensräume im Root-Element angegeben werden. Eine andere Lokation ist unzulässig.	
RX-AN-4	[MUSS]	Die Verwendung von Namensräumen ist für Elemente verbindlich	Siehe hierzu auch RX-AH-2
RX-AN-5	[MUSS]	Für jeden importierten Namensraum MUSS ein Präfix im **<xs:schema>** Element deklariert werden.	

4.2 Design

4.2.1 Designvorgaben

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-DV-1	[SOLL]	In aller Regel sind Element – Kindelement Strukturen zu bevorzugen, da sich diese im Bedarfsfall in einer beliebigen Granularität einfacher erweitern lassen. Hierdurch lassen sich die damit verbundenen Änderungsaufwände minimieren.	
RX-DV-2	[KANN]	Attribute sollten mit Vorsicht verwendet werden, da diese gerade auch hinsichtlich zukünftiger Änderungen deutlichen Restriktionen unterliegen (insbesondere fehlende strukturelle Granularität).	
RX-DV-3	[SOLL]	Für die Modellierung von Meta-Daten, die in aller Regel keinen strukturellen Änderungen unterworfen sind und darüber hinaus eher atomarer Natur sind, kann eine Element-Attribut-Struktur durchaus sinnvoll sein.	
RX-DV-4	[SOLL]	Die Definition eines Schemas sollte sich daran orientieren, in welcher Art und	

		Weise die Daten aus fachlicher Sicht strukturiert und nachfolgend weiterverarbeitet werden sollen (vgl. konzeptueller und physischer Schemaentwurf).[1]	
RX-DV-5	[SOLL]	Jedes Schema sollte lediglich ein Root-Element definieren.	
RX-DV-6	[SOLL]	Elemente und Attribute sollten nur dann global definiert werden, wenn diese auch wirklich an anderer Stelle wiederverwendet werden können oder sollen.	
RX-DV-7	[MUSS]	Lokale Definitionen an unterschiedlichen Stellen, jedoch mit gleichem Namen und gleicher inhaltlichen Bedeutung, sind global zu definieren.	
RX-DV-8	[DARF NICHT]	Lokale Definitionen an unterschiedlicher Stelle, jedoch mit gleichem Namen aber unterschiedlicher inhaltlichen Bedeutung, sind unzulässig.	
RX-DV-9	[DARF NICHT]	Mixed Content darf nicht verwendet werden.	
RX-DV-10	[SOLL]	Import und Include sollen verwendet werden.	
RX-DV-11	[SOLL]	blockDefault bzw. finalDefault sollen verwendet werden	

4.2.2 Designprinzipien

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-DP-1	[DARF NICHT]	Russian Doll darf nicht angewandt werden: Alle Elemente, außer dem Root, und alle Typen sind lokal definiert.	
RX-DP-2	[DARF NICHT]	Garden of Eden: Alle Elemente und Typen sind global definiert.	
RX-DP-3	[SOLL NICHT]	Salami Slice: Alle Elemente sind global und alle Typen sind lokal definiert.	
RX-DP-4	[SOLL]	Ein Schemaentwurf sollte auf Grundlage des Venetian Blinds Designprinzips erfolgen, in dem alle Typen global, und Elemente und Attribute grundsätzlich zunächst lokal definiert werden sollten.	

[1] Beim konzeptuellen Schemaentwurf wird eine saubere Abbildung der natürlichen Beziehungen der repräsentierten Realwelt-Objekte in Form spezifischer hierarchischer Strukturen vorgenommen. Der physikalische Schemaentwurf orientiert sich mehr an den Anforderungen der nachgelagerten Prozesse. In diesem Zusammenhang werden häufig flache Strukturen für einen einfacheren Datenbankimport gewählt.

RX-DP-5	[DARF NICHT]	Chamäleon-Schemata dürfen nicht verwendet werden.	Siehe hierzu RX-DV-6.

4.2.3 Strukturelemente

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-DS-1	[SOLL]	Die Strukturelemente <xs:sequence> und <xs:choice> können verwendet werden.	
RX-DS-2	[DARF NICHT]	Das Strukturelement <xs:all> ist nicht zu verwenden, da insbesondere durch dessen Verwendung die Reihenfolge der in dem entsprechenden Inhaltsmodell enthaltenen Elemente beliebig und deren Kardinalität auf {0,1} bzw. {1,1} beschränkt ist.	

4.2.4 Kardinalitäten

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-DK-1	[SOLL]	Die Kardinalitäten unterliegen keinerlei Beschränkungen und sind zu verwenden	

4.3 Bezeichner

4.3.1 Allgemein

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BA-1	[SOLL]	Alle Bezeichner SOLLTEN **deutsche Wörter** (auch Zusammensetzungen aus mehreren Wörtern) verwenden. Abkürzungen SOLLEN weitestgehend vermieden werden. Als Ausnahme KÖNNEN (verfahrens-) bekannte Abkürzungen, wie beispielsweise BBNR, PLZ verwendet werden.	
RX-BA-2	[SOLL]	Die Bezeichner SOLLTEN kurze, aussagekräftige Namen haben.	
RX-BA-3	[MUSS]	Zulässig für Bezeichner sind nur folgende Zeichen: [0-9]: Numerische Zeichen [A-Z, a-z]: Buchstaben des deutschen Alphabets ohne Umlaute und ß	

		[_]: Als Trennzeichen ist nur _ zulässig.	

4.3.2 Elemente/Attribute

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BE-1	[DARF NICHT]	Schema-Elemente DÜRFEN NICHT mit numerischen Zeichen beginnen.	
RX-BE-2	[MUSS]	Ein Identifikator zur Unterscheidung der einzelnen Schema-Elemente MUSS verwendet werden.	
RX-BE-3	[MUSS]	Elementnamen beginnen immer mit einem Großbuchstaben: <xs:element name="Name">.	
RX-BE-4	[MUSS]	Attributnamen dürfen keine Großbuchstaben verwenden: <xs:attribute name="name">	

4.3.3 Attributgruppe

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BG-1	[MUSS]	Attributgruppennamen unterliegen den Restriktionen für Attributnamen und enden mit dem Identifikator _Grp: <xs:attributeGroup name="name_Grp">.	

4.3.4 Modellgruppe

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BM-1	[MUSS]	Modellgruppennamen beginnen immer mit einem Großbuchstaben und enden mit dem Identifikator _Grp: <xs:group name="Name_Grp">.	

