Kassenärztliche
Bundesvereinigung
Körperschaft des öffentlichen Rechts

ehd – eHealthData

Richtlinie

Grundstrukturen, Regeln und
Namensgebung beim Entwurf
von XML-Schnittstellen

Dezernat 5

IT-Bereich

10623 Berlin, Herbert-Lewin-Platz 2
50859 Köln, Ottostraße 1

Version: 1.40

Datum: 15.02.2006

Freigabe: 15.02.2006

ÄNDERUNGSVERZEICHNIS

Version	Datum	Autor	Änderung	Begründung	Seite
1.40	14.02.06	Usorov	Kommentar ergänzt: Zusätzliche Angabe von Zeit und Zeitzone bei originati-on_dttm ist möglich	entsprechend W3C-Spezifikation	20
1.40 Entwurf	16.08.05	Usorov	xs:decimal durch xs:string bei ehd_version_typ ersetzt. In local_header-cont.model und ehd_body_typ <se-quence> durch <choice> bei any-Elementen ersetzt. Hinweis zu Schemadatei-namen. Eine feste OID bei Bearbei-tungszustand entfällt. In Ehd-Dateinamen wird das Trennzeichen „#“ durch „+“ eretzt. In keytabs wird neues opti-onales Element fkey hinzu-gefügt. Namensgebung für gezippte Archive (KV-DTA)	Probleme mit XML-Spy Probleme mit XML-Spy. Verbesserte Versionierung der Schematadateien. Der Bearbeitungszustand kann in jedem Projekt eigen definiert wer-den. Probleme mit Java-Programmen und config-Daeien. Es wird für die Veröffentlichung der Schlüsseltabellen benötigt. gezippte Archive können der KV-DTA Richtlinie entsprechen	13 43 52 40 50 48 52
1.30	10.01.05	Usorov	OID-Änderungen Hinweis zur Namensgebung der ehd-Dateien.	Die OIDs wurden an das „OID-Konzept für das deutsche Gesund-heitswesen“ angepasst.	63 50
1.20	09.06.04	Usorov	Jedes Header-Element wurde an CDA-Header angenähert. Anpassung Dateinamesge-bung Neu: Anleitung für Schnitt-stellenentwickler		

Version	Datum	Autor	Änderung	Begründung	Seite
			Neu: Schlüsseltabellen Anpassung Namespaces Kollektionen sind nicht mehr obligatorisch		
1.00	02.04.04	Sattler (Heraus-geber)	neues Dokument	Nach RFC-Phase und Workshop unter Mitarbeit von Beringuier-Manhart, Greve, Obermeyer, Renz, Treichel, Usorov, Weiler, Welteroth, Wieseke. Umfangreiche Auszüge aus der Richtlinie XML-Schnittstellen und Richtlinie XML-Stammdateien der KBV wurden übernommen.	

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

XML-CODE-VERZEICHNIS

TABELLEN-VERZEICHNIS

1 Einführung

Im Zuge der Umstellung von xDT- zu XML-Schnittstellen werden im Bereich der Kassenärztlichen Vereinigungen viele neue Schnittstellen für den Datenaustausch und die Datenspeicherung definiert. Damit es keinen „Wildwuchs“ gibt, braucht man ein übergreifendes Konzept.

Leider kann CDA von SCIPHOX (=CDA 1.0 von HL7) nicht direkt für alle neu zu entwerfenden Schnittstellen genommen werden, da bei CDA immer ein Patient als Bezugspunkt der Datei gefordert wird, aber bei den Datenlieferungen nicht immer ein Patient relevant ist. Wenn z.B. „Meldedaten“, also Informationen über den Einsatz von Praxiscomputersystemen, von einer KV an die KBV (zum Zweck der statistischen Erfassung) geliefert werden müssen, lässt sicht beim besten Willen kein Patient unterbringen. Für sogenannte Massendaten lässt sich SCIPHOX also nicht verwenden. Es wäre auch nicht möglich, innerhalb der SCIPHOX-Arbeitsgruppe den CDA-Header so anzupassen, dass er für Massendaten geeignet ist, weil CDA direkt von HL7 übernommen wurde und, abgesehen von der zeitlichen Verfügbarkeit, auf die Weiterentwicklung des amerikanischen Standards HL7 keine planbare Einflussnahme möglich ist.