4.3.5 Einfache und komplexe Typen

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BC-1	[MUSS]	Die Namen von einfachen - und komplexen Typen unterliegen den Restriktionen für Elemente. Einfache Typen enden zusätzlich mit dem Identifikator _Stp, komplexe Typen mit _Ctp. Eine Ausnahme bilden Listen und Vereini-	

		gung, die statt mit _Stp mit den Identifikatoren _Lst bzw. _Unn enden: <xs:simpleType name="Name_Stp">, <xs:complexType name="Name_Ctp">,	

4.3.6 Listen

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BL-1	[MUSS]	Der Name von Listen unterliegt den Restriktionen für Elemente und endet zusätzlich mit dem Identifikator _Lst: <xs:simpleType name="Name_Lst"> <xs:list itemType="xs:string"/> </xs:simpleType>	

4.3.7 Vereinigungen

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BV-1	[MUSS]	Der Name von Vereinigungen unterliegt den Restriktionen für Elemente und endet zusätzlich mit dem Identifikator _Unn: <xs:element name="Dateigroesse_Unn"> <xs:simpleType> <xs:union memberTypes="Union1_Stp Union2_Stp" /> </xs:simpleType> </xs:element>	

4.3.8 Identitätseinschränkung

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-BI-1	[MUSS]	Für Identitätseinschränkungen gelten die Restriktionen für Elemente in Verbindung mit folgenden Restriktionen: <xs:unique> – Der Name endet mit dem Identifikator _Uqe. <xs:key> – Der Name endet mit dem Identifikator _Key. <xs:keyref> – Der Name endet mit dem Identifikator _Krf: <xs:unique name="Name_Uqe"> <xs:selector xpath="Ausgangsmenge" /> <xs:field xpath="eindeutiger Wert" /> </xs:unique> <xs:key name="Name_Key"> <xs:selector xpath="Ausgangsmenge"/> <xs:field xpath="eindeutiger Wert " /> </xs:key> <xs:keyref name="Name_Krf" refer=" Name_Key"> <xs:selector xpath="Ausgangsmenge" /> <xs:field xpath=" eindeutiger Wert " /> </xs:keyref>	

4.4 Typen

4.4.1 Einfache Typen (Simple Types)

4.4.1.1 Allgemein

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-TEA-1	[MUSS]	Aus Praktikabilitätsgründen ist die Nutzung auf die folgenden vordefinierten Primitivtypen von XML-Schemata zu beschränken: string, normalizedString, token, boolean base64binary, hexBinary, float, decimal, integer, positiveInteger, long, int, unsignedInt, double, anyURI, QName, duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth.	
RX-TEA-2	[SOLL]	Bei der Verwendung der obigen Datentypen soll der spezifischste Datentyp verwendet werden. So soll zum Beispiel int – und nicht Integer – verwendet werden, wenn der Wertebereich von int zur Modellierung der Werte ausreicht.	Siehe hierzu RX-TEA-1
RX-TEA-3	[KANN]	Weiterhin ist der zulässige Wertebereich mit Hilfe von regulären Ausdrücken (pattern) und/oder Enumerationen und Identitätseinschränkungen sinnvoll einzugrenzen. Insbesondere ist der abgeleitete Datentyp <xs:token> dem primitiven Datentyp <xs:string> für die Definition von reinen „Textfeldern“ vorzuziehen.	Für Enumeration siehe Kapitel: 4.4.1.2. Für Identitätseinschränkungen siehe Kapitel: 4.3.8.
RX-TEA-4	[KANN]	Eine Einschränkung und Erweiterung von einfachen Typen ist zulässig. Insbesondere sind Ableitungen mittels Fassetten zu verwenden (minLength, maxLength, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, fractionDigits, enumeration, pattern).	

4.4.1.2 Enumeration

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-TEE-1	[KANN]	Enumerationen sind zu verwenden. Sie sind für die Einschränkung von Typen und Deklarationen und in Verbindung mit Identitätseinschränkungen in solchen Fällen nutzbar, in denen eine Menge an zulässigen Werten (sog. statische Wertelisten) die Grundlage für eine Einschränkung sein soll.	
RX-TEE-2	[KANN]	Enumerationen können für die Modellierung kleiner, atomarer, sich selten ändernder Schlüsseltabellen eingesetzt werden.	

4.4.1.3 List/Union

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-TEL-1	[SOLL]	Vereinigungen <xs:union> und Listen <xs:list> sind zu verwenden.	

4.4.1.4 Wertelisten

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-TEW-1	[MUSS]	Sind die typeinschränkenden Wertelisten endlich und im Zeitablauf eher stabil hinsichtlich der Änderungshäufigkeit, so sind statische Wertelisten zu verwenden. Solche Wertelisten sind als einfache Typen im entsprechenden Schema zu hinterlegen.	
RX-TEW-2	[MUSS]	Sind die typeinschränkenden Wertelisten im Zeitablauf eher häufigen Änderungen unterzogen und deren Werte nicht abschließend bekannt, so sind dynamische Wertelisten in Form von möglichst exakten lexikalischen Einschränkungen zu verwenden.	
RX-TEW-3	[SOLL]	Solche dynamischen Wertelisten sollten einen Verweis auf eine externe Liste enthalten. Hierzu ist <xs:annotation> in Verbindung mit <xs:documentation> zu verwenden. Über das Attribut source	

		SOLL die URL, über die die externe Liste erreichbar ist, angegeben werden.	

4.4.2 Komplexe Typen (Complex Types)

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-KT-1	[MUSS]	Komplexe Typen sind zu verwenden. Sie tragen zur Modularisierung und Wiederverwendung bei.	
RX-KT-2	[KANN]	Eine Einschränkung und Erweiterung von komplexen Typen ist zulässig. Jedoch sind Einschränkungen mit Vorsicht zu verwenden, da sie mitunter sehr komplex werden können und im Gegensatz zur Erweiterung weder den Konzepten der objektorientierten Programmierung noch denen der relationalen Datenbanken genau entsprechen. Insbesondere muss darauf geachtet werden, dass jede Änderung an einem Vorgängertyp manuell in die gesamte Ableitungsstruktur übertragen werden muss. Sie können jedoch durchaus zweckmäßig sein, wenn sekundäre Typen einem generischen primären Typen entsprechen müssen, aber dennoch ihre eigenen Einschränkungen verwenden, um über die des primären Typs hinauszugehen.	

4.4.3 Einschränkungen/Erweiterungen

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-EE-1	[MUSS]	Für diejenigen Typdefinitionen, die nicht weiter abgeleitet bzw. keine abgeleiteten Typen verwenden dürfen, sind die Attribute final bzw. block entsprechend zu setzen.	
RX-EE-2	[SOLL]	Eine Einschränkung des Wertebereichs in Form von statischen oder dynamischen Wertelisten ist vorzunehmen, soweit die Kriterien hierzu bekannt und vom Fachverfahren vorgegeben werden können.	