Die Hauptintention von ehd ist es, einen Rahmen zu schaffen, in dem beliebige XML-Schnittstellen entstehen können, für die der CDA-Header von SCIPHOX ungeeignet ist. Dabei wird Wert darauf gelegt, weitgehend kompatibel zu SCIPHOX zu sein: Von begründeten Ausnahmen abgesehen, werden für alle ehd-Schnittstellen die Elemente aus SCIPHOX übernommen. Die Weiterentwicklung der ehd-Richtlinie ist von den Entwicklungen bei SCIPHOX abhängig. Wenn in SCIPHOX bedingt durch Enwicklungen bei HL7 mit CDA 2 und HL7-V3 die Möglichkeiten für Massendatentransport etabliert werden, so ist schon jetzt vorsagbar, dass die ehd-Richtlinie vollständig zu SCIPHOX konvergieren wird.

ehd ist also ein Familienkonzept und kein eigener Schnittstellenstandard wie SCIPHOX. ehd ist auch keine Konkurrenz zu SCIPHOX, sondern eine Ergänzung für alle die Fälle, bei denen SCIPHOX nicht genommen werden kann.

Das vorliegende Dokument ist geschaffen worden, um Schnittstellenentwicklern eine Richtlinie bereitzustellen, nach der sie eine XML-Schnittstelle entwickeln können, die (bei vollständiger Beachtung) dann zur ehd-Schnittstellenfamilie gehört. Eine ehd-Schnittstelle kann zu den unterschiedlichsten Inhalten erfunden werden - zur Familie der ehd-Schnittstellen zählt sie dann, wenn sich der Schnittstellenerfinder an die aktuelle Version der ehd-Richtlinie gehalten hat.

ehd steht für eHealth-Data. Die ehd-Richtlinie ist bewusst so gestaltet, dass sie nicht an eine Institution gebunden ist.

2 Semantik der verwendeten Diagrammsymbole

Zur Visualisierung der verwendeten XML-Schemata werden Diagramme verwendet, deren Symbole in den folgenden Kapiteln kurz erläutert werden.

2.1 Kardinalität

Es existieren verschiedene Kardinalitäten:

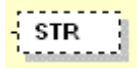
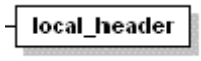
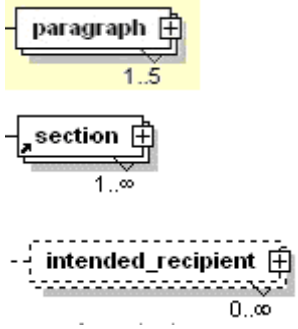
Kardinalität	Symbol	Beschreibung
0..1		**Optionale Elemente** Ein optionales Element wird als Rechteck mit gestrichelter Linie dargestellt. Es kann keinmal oder einmal vorkommen.
1		**Obligatorische Elemente** Elemente, welche als Rechteck mit durchgezogener Linie dargestellt sind, müssen genau einmal vorkommen.
n...m		**Mehrfache Elemente** Bei Elementen, welche mehrfach vorkommen können, wird die erlaubte Anzahl rechts unter dem Symbol dargestellt. Die Werte können von 0 bis ∞ (unbounded) reichen.

Tabelle 1 Beschreibung der Kardinalitäten eines XML-Elements

2.2 Strukturelemente

Die Elemente eines Schema-Diagramms werden über sogenannte Strukturelemente miteinander verknüpft. In diesem Dokument werden zwei Strukturelemente verwendet: ***<xs:choice>*** und ***<xs:sequence>***.