RX-EE-3	[SOLL NICHT]	Einschränkungen von Wertebereichen mittels <xs:length> sind zu vermeiden.	
RX-EE-4	[SOLL]	Einschränkungen von Wertebereichen mittels <xs:minLength> und <xs:max-Length> sind zu verwenden.	
RX-EE-5	[MUSS]	Alle Pflichtelemente und Attribute müssen eine Mindestlänge von 1 haben.	
RX-EE-6	[KANN]	Alle optionalen Elemente und Attribute können eine Mindestlänge von 0 haben. Dies ist insbesondere für die Repräsentation von leeren Werten notwendig.	Vgl. RX-LW-2

4.4.4 Typersetzungen

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-TE-1	[DARF NICHT]	Typersetzungen mittels Ersetzungsgruppen oder xs:type dürfen nicht verwendet werden. Hierzu MUSS auch das Attribut blockDefault="substitution" von <xs:schema> gesetzt werden.	

4.4.5 Typen – und Gruppenneudefinitionen (Redefines)

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-TR-1	[DARF NICHT]	Neudefinitionen durch <xs:redefine> sind nicht zu verwenden. Die Anwendung von Neudefinitionen wirkt sich nicht nur auf das einbettende Schema, sondern auch auf das eingebettete Schema aus. Somit zeigen alle Verweise auf den ursprünglichen Typ dann auf den neudefinierten Typ – der alte Typ bleibt verdeckt. Dies kann zu Problemen und Konflikten führen, da dies im Gegensatz zu einer Ableitung nicht durch die Verwendung der block oder final Attribute verhindert bzw. begrenzt werden kann. Damit kann die ursprüngliche Semantik vollständig geändert werden (Gefahr des Wildwuchses).	

4.4.6 Abstrakte Typen (Abstract Types)

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-AT-1	[SOLL NICHT]	Abstrakte Typen sind mit Vorsicht zu verwenden. Abstrakte komplexe Typen und Elementdeklarationen werden häufig für die Erstellung von generischen Basistypen verwendet, die gemeinsame Informationen für einen Satz von Typen enthalten und deren Definition jedoch noch unvollständig sind und durch Ableitungen konkretisiert werden müssen (Polymorphismus).	

4.5 Identitätseinschränkungen und Schlüsseltabellen (Identity Constraints)

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-IS-1	[DARF NICHT]	Für die Modellierung von Identitätseinschränkungen dürfen xs:ID/xs:IDREF nicht verwendet werden.	
RX-IS-2	[SOLL]	Für die Modellierung von Identitätseinschränkungen sind <xs:unique>, <xs:key> und <xs:keyref> zulässig und können verwendet werden.	
RX-IS-3	[MUSS]	Die Modellierung von Identitätseinschränkungen ist genau dann verpflichtend, wenn geeignete Schlüssellisten für das Fachverfahren existieren oder die Werte bestimmter Elemente/Attribute eindeutig sein müssen.	
RX-IS-4	[DARF NICHT]	Öffentliche Schlüsseltabellen dürfen nicht in dem Instanzdokument enthalten sein, sondern müssen vielmehr extern referenziert werden. Zur Gewährleistung der Unveränderbarkeit von öffentlichen Schlüsseltabellen in XML-Instanzdokumenten können Hash-Werte herangezogen werden.	
RX-IS-5	[MUSS]	Bei der Verwendung von Identitätsconstraints zur Validierung von Werten müssen zugehörige Schlüsseltabellen über	

		XInclude referenziert werden. Der Wert des XInclude href attributes muss eine relative URL sein, die ausschließlich aus dem Schlüsseltabellen-Dateinamen bestehen darf[2].	
RX-IS-6	[DARF NICHT]	Schlüsseltabellen, die sich oft ändern, dürfen nicht über Datentyp Enumerations Fassetten modelliert werden, da dies zu unnötig vielen Schema-Versionen führen würde.	Des Weiteren sind die Regeln von 4.4.1.2 bei Erstellung von Schlüsseltabellen zu beachten.
RX-IS-7	[MUSS]	Kleine, atomare, statische Schlüsseltabellen, die sich nicht oft ändern, müssen im XML-Schema als Datentyp Enumerations Fassetten modelliert werden.	
RX-IS-8	[KANN]	Kleine, sich oft ändernde Schlüsseltabellen als auch mittelgroße Schlüsseltabellen können über Identitätsconstraints/XInclude instanzbasiert modelliert werden.	
RX-IS-9	[KANN]	Die Prüfung von Werten gegen Schlüsseltabellen kann durch externe Prüfprogramme erfolgen, soweit keine der obigen MUSS Festlegungen erfüllt ist.	

4.6 Gruppen

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-GR-1	[SOLL]	Attribut- und Modellgruppen sind zu verwenden. Sie tragen zur Modularisierung und Wiederverwendung bei. Modellgruppen können jedoch keine komplexen Typen ersetzen.	

[2] Dies ermöglicht und erzwingt, dass das Verarbeitungsprogramm dem Parser die Datei geeignet übergeben muss.

4.7 Leere Werte/Whitespaces/Platzhalter

4.7.1 Leere Werte

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-LW-1	[SOLL]	Für das Nichterscheinen der jeweiligen Komponente, soll die Komponente explizit im Schema als optional deklariert werden.	
RX-LW-2	[SOLL]	Für leere Inhalte soll die Einschränkung des Wertbereichs der jeweiligen Komponente leere Inhalte ermöglichen.	Vgl. RX-EE-6
RX-LW-3	[KANN]	Falls die Einschränkungen des Wertebereiches keinen leeren Inhalt zulassen, sollen die leeren Inhalte durch die Nutzung des Attributs <**xs:nil**> innerhalb des Instanzdokuments dargestellt werden. Hierzu muss das Attribut **nillable="true"** in <**xs:element**> gesetzt werden.	
RX-LW-4	[SOLL]	Die Darstellung und Unterscheidung von Null-Werten und leeren Werten sollte auf Basis von optionalen Komponenten (Attribute und Elemente) erfolgen. Durch Nichterscheinen der jeweiligen Komponente sind Null-Werte und durch leere Inhalte sind wirklich leere Werte darzustellen. Damit ergibt sich ein einheitlicher Ansatz sowohl für Elemente als auch für Attribute.	Für Nullwerte siehe Regel RX-LW-1 Für leere Werte siehe Regeln RX-LW-2 und RX-LW-3

4.7.2 Whitespaces

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-WS-1	[MUSS]	Bei der Behandlung von Leerzeichen, Tabulatoren u. ä., sind die Vorgaben von [XML-Schema] zu berücksichtigen. Abweichungen und Einschränkungen zu dem beschriebenen Vorgehen sind unzulässig (vgl. hierzu [Whitespace]). Damit können all diejenigen Wertebereiche von Typen eingeschränkt werden,	

		die von <xs:string> abgeleitet sind. Zulässige Werte für <xs:whiteSpace> sind preserve, replace und collapse.	