Symbol	Beschreibung
	Das Strukturelement ***<xs:choice>*** zeigt an, dass zwischen verschiedenen Kindelementen genau eins ausgewählt werden muss.
	Das Strukturelement ***<xs:sequence>*** beschreibt, dass die Kindelemente in festgelegter Reihenfolge aufgeführt werden müssen.

Tabelle 2 Beschreibung der Strukturelement-Symbole

2.3 Sonstige Symbole

Es werden außerdem folgende Diagramm-Symbole verwendet:

Symbol	Beschreibung

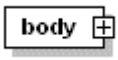
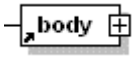
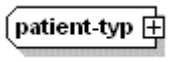
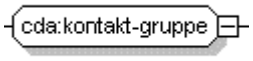
	**Element mit Kindelementen** Ein Element mit einem oder mehreren Kindelementen wird durch ein Pluszeichen am Rechteckrand symbolisiert.
	**Referenzelement** Der Pfeil links unten im Element zeigt an, dass das Element an anderer Stelle im Schema definiert wurde.
	**Datentyp** Ein Rechteck mit zwei abgeflachten Ecken links symbolisiert einen Datentyp.
	**Gruppenelement** Ein Rechteck mit vier abgeflachten Ecken stellt ein Gruppenelement dar, welches mehrere Elemente zusammenfasst.

Tabelle 3 Beschreibung sonstiger Symbole

3 ehd-Grundstruktur

Für die XML-Dateien ist der Zeichensatz ISO-8859-1 vorgeschrieben. Bei allen Elementen, die in diesem Dokument beschrieben werden, ist es wichtig, die Groß-/Kleinschreibung zu beachten.

Grundsätzlich besteht eine ehd-Datei aus dem Wurzelelement **<*ehd*>**, welches sich aus den beiden Kindelementen **<*header*>** und **<*body*>** zusammensetzt, wie es in Abbildung 1 dargestellt ist.

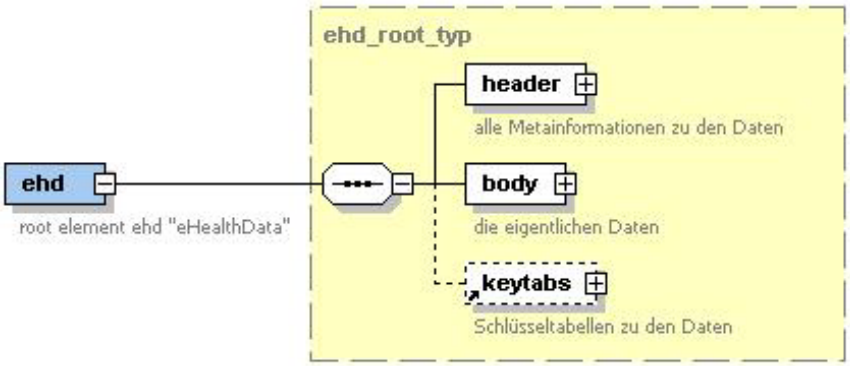


Abbildung 1 /ehd (Root-Element)

Die Platzhalter (entsprechen immer drei Punkten "...") müssen durch die in den folgenden Abschnitten beschriebenen Kindelemente von **<*header*>** und **<*body*>** ersetzt werden.

Das **<*ehd*>** - Element hat folgende Elemente/Attribute:

Kardinalität	1..1
children	**header (1..1)** **body (1..1)**

	keytabs (0..1)					
attributs	Name	Type	Use	Default	Fixed	Annotation
	ehd_version	xs:string	required			

ehd_version: Im XML-File wird die Versionsnummer zur zugrundeliegenden ehd-Richtlinie bzw. des verwendeten ehd-Schemas angeben. Der Wertebereich wird auf 0.00 bis 99.99 festgelegt, anderenfalls wird der Parser Fehler melden.

Um die Aufwärtskompatibilität zu gewährleisten, wird kein fester Wert für die Version vorgegeben. Die Aufwärtskompatibilität ist dann gegeben, wenn kleine optionale Änderungen am ehd-Schema gemacht werden, so dass früher erstellte XML-Dateien trotzdem ohne Fehler gelesen werden können. Im Schema, im Attribut version ist die aktuelle Schema-Version angegeben, so dass Entwickler selbst entscheiden können, ob bestimmte Versionen akzeptiert werden, oder nicht. Bei einem festvorgegebenem ehd_versions-Wert müssten alle XML-Dateien aktualisiert werden, auch wenn die Änderungen nicht diese Dateien betreffen.

Der Schnittstellenentwickler kann also selbst entscheiden, ob im Schema das Attribut auf Übereinstimmung mit einer bestimmten ehd-Richtlinienversion oder auf „>“, „<“,oder „=“ oder beliebig geprüft wird.

<header> Der Header ist ein Pflichtelement, hier befinden sich die Metadaten zu den im body liegenden eigentlichen Inhaltsdaten. Die Grundstruktur wird im Kapitel 4 vorgestellt.

<body> Hier liegen die eigentlichen Inhalte der Datenlieferung. In diesem Bereich kann der Schnittstellenerfinder seine eigenen Strukturen definieren, wobei die im Abschnitt 8 befindlichen Designregeln beachtet werden müssen. Die Grundstruktur wird im Kapitel 5 vorgestellt.

Der Namensraum für die ehd-Schnittstelle ist zwingend vorgeschrieben:„***urn:ehd/001***“. Der Namensraumkonzept wird im Kapitel 7 vorgestellt.