4.7.3 Platzhalter any/anyAttribut

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-PA-1	[SOLL NICHT]	Platzhalter sollten generell vermieden werden.	
RX-PA-2	[DARF NICHT]	Platzhalter dürfen in den Nutzdatenteilen nicht verwendet werden, als Nutzdaten werden ausschließlich die fachlichen Daten angesehen.	
RX-PA-3	[KANN]	Platzhalter können auf der Transportebene (vg. Header-Strukturen), wenn diese vorhanden und nutzbar sind, verwendet werden.	

4.8 Dokumentation

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-DO-1	[MUSS]	Lediglich **<xs:documentation>**, **<xs:appinfo>(jeweils inkl. des Attributs source) und <!-- Dokumentation -->** sind zulässig und somit zu verwenden, um damit neben einer allgemeinen Dokumentation auch insbesondere die Lesbarkeit von Schemadateien zu erhöhen.	
RX-DO-2	[MUSS]	Das Root-Element MUSS in der jeweiligen XSD in Form einer geeigneten Kommentierung hervorgehoben werden, da es hierzu keinen durch XML-Schemata definierten Mechanismus gibt.	
RX-DO-3	[MUSS]	Jede einzelne Dokumentation muss kurz und prägnant sein. Längere Erläuterungen sind separat außerhalb des Schemas zu pflegen und über das „source“ Attribut (Verweis auf das externe Dokument) einzubinden.	
RX-DO-4	[MUSS]	Die Versionshistorie muss in der entsprechenden Schemadatei hinterlegt werden. Die protokollierten Änderungen	

		müssen die Versionsnummer, das Änderungsdatum sowie Grund/Art der Änderung umfassen.	
RX-DO-5	[SOLL]	Alle Elemente und Attribute sollten im Schema dokumentiert werden.	
RX-DO-6	[SOLL]	Prüfroutinen und sonstige verfahrensspezifische Merkmale, die nicht durch das jeweilige Schema definiert werden, sind in einer eigenständigen Dokumentation außerhalb dieses Schemas zu beschreiben.	
RX-DO-7	[SOLL]	xs:documentation enthält menschlich interpretierbare Dokumentation, xs:appinfo maschineninterpretierbare Informationen.	

4.9 Versionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VN-1	[MUSS]	Die Schemaversion MUSS in dem Attribut version von <xs:schema> hinterlegt werden. Die erste produktive Schemaversion sollte mit 1.0.0 beginnen. Führende Nullen, wie z. B. 01.0.0 sind nicht erlaubt.	Der Aufbau der Versionsnummer ist im Folgenden Kapitel beschrieben: 5.2.2
RX-VN-2	[MUSS]	Verfahrensspezifische Nachrichten MÜSSEN das Attribut logische_version in ihrem Wurzel-Element enthalten[3]. Die korrespondierende verfahrensspezifische Schemadatei MUSS entsprechend erweitert werden. Die logische Version MUSS zur Kennzeichnung von strukturellen und semantischen Änderungen in verfahrensspezifischen Schemadateien verwendet werden. Sie entspricht in ihrem Aufbau der Schemaversion.	Der Aufbau, das Einsatzgebiet, sowie die Änderung der logischen Version ist im Folgenden Kapitel beschrieben: 5.2.2

[3] Die Wurzel einer Nachricht kann mit der Wurzel des Instanzdokuments zusammenfallen. Wenn Nachrichten aber in ein Transportformat eingebettet werden, trifft dies nicht mehr zu, siehe auch das Beispiel in Kapitel

RX-VN-3	[MUSS]	Aufeinander folgende Versionsnummern MÜSSEN aufsteigend vergeben werden, so dass für eine Versionsnummer vn2, die auf vn1 folgt, immer gilt: a) HVNR(vn2) > HVNR(vn1), oder b) HVNR(vn2) = HVNR(vn1) UND NVNR(vn2) > NVNR(vn1) , oder c) HVNR(vn2) = HVNR(vn1) UND NVNR(vn2) = NVNR(vn1) UND RENR(vn2) > RENR(vn1)	
RX-VN-4	[SOLL]	Aufeinander folgende Versionsnummern SOLLEN, sofern sich Änderungen in einer Komponente ergeben, in Einser-Schritten erhöht werden.	

4.9.1 Regeln zur Vergabe/Erhöhung von Versionsnummern

4.9.1.1 Erhöhung der Revisionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VRE-1	[MUSS]	Die Revisionsnummer MUSS erhöht werden, wenn die Schemadatei geändert wurde, ohne dass die Änderung Auswirkungen auf die Validierung und Verarbeitung des Inhaltes hat.	Die Änderungsgründe für die Revisionsnummer sind im Folgenden Kapitel aufgeführt: 5.2.3.2.1
RX-VRE-2	[MUSS]	Die Revisionsnummer des Schemas MUSS ebenfalls erhöht werden, wenn sich die Revisionsnummer (und nur diese!) eines referenzierten Schemas ändert.	
RX-VRE-3	[SOLL]	Die Revisionsnummer SOLL auf '0' zurückgesetzt werden, wenn sich die Nebenversionsnummer und/oder Hauptversionsnummer ändert.	

4.9.1.2 Erhöhung der Nebenversionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VNE-1	[MUSS]	Die Nebenversionsnummer MUSS erhöht werden, wenn Erweiterungen an der XSD-Datei vorgenommen werden, die als kompatibel zu XSD-Dateien mit der gleichen Hauptversionsnummer eingestuft werden	Die Änderungsgründe für die Nebenversionsnummer sind im Folgenden Kapitel aufgeführt: 5.2.3.2.2

RX-VNE-2	[MUSS]	Die Nebenversionsnummer MUSS außerdem erhöht werden, wenn sich die Nebenversionsnummer (und nur diese!) eines referenzierten Schemas erhöht hat.	
RX-VNE-3	[SOLL)	Die Nebenversionsnummer SOLL auf '0' zurückgesetzt werden, wenn sich die Hauptversionsnummer ändert.	

4.9.1.3 Erhöhung der Hauptversionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VHE-1	[MUSS]	Die Hauptversionsnummer muss bei den Änderungen erhöht werden, die nicht durch eine Erhöhung der Revisionsnr. oder Nebenversionsnr. abgebildet werden können.	Die Änderungsgründe für die Hauptversionsnummer sind im Folgenden Kapitel aufgeführt: 5.2.3.2.3
RX-VHE-2	[MUSS]	Die Hauptversionsnummer MUSS erhöht werden, wenn sich die Hauptversionsnummer (und nur diese!) eines referenzierten Schemas erhöht hat.	

4.9.2 Regeln zur Vergabe/Erhöhung der logischen Versionsnummer

4.9.2.1 Erhöhung der logischen Revisionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VLR-1	[MUSS]	Die logische Revisionsnummer MUSS erhöht werden, wenn sich die Revisionsnummer mindestens eines der Schnittstelle zugrunde liegenden XML-Schema-Dateien erhöht (und nur diese!).	Die Änderungsgründe für logische Versionsnummern sind im Kapitel 5.2.4.1 aufgeführt.