Folgender Code ist für die Implementierung einer ehd-Schnittstelle vorgeschrieben:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ehd xmlns="urn:ehd/001" ehd_version="...">
    <header>
    ...
    </header>
    <body>
    ...
    </body>
    <keytabs>
    ...
    </keytabs>
</ehd>
```

XML-Code 1 /ehd (Root-Element)

4 header (Metadaten)

Der Header enthält die Metadaten zu den im body liegenden eigentlichen Inhaltsdaten, er hat folgende Grundstruktur:

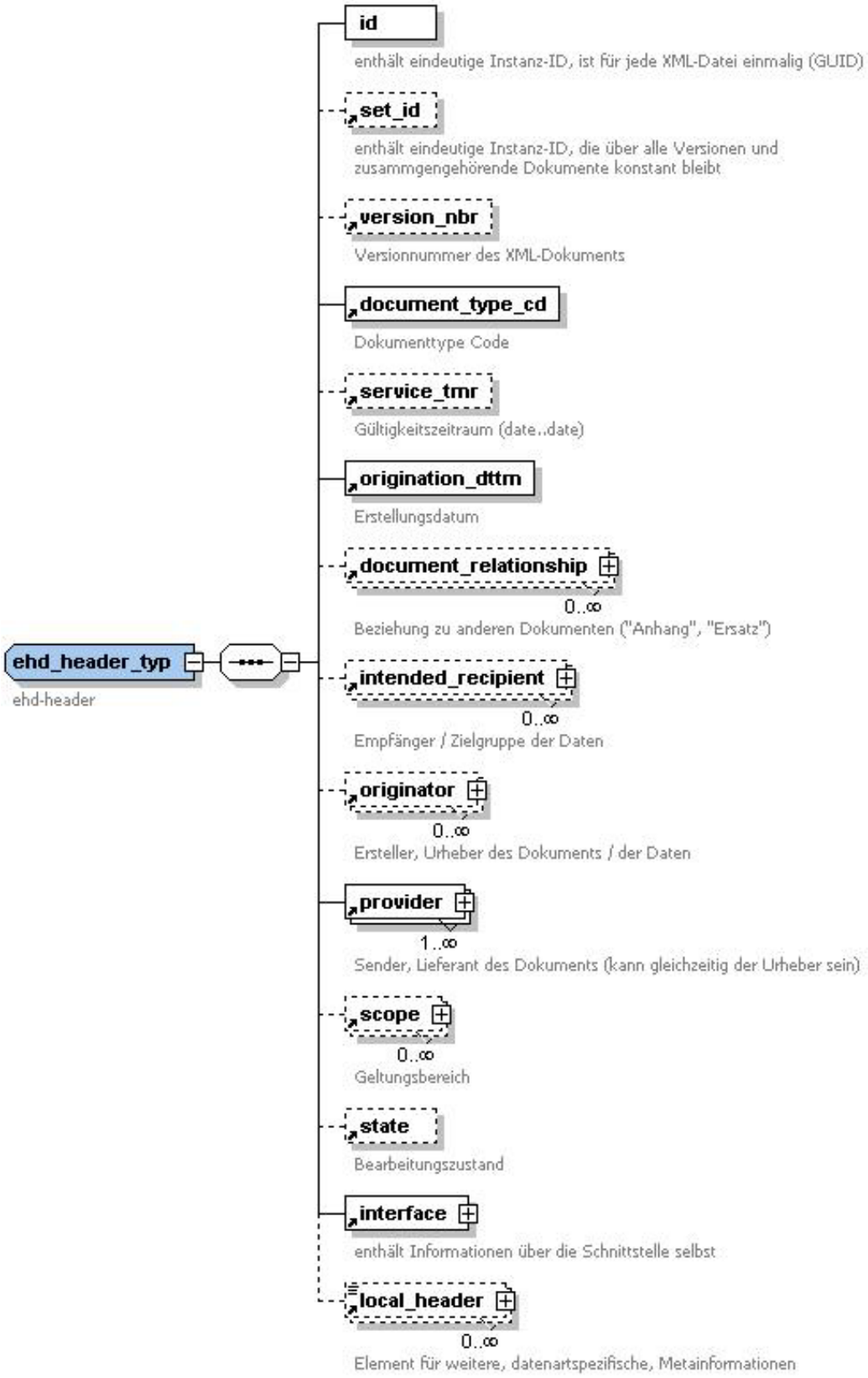


Abbildung 2 /ehd/header

Folgendes Beispiel zeigt die Grundstruktur des **<header>** -Elements. Ein komplett ausgefüllter Beispiel mit Daten befindet sich in der XML-Datei „ehd_beispiel.xml“.

```
<header ehd_version=". . .">
    <id EX=". . ." RT=". . ."/>
    <set_id EX=". . ." RT=". . ."/>
    <version_nbr V=". . ."/>
    <document_type_cd V=". . ."/>
    <service_tmr V=". . ."/>
    <origination_dttm V=". . ."/>
    <document_relationship>
        . . .
    </document_relationship>
    <intended_recipient>
        . . .
    </intended_recipient>
    <originator>
        . . .
    </originator>
    <provider>
        . . .
    </provider>
    <scope>
        . . .
    </scope>
    <state V=". . ."/>
    <interface>
        . . .
    </interface>
</header>
```

XML-Code 2 /ehd/header

4.1 id (Dokument-ID)

Das Element **<id>** ist ein eindeutiger Instanz-Identifikator, mit welchem jedes XML-Dokument bzw. jede XML-Datei weltweit identifiziert werden kann. Jede XML-Datei hat eine andere id, und auch bei Korrekturen muss eine neue id vergeben werden.

Das Element hat folgende Attribute:

Kardinalität	1..1					
attributs	Name	Type	Use	Default	Fixed	Annotation
	EX	xs:string	optional			
	RT	xs:string	required			
	RTV	xs:string	optional			

EX: („extension“=“Erweiterung“ zu dem RT-Attribut) dieses Attribut enthält eine OID bzw. Nummer des Objektes, welches das Objekt innerhalb der Organisation bzw. eines Bereichs eindeutig identifiziert. Zusammen mit dem RT-Attribut ergibt sich ein weltweit einmaliger Identifikator.

Es wird empfohlen im EX-Attribut einen Identifikator in Form eines GUID (global unique identifier), zu verwenden. Jede, auf Basis dieser Beschreibung, erstellte konkrete XML-Datei muss daher einen anderen Identifikator haben.

Bei einer GUID handelt sich um eine 128 Bit-Zahl, die aus einer Menge von Informationen gebildet wird. Zum Erzeugen von GUID gibt es zahlreiche Tools, auch mit Java

gibt es solche Werkzeuge. Eine GUID ist z.B. nach den Microsoft-Vorgaben „XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX“ aufgebaut, wobei die 128 Bit-Zahl in hexadezimal dargestellt wird. Damit entspricht X einem beliebigen Buchstaben aus der Menge (a-f,A-F) oder einer der Ziffern 0-9.

RT: („root“=“Wurzel“) dieses Attribut enthält eine OID/Nummer der Organisation, welche weitere OIDs bzw. Nummern ihr unterstehender Objekte vergibt. Hat ein Objekt eine Nummer erhalten, wird diese im EX-Attribut angegeben.

RTV: („Schlüsseltabelle Version“). Dieses Attribut ist optional und wird nur bei Schlüsseltabellen verwendet, wenn eine Referenzierung von Objekten über Schlüsseltabellen erfolgt. Für die Identifizierung von XML-Dokumenten wird dieses Attribut nicht verwendet.

```
<id EX="0123A5Z7-89BB-2rt5-67er-0123zeuA7890" RT="123.345.66446.3"/>
```

XML-Code 3 /ehd/header/id

4.2 set_id (Set-Kennung)

Wenn mehrere Dokumente bzw. Datenlieferungen zu einer logischen Einheit gehören, ist es möglich eine logische Kennung für das Set zu vergeben. Damit wird signalisiert, dass mehrere Dokumente miteinander in Beziehung stehen. Die ***<set_id>*** bleibt über mehrere Versionen und Unterschiedlichen zusammengehörende Dokumente konstant. Die Struktur gleicht dem ***<id>*** Element, es muss jedoch keine GUID im EX-Attribut angegeben sein, weil im EX- und RT-Attribut beliebige Zeichenfolgen als logische Kennung verwendet werden können.

Das Element hat folgende Attribute:

Kardinalität	0..1					
Attributs	Name	Type	Use	Default	Fixed	Annotation
	EX	xs:string	optional			
	RT	xs:string	required			
	RTV	xs:string	optional			

EX: („extension“=“Erweiterung“ zu dem RT-Attribut), beliebige Zeichenfolge

RT: („root“=“Wurzel“). Basiswert, beliebige Zeichenfolge

RTV: („Schlüsseltabelle Version“) Dieses Attribut ist optional und kann verwendet werden, wenn die Referenzierung über Schlüsseltabellen erfolgt. Für die Identifizierung von logischen Einheiten wird dieses Attribut nicht verwendet.

```
<set_id EX="121223" RT="go2342A"/>
```

XML-Code 4 /ehd/header/set_id

4.3 version_nbr (Versionsnummer)

Das Element **<*version_nbr*>** enthält die Versionsnummer der Datenlieferung.

Das Element hat folgende Attribute:

Kardinalität	0..1					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:nonNegativeInteger	required			

V: Im V-Attribut steht die Versionsnummer, bei der Erstlieferung erhält das V-Attribut die Ziffer „1“. Für jede nachfolgende Austausch-, Korrektur- u.a. Lieferung wird die Versionsnummer um eins hochgezählt.