4.9.2.2 Erhöhung der logischen Nebenversionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VLN-1	[MUSS]	Die logische Nebenversionsnummer MUSS erhöht werden, wenn sich die Nebenversionsnummer mindestens eines der Schnittstelle zugrunde liegenden XML-Schema-Dateien erhöht (und nicht die Hauptversionsnummer).	Die Änderungsgründe für logische Versionsnummern sind im Kapitel 5.2.4.1 aufgeführt.

4.9.2.3 Erhöhung der logischen Hauptversionsnummer

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen

RX-VLH-1	[MUSS]	Die logische Hauptversionsnummer MUSS erhöht werden, wenn sich die Hauptversionsnummer mindestens einer der für ein Verfahren verwendeten XML-Schema-Dateien erhöht.	Die Änderungsgründe für logische Versionsnummern sind im Kapitel 5.2.4.1 aufgeführt.

4.9.2.4 Validierung von logischen Versionsnummern

Nr.	Verbindlichkeit	Kurzbeschreibung	Referenzen
RX-VLV-4	[MUSS]	Der Verarbeitungsprozess zu einer Schnittstelle MUSS überprüfen, ob eine gegebene logische Versionsnummer zu der konkret verwendeten XML-Schema-Version kompatibel ist.	Grund und Vorgehensweise für die Validierung sind im Kapitel 5.2.4.2 aufgeführt.

5. Zusatzinformationen zu den Regeln

5.1 Zusatzinformationen Dateinamenskonvention

Dieses Kapitel dient dazu, eine verbindliche und verfahrensübergreifende Dateinamenskonvention für XML-Schema-Dokumente und XML-Schlüsseltabellen zu definieren, da diese eine wesentliche Grundlage für den Austausch und die Validierung verfahrensspezifischer Nutzdaten darstellen.

Die Benennung von XML-Nutzdatendateien[4] ist von dieser Konvention unberührt und wird weiterhin verfahrensspezifisch geregelt werden.

5.1.1 Dateinamen

Dateinamen dürfen aus Groß- und Kleinbuchstaben sowie Unterstrich '_', Punkt '.', und Bindestrich '-' (Minus) bestehen. Der Bindestrich ist als Trennelement reserviert. Umlaute und 'ß' DÜRFEN in Dateinamen NICHT verwendet werden, da sie nicht URL konform sind. Für Namen SOLLEN vorrangig deutsche Begriffe verwendet werden. Bereits aus dem Englischen in den Duden übernommene Begriffe wie ‚Plug-in' können aber beibehalten werden. Dateinamen besitzen eine vorgegebene Grundstruktur, die bei der Vergabe von Dateinamen zu berücksichtigen ist. Die folgenden Komponenten sind Teil der Grundstruktur:

[VK] Verfahrenskennung.

> Die Verfahrenskennung ist verpflichtend und beschreibt die logische Zugehörigkeit der Datei zu einem bestimmten Verfahren oder einer Verfahrenslandschaft. Die Verfahrenskennung „GI4X" ist für zentrale, d. h. alle Verfahren betreffende, Dateien reserviert. Darüber hinaus können durch Koordinationsgremien weitere verfahrensübergreifende Kennungen verabschiedet werden.

[4] Zu diesen zählen die verfahrensspezifischen Datenlieferungen, nicht jedoch von diesen Dokumenten referenzierte Schlüsseltabellen!

Verfahrensspezifische Kennungen besitzen den Aufbau/Name wie in Anlage 4 der Gemeinsamen Grundsätze Technik definiert und werden verfahrensintern festgelegt.

Beispiel: EMDP0 (Verfahrensspezifisch: Echtdaten-Datenaustausch „MDK Bereich Pflege“)

[QN] Qualifizierender Name.

Dieser Bestandteil ist verpflichtend und dient zur weiteren Unterscheidung von Dateien im Kontext einer Verfahrenskennung. Der QN darf dabei aus Groß- und Kleinbuchstaben sowie Unterstrich „_“ bestehen.

[VN] Versionsnummer.

Dieser Bestandteil ist verpflichtend und enthält die Versionsnummer der Datei, wobei die in dieser Empfehlung vorgegebenen Versionierungsvorschriften das Format der Versionsnummer vorgeben.

[LN] Laufende Nummer.

Dieser Bestandteil ist optional und dient dazu, Dateien zu identifizieren, die von einem ‚Master' Dokument (das keine [LN] Komponente hat) importiert/inkludiert/referenziert werden. Laufende Nummern beginnen bei 1.

[SUF] Suffix.

Dieser Bestandteil ist verpflichtend und beschreibt den Typ der Datei: „xml“ für Schlüsseltabellen und „xsd“ für XML-Schema-Dokumente.

Der Dateiname ergibt sich aus diesen Komponenten wie folgt: [VK]-[QN]-[VN]-[LN].[SUF]

D. h. bis auf [SUF] werden die Komponenten durch Bindestrich getrennt, das Suffix durch Punkt von den anderen Komponenten.

Hinweis: Die [VK] Komponente soll die Unterscheidung zwischen (E) Echt- und (T) Testdaten betrieb unterstützen.

5.1.2 Schema-Dateien (XSD)

Insbesondere für XSD-Dateien, MUSS eine verbindliche Dateinamenskonvention vorgegeben und eingehalten werden. [VK], [QN], [VN], und [SUF] sind verbindliche (MUSS) Namensbestandteile, [LN] ist

optional (KANN). Die Versionsnummernkomponente [VN] besteht aus der Haupt-, Neben-, und Revisionsnummer der Schemaversion. Der [QN] ‚basis' ist für verfahrensübergreifende oder verfahrensspezifische Schema-Definitionen reserviert, die von mehreren Schemata importiert werden:

Dateiname	Art	Beschreibung
GI4X-basis-[VN].xsd	MUSS	GKV Basis Schema: Dieses Schema enthält den kleinsten gemeinsamen Nenner an GKV weit zu verwendenden Schemadefinitionen (einfache/komplexe Typen, Elemente und Attribute), d. h. Definitionen, die für alle Verfahren relevant sind.
[VK]-basis-[VN].xsd [VK]-[QN]-[VN].xsd	KANN	Verfahrensübergreifendes/-spezifisches Basis Schema: Die so benannten Dateien enthalten gemeinsam genutzte Definitionen, die von anderen Schemata importiert werden – aber nur im Kontext eines Verfahrens oder einer Verfahrenslandschaft, z. B. MDK, Leistungserbringer oder Arbeitgeber sinnvoll sind. Neben dem QN ‚basis' können für verfahrensübergreifende Schemata auch noch andere QN Bezeichner verwendet werden.
[VK]-[QN]-[VN].xsd	MUSS	Verfahrensspezifisches Schema: Diese Schemata enthalten Nachrichtentyp-spezifische Schemadeklarationen für ein Verfahren (einfache und komplexe Typen, Elemente und Attribute). Die QN Komponente sollte dabei den Nachrichtentyp beschreiben.

Tabelle 1: Dateinamenskonvention für XSD-Dateien

Beispiele:

1. Die Datei ‚EAPO0-ABRP-1.2.0.xsd' enthält das fachspezifische Verfahrensschema Echtzeitbetrieb für APO, Nachrichtentyp ABRP, in der Schemaversion 1.2.0. Die Datei ‚EAPO0-ABRP-1.2.0-1.xsd' ist eine Subschemadatei dieses Schemas mit laufender Nummer 1, das von dem Hauptschema inkludiert wird.

2. Zwischen Krankenkassen und MDK (Medizinischer Dienst der Krankenkassen) werden demnächst zwei XML-Verfahren eingeführt: MDK-Krankenhaus und MDK-Pflege. Diese haben die Verfahrenskennungen EMDK0/TMDK0 und EMDP0/TMDP0. In beiden Verfahren werden zum Teil dieselben Schemadefinitionen benötigt. Um diesem Umstand gerecht werden zu können und redundante Schemateile zu vermeiden, könnte ein MDK Koordinationsgremium als verfahrensübergreifende [VK] für Dateinamen ‚MDK´ einführen. Die von beiden Verfahren gemeinsam zu nutzenden Schemadefinitionen könnten dann in einer ‚MDK-basis-1.0.0.xsd' Schemadatei verfahrensübergreifend definiert werden.

D. h. für Beispiel 2 könnte sich die folgende Import Hierarchie ergeben:

- ‚GI4X-basis-1.0.0.xsd' wird von allen MDK Schemadateien importiert.

- ‚MDK-basis-1.0.0.xsd' wird von allen MDK verfahrensspezifischen Schemadateien importiert, z. B. den Schemadateien ‚EMDK0-Beauftragung-1.0.0.xsd' und ‚EMDP0-Beauftragung-1.0.0.xsd'.

Eine Ausnahme von dieser Dateinamenskonvention gilt für sogenannte ‚Brücken'-Schemata. Brücken-Schemata werden notwendig, wenn man komplexe Typ-Definitionen aus einem Schema für ein anderes Schema (mit einem anderen Namensraum) einschränken will. Dies ist im XML-Schema nicht direkt im Zielschema möglich, sondern muss in einem Schema erfolgen, das denselben Zielnamensraum wie die komplexen Typen besitzt, die eingeschränkt werden sollen. Konzeptuell gehören diese eingeschränkten Typen somit zum Zielschema, vom Namensraum her aber zum Schema, das die Basistypen definiert.

Bei der Benennung solcher Brücken-Schemata wird dem Umstand, dass die enthaltenen Definitionen zu 2 Schemata gehören, durch ein Benennungsschema Rechnung getragen, das beide Schemata berücksichtigt:

[VK1]-[QN1]-[VN1]--[VK2]-[QN2]-[VN2].xsd

Der erste Teil „[VK1]-[QN1]-[VN1]" des Dateinamens MUSS dabei der Dateiname (ohne Suffix) des Schemas sein, aus dem die inkludierten Basistypen des Brückenschemas stammen. Der zweite Teil des Dateinamens [VK2]-[QN2]-[VN2] MUSS dabei der Dateiname (ohne Suffix) des Schemas sein, das die eingeschränkten Typen verwendet (importiert). Der Trenner zwischen den beiden Bestandteilen sind zwei ‚-' Bindestriche.

5.1.3 Schlüsseltabellen

Obwohl das Austauschformat von Schlüsseltabellen in der Empfehlung nicht weiter spezifiziert wird, werden externe Schlüsseltabellen bei der Modellierung von Identitätsconstraints (s. Kapitel 4.5) explizit eingeführt und daher ist es nötig, Konventionen für die Benennung von Schlüsseltabellendateien zu definieren.

Die Benennung von Schlüsseltabellen erfolgt nach den allgemeinen Dateinamenskonventionen. Die Komponenten [VK], [QN], [VN], und [SUF] sind verpflichtend, [LN] darf nicht verwendet werden. Der Qualifizierende Name [QN] enthält eine Beschreibung des Inhaltes der Schlüsseltabelle und muss mit ‚_keys' enden. Die Versionsnummer [VN] einer Schlüsseltabelle ist nicht strukturiert und besteht aus einer einfachen fortlaufenden Nummer [1...n].

Das folgende Beispiel nimmt an, dass GI4X verfahrensübergreifend Schlüsseltabellen für Betriebsnummern und Institutskennzeichen modelliert. Die zugehörigen Schlüsseltabellennamen könnten wie folgt vergeben werden:

GI4X-BN_keys-1.xml 1. Version der Liste von gültigen Betriebsnummern

GI4X-IK_keys-2.xml 2. Version der Liste von gültigen Institutskennzeichen

5.2 Zusatzinformationen Versionierung

5.2.1 Allgemeine Anforderungen und Festlegungen

GI4X und die davon abgeleiteten verfahrensspezifischen Schnittstellen können, aus fachlichen, organisatorischen und/oder technischen Gründen, unterschiedlichsten Änderungen und Anpassungen im Zeitablauf unterworfen sein. Daraus ergibt sich der Bedarf für eine einheitliche Konvention zur eindeutigen Bezeichnung von Schnittstellenversionen. In diesem Zusammenhang müssen auch evtl. mögliche Abwärtskompatibilitäten über Versionsbezeichner darstellbar sein.

Durch die Vergabe von einheitlichen Versionsnummern sollen insbesondere die Schemadateien zeitlich parallel nutzbar und referenzierbar sein. Weiterhin kann damit deren Verwendungszweck (externe und interne Nutzung) gekennzeichnet werden.

Die Versionierung von Schnittstellenversionen und Schemadateien bildet somit die Grundlage für die Erweiterbarkeit und Änderbarkeit von Schnittstellen. Darüber hinaus soll sie die nötigen Voraussetzungen für einen möglichen Mehrversionsbetrieb schaffen, das heißt die parallele Nutzung von Schnittstellenversionen ermöglichen.