```
<version_nbr V="1"/>
```

XML-Code 5 /ehd/header/version_nbr

4.4 document_type_cd (Bezeichnung des Datentyps)

Das Element **<*document_type_cd*>** beschreibt den Dokumenttype, d.h. um welche Schnittstelle bzw. Satzart es sich handelt. Das Element ist dafür ausgelegt um kodierte Werte darzustellen.

Kode-Elemente:

Genereller Aufbau von Elementen, die kodierte Werte und Schlüsseltabellen enthalten, am Beispiel des Elements **<*document_type_cd*>**. Gemäß der Namenskonvention haben Elemente, die kodierte Werte enthalten, in der Regel die Endung _cd. Kode-Elemente haben den Datentyp **v_s_string_typ**, der folgende Attribute enthält:

Das Element hat folgende Attribute:

Kardinalität	1..1					
children						
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:string	required			
	DN	xs:string	optional			
	S	xs:string	optional		1.2.276.0.76.2.2.100	
	SN	xs:string	optional			
	SV	xs:string	optional			

V: Kürzel, kodierter Wert

DN: menschenlesbarer Klartextname des Wertes

S: OID der Schlüsseltabelle, in der kodierte Werte verwaltet werden

SN: menschenlesbarer Klartextname der Schlüsseltabelle

SV: Version der Schlüsseltabelle, wenn die Schlüsseltabelle geändert bzw. ergänzt wird, wird die Version hochgezählt. Es ist empfehlenswert die Version der Schlüsseltabelle anzugeben,

um auch nach mehreren Schlüsseltabellenänderungen noch feststellen zu können, wann ein Wert gültig war.

Bei dem Element <document_type_cd> wird im V-Attribut das Kürzel (Kode) der Schnittstelle bzw. Satzart eingetragen, im S-Attribut steht die Schlüsseltabelle, in der alle Kodes verwaltet werden.

```
<document_type_cd V="DE.SDKT" DN="Kostenträgerstammdatei" S="1243.23.523.1T.e.s.t"
            SN=" KBVSchnittstellen " SV="1.0"/>
```

XML-Code 6 /ehd/header/document_type_cd

4.5 service_tmr (Gültigkeitszeitraum)

Das Element **<*service_tmr*>** beschreibt den Gültigkeitszeitraum der enthaltenen Daten.

Das Element hat folgende Attribute:

Kardinalität	0..1					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	zeitraum_typ	required			

V: hier wird der Gültigkeitszeitraum z.B. im Format „YYYY-MM-DD..YYYY-MM-DD“ angegeben. Das erste Datum steht dabei für den Anfang, das zweite Datum für das Ende des Gültigkeitszeitraums. Die beiden Angaben werden durch zwei Punkte voneinander getrennt

Es sind folgende Zeitraumangaben möglich:

YYYY-MM-DD..YYYY-MM-DD	gilt von YYYY-MM-DD bis YYYY-MM-DD
..YYYY-MM-DD	gilt bis YYYY-MM-DD
YYYY-MM-DD..	gilt ab YYYY-MM-DD bis auf weiteres
YYYY-MM-DD	gilt am YYYY-MM-DD

Y.. Jahreswert, M.. Monatswert, D.. Tageswert

Achtung: Hier gibt es eine Abweichung zur Notation im entsprechenden timeframe-Namepart bei der Dateinamensgebung (s. Kapitel 7). Die Sonderfälle, die dort möglich sind (Quartal, Monat, Woche) sind bei **<*service_tmr*>** bewusst nicht erlaubt.

```
<service_tmr V="2003-11-01..2003-11-22"/>
```

XML-Code 7 /ehd/header/service_tmr

4.6 origination_dttm (Erstellungsdatum)

Das Element <*origination_dttm*> beschreibt das Erstellungsdatum der Datei.

Das Element hat folgende Attribute:

Kardinalität	1..1					
attributs	Name V	Type xs:date	Use required	Default	Fixed	Annotation

V: hier wird das Erstellungsdatum der Datei im Format „YYYY-MM-DD“ angegeben. Zusätzlich kann auch die Zeit und Zeitzone angegeben werden (siehe date-Datentyp in W3C-Schema-Spezifikation [3]).