5.2.2 Allgemeiner Aufbau und Vergabe von Versionsnummern

XML-Richtlinien kompatible Versionsnummern orientieren sich an den Vorgaben der [Gematik1] und besitzen folgenden Aufbau:

Bestandteil	Eigenschaft	Art	Beschreibung
Hauptversionsnummer (HVNR)	Numerisch, maximal 3 Stellen	MUSS	Die HVNR SOLL sich erhöhen, falls signifikante Änderungen durchgeführt werden, die zur aktuellen Version inkompatibel sind.
Nebenversionsnummer (NVNR)	Numerisch, maximal 3 Stellen	MUSS	Die NVNR SOLL sich erhöhen, falls Erweiterungen an der Schemadatei vorgenommen werden, die kompatibel zu den Schemadateien mit der gleichen HVNR sind.
Revisionsnummer (RENR)	Numerisch, maximal 3 Stellen	MUSS	Die RENR MUSS sich erhöhen, wenn die Änderungen für die Schemavalidierung irrelevant sind.

Tabelle 2: Versionierung - Allgemeiner Aufbau von Versionsnummern in Schemadateien

Nicht jede fachliche Änderung bedingt eine entsprechende Änderung an der Schemadatei. Hierzu zählen beispielsweise Änderungen an der Semantik bestimmter Inhalte, die zwar nicht zu Änderungen an der zugrundeliegenden Schemadatei führen, jedoch bei der Interpretation dieser fachlichen Inhalte zu berücksichtigen sind. Hierzu ist die logische Version zu verwenden. Folglich muss eine

Änderung der logischen Version nicht zwangsweise eine Änderung der Schemaversion bedeuten. Andererseits hat jede Änderung der HVNR oder NVNR der Schemaversion eine Änderung der logischen Version zur Folge.

Die logischen Versionen sind zu den Schemaversionen der korrespondierenden Schemadateien unterschiedlich, jedoch sind diese eindeutig über eine Zuordnungstabelle zuzuordnen.
Die Zuordnungstabelle MUSS zentral verwaltet und öffentlich bereitgestellt werden.

5.2.3 Schemadatei Versionierung

5.2.3.1 Änderungsaspekte

Die folgende Liste gibt einen Überblick über mögliche Änderungen an Schema-Dateien:

- Initiale Erstellung einer XML-Schema-Datei

- Änderungen an referenzierten Schema-Dateien
    - Das Schema wird dahingehend geändert, dass eine neue Version eines anderen Schemas importiert oder inkludiert wird.
    - Das Schema wird in mehrere Komponenten-Schemata aufgeteilt oder aber umgekehrt Komponenten-Schemata werden durch literales Einfügen aufgelöst.

- Änderungen an Datenstrukturen
    - Hinzufügen/Entfernen von Attributen und Elementen
    - Hinzufügen/Entfernen von Datentyp-Definitionen (Simple-Types, Complex-Types)
    - Ändern der Reihenfolge (Sequenz/Alternative)
    - Änderung der Kardinalität (Optionales und/oder mehrfaches Auftreten von Attributen/Elementen)

- Änderungen an Datentypen von Attributen und/oder Elementen
    - Redefinition des Datentyps (xs:Integer → xs:string)
    - Ersetzung mehrerer gleicher lokaler Typ-Definitionen durch einen global definierten Typ

- Änderung des Wertebereichs (Fassetten) eines Typs
    - Ein Attribut wurde initial als xs:string definiert, danach bekommt es bei einer Änderung einen einschränkenden regulären Ausdruck zugewiesen.
    - Ein Element vom Typ xs:int bekommt einen min/max value zugewiesen.
    - Der Enumerationstyp eines Attributs/Elementes wird durch Hinzufügen und/oder Wegnehmen von Einträgen geändert.

- Umgruppieren und Kommentieren
    - Die Schema-Datei wird neu formatiert und/oder Schemakomponenten in der Datei umgruppiert.
    - Es werden Kommentare oder Dokumentations-Annotationen in der Datei hinzugefügt, geändert, oder weggenommen.

5.2.3.2 Regeln zur Vergabe/Erhöhung von Versionsnummern

Im Folgenden wird aufgeführt unter welchen Aspekten die Bestandteile der Versionsnummer zu erhöhen sind. Hierbei werden die Vorgaben in [Gematik1] zur Vergabe von Versionsnummern von Schemadateien verfeinert und konkretisiert.

5.2.3.2.1 Erhöhung der Revisionsnummer

Zu den Änderungen, die zu einer Änderung der Revisionsnummer führen, gehören vor allem die Änderungen, die unter 'Umgruppieren und Kommentieren' beschrieben sind als auch das Aufteilen eines Schemas in mehrere Komponenten-Schemata und/oder das Auflösen von Komponenten-Schemata.

5.2.3.2.2 Erhöhung der Nebenversionsnummer

Die zu einer früheren Version kompatiblen Änderungen umfassen vor allem folgende validierungsrelevanten Aspekte:

- Das Ersetzen mehrerer gleicher lokaler Datentypdefinitionen durch eine Referenz auf einen globalen Datentyp mit der gleichen Definition wie die lokalen Datentypen.

- Das Zusammenfassen von Attributen in globalen Attributgruppen und die Verwendung der Gruppe anstatt der lokalen Attributdefinitionen.[5]

- Das Einschränken des Wertebereichs eines Datentyps (die Menge der zulässigen neuen Werte ist eine echte Teilmenge der Menge der Werte der zuvor zulässigen Werte. D. h. es dürfen keine Werte akzeptiert werden, die vorher nicht akzeptiert wurden)

- Die Einschränkung der Kardinalität entweder durch Erhöhen der min-occurence oder Erniedrigung der max-occurence (nicht jedoch umgekehrt). D. h. ein Attribut/Element das optional war, kann erforderlich gemacht werden - jedoch nicht umgekehrt.

- Das Entfernen von optionalen Attributen/Elementen aus dem Schema.

Die praktische Bedeutung der obigen Definitionen liegt darin, dass man die Grenze zwischen der Erhöhung der Nebenversionsnummer und Hauptversionsnummer anhand der Rückwärtskompatibilität der Instanzdokumente ziehen kann. Ein Instanzdokument, das valide zu einem Schema mit einer höheren Nebenversionsnummer ist, ist auch immer noch valide in Bezug auf die Schemata mit früheren Nebenversionsnummern. Der umgekehrte Fall gilt nicht!

5.2.3.2.3 Erhöhung der Hauptversionsnummer

Zu den Änderungen, die zu einer Änderung der Hauptversionsnummer führen, gehören vor allem:

- Das Hinzufügen von neuen Strukturen (z. B. Elemente/Attribute) und Datentypdefinitionen

[5] Im Gegensatz zum Umgruppieren von Definitionen werden durch die beiden Änderungen neue Typen/Definitionen zum Schema hinzugefügt, daher muss die Nebenversionsnummer erhöht werden.

- Das Ändern existierender Datenstrukturen (z. B. Umgruppierung der Elemente in einer Sequenz).

- Die Erweiterung der Kardinalität von Schemakomponenten (Erniedrigen der min-occurence oder Erhöhung der max-occurence).

- Das Entfernen vormals erforderlicher Strukturen (z. B. Elemente/Attribute)

- Die Erweiterung des Wertebereichs von Datentypen (Die Menge der akzeptierten Werte ist keine echte Teilmenge der vorher akzeptierten Werte mehr).

5.2.3.3 Geltungsbereich von Versionsnummern

Wie bereits bei den Namensräumen besprochen, gibt es eigene Geltungsbereiche für Echt- und Testdatenbetrieb. D. h., dass die Versionierung für beide Bereiche unabhängig voneinander erfolgt, und dass eventuell eine Schemaversion aus dem Testdatenbetrieb niemals in die Produktion übergeht.

5.2.4 Schnittstellen Versionierung (logische Version)

5.2.4.1 Änderungsaspekte

Die folgende Liste gibt einen Überblick über Änderungen, die einen Einfluss auf die Versionierung von Schnittstellen haben können:

- Schemaänderung
    - Ein der Schnittstelle zugrundeliegendes XML-Schema hat sich geändert.

- Änderung der Bedeutung von Werten
    - Die Interpretation eines ganzen Wertebereichs kann sich ändern (z. B. indem man die implizite Maßeinheit ändert: DM-> EUR, oder indem ein Schlüsselsystem durch ein anderes ausgetauscht wird: Krankenversicherungsnummer -> Sozialversicherungsnummer)

- Änderung von externen Daten
    - Zur Validierung und Verarbeitung von Verfahrensdaten werden oft externe Datenbestände herangezogen (z. B. Schlüsseltabellen), die regelmäßigen oder unregelmäßigen Änderungen unterworfen sein können.

- Veränderung des Verhaltens bei der Validierung/Verarbeitung
    - Die Regeln zur Validierung komplexer Sachverhalte und/oder Verarbeitung ändern sich, z. B. kann beschlossen werden, dass ein Fachverfahren beim Datenabgleich von Stammdaten keine Fehler mehr liefern darf, wenn Schlüsselinformationen nicht vollständig mit Detailinformationen übereinstimmen (Arztnummer -> falsche Adresse des Arztes, aber korrekte Bankverbindung)

Es gilt dabei zu beachten, dass obige Änderungen (außer dem ersten Punkt) keine Änderung der zugrundeliegenden Schemata erfordern und somit die Schemaversionsnummer alleine nicht ausreicht, um eine Schnittstellenversion eindeutig zu charakterisieren.

Aufgrund der Zielstellung dieses Dokumentes können jedoch nur Regeln zur Vergabe von logischen Versionsnummern definiert werden, die durch Änderung von Schemaversionen erforderlich sind. Die Regeln zur Änderung der logischen Versionsnummern aufgrund der Änderung der Bedeutung von Werten, Änderung von externen Daten oder der Veränderung des Verhaltens bei der Validierung/Verarbeitung werden hier nicht vorgegeben und sind vom Verfahren festzulegen. Die Regeln zur Vergabe und Erhöhung der logischen Versionsnr. sind im Kapitel 4.9.2 aufgelistet.

5.2.4.2 Validierung von Versionsnummern

Die Beziehung zwischen logischer Schnittstellenversionierung und Schemaversionierung erfordert eine Prüfung durch die Schnittstellenanwendungen. Diese Prüfung muss feststellen, ob eine gegebene logische Versionsnummer kompatibel zu dem Schema des Root-Elementes ist, in dem sie verwendet wird.

Dazu ist es notwendig für jede logische Version eine Abbildung der zu ihr kompatiblen Schema-Versionen zu definieren. Da sich bei jeder signifikanten Änderung des Schemas auch die logische Version ändert, kann man die Abbildung eindeutig definieren

5.2.5 Beispiel Zuordnung logische Versionsnummer/Schemaversionierung

Das folgende Beispiel dient dazu, die Abhängigkeiten zwischen logischer Versionierung und Schemaversionierung zu illustrieren. Die in der rechten Tabellenhälfte angegebenen Versionsnummern resultieren aus der links angegebenen Aktivität. Das Resultat der jeweiligen Aktivität auf die Versionierung ist grafisch hervorgehoben.

Aktivität	Schema-version	Namens-raum-ver-sion	Logische Version
1. Erstellung der Schemadatei	**1.0.0**	**1.0**	**1.0.0**
2. Fachliche Änderung hinsichtlich der inhaltlichen Bedeutung von Elementen/Attributen. Die fachspezifische Datenstruktur bleibt davon unberührt. Beispiel: Änderungen/Erweiterungen an externen Schlüsseltabellen (Einschränkung zulässiger Werte)	1.0.0	1.0	**1.0.1**
3. Hinzufügen von Typeinschränkungen Beispiel: Enumerationen (Einschränkung zulässiger Werte)	**1.1.0**	**1.1**	**1.1.0**
4. Hinzufügen neuer Strukturen (basierend auf bestimmten fachlichen Änderungen)	**2.0.0**	**2.0**	**2.0.0**
5. Fachliche Änderung hinsichtlich der inhaltlichen Bedeutung von Elementen/Attributen. Die fachspezifische Datenstruktur bleibt davon unberührt.	2.0.0	2.0	**3.0.0**

Beispiel: Ein Attribut beinhaltet nun andere Informationen: Als Schlüsselattribut enthält ‚key' einen eindeutigen Wert zur Herstellung eines Personenbezugs. Bis einschließlich zur logischen Version 2.0.0 wurde damit die Rentenversicherungsnummer transportiert. Nun wird die KV-Nummer verwendet.			
6. Aus 5. wird der lexikalische Ausdruck in Form einer Regular Expression angepasst.	**2.1.0**	2.1	**3.1.0**
7. Aus 5. werden die weiterführenden Prüfroutinen angepasst. (Außerhalb der Schemadatei)	2.1.0	2.1	**3.1.1**

Daraus ergibt sich die folgende Zuordnungstabelle zwischen logischer Version und Schema-Version:

Logische Version	Schema Version
1.0.0, 1.0.1	1.0.0
1.1.0	1.1.0
2.0.0, 3.0.0	2.0.0
3.1.0, 3.1.1	2.1.0

6. Literaturverzeichnis

[Gematik1]	Spezifikation von Versionsnummern in Schnittstellenspezifikationen und Software-Komponenten, https://fachportal.gematik.de/fileadmin/user_upload/fachportal/files/Spezifikationen/Basis-Rollout/Architektur_und_uebergreifende_Dokumente/gematik_GA_Spezifikation_Versionsnummern_V1_1_0.pdf
[RFC 2119]	Best Current Practice - Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt
[RFC 2396]	Uniform Resource Identifiers (URI): Generic Syntax,http://www.ietf.org/rfc/rfc2396.txt
[Whitespace]	http://www.w3.org/TR/xmlschema-2/#rf-whiteSpace
[XML-Schema]	http://www.w3.org/XML/Schema