```
<origination_dttm V="2003-09-30"/>
```

XML-Code 8 /ehd/header/origination_dttm

4.7 document_relationship (Beziehungen zu anderen Dokumenten)

Durch das Element <*document_relationship*> werden die Beziehungen („Anhang“ oder „Ersatz“) zu anderen Dokumenten repräsentiert. Bei Verweisen ist es empfehlenswert die Elemente <*set_id*> und <*document_relationship*> anzugeben.

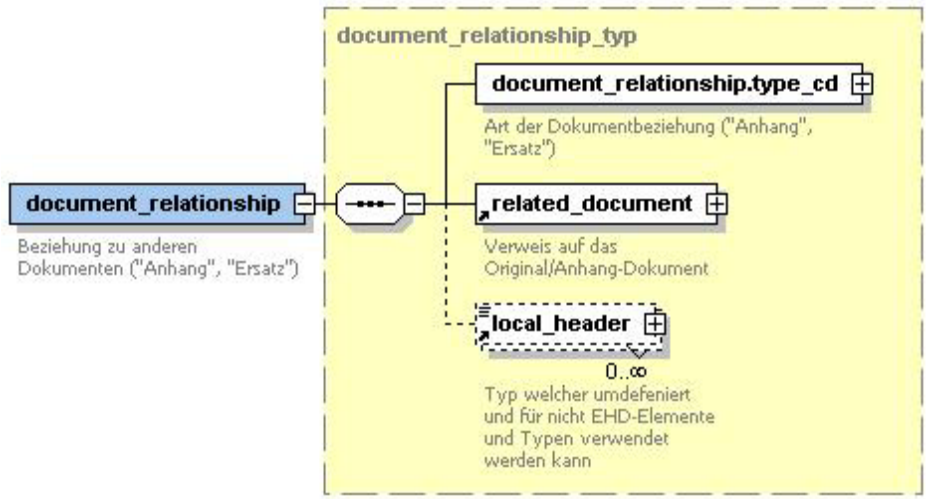


Abbildung 3 /ehd/header/document_relationship

Das Element hat folgende Attribute:

Kardinaldiät	0..n					
children	**document_relationship.type_cd (1..1)** **related_document (1..1)** **local_header (0..n)**					
attributs	Name	Type	Use	Default	Fixed	Annotation

<document_relationship.type_cd>, Der Beziehungstyp wird in diesem Kode-Element festgelegt.

<related_document>, Die Verbindung zu anderen Dokumenten erfolgt über dieses Element, in dem die Referenz zum anderen Dokument, über die Dokument-Identifikatoren (**<id>**, **<set_id>**, **<version_nbr>**), angegeben wird.

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<document_relationship>
    <document_relationship.type_cd V="APND" DN="Append/Anhang" S="2.32.442.3"
                SN="Dokumentbeziehungstypen" SV="1.0"/>
    <related_document>
        <id EX="0123A5Z7-89BB-2rt5-67er-0123zeuA7890" RT="123.345.66446.3"/>
        <set_id EX="34453" RT="St_kbv"/>
        <version_nbr V="1" />
    </related_document>
</document_relationship>
```

XML-Code 9 /ehd/header/document_relationship

4.7.1 document_relationship.type_cd (Dokument-Beziehungstyp)

Im Element **<document_relationship.type_cd>** wird der Typ der Beziehung („Anhang“ oder „Ersatz“) zum Dokument, der im Element **<related_document>** angegebenen ist, genannt.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
<document_relationship.type_cd V="APND" DN="Append/Anhang" S="1.2.276.0.76.2.2.101"
                SN="Dokumentbeziehungstypen" SV="1.0"/>
```

XML-Code 10 /ehd/header/document_relationship/document_relationship.type_cd

4.7.2 related_document (Verweis auf anderes Dokument)

Im Element **<related_document>** wird der Verweis auf das Dokument angegeben, mit dem die Daten in Verbindung stehen.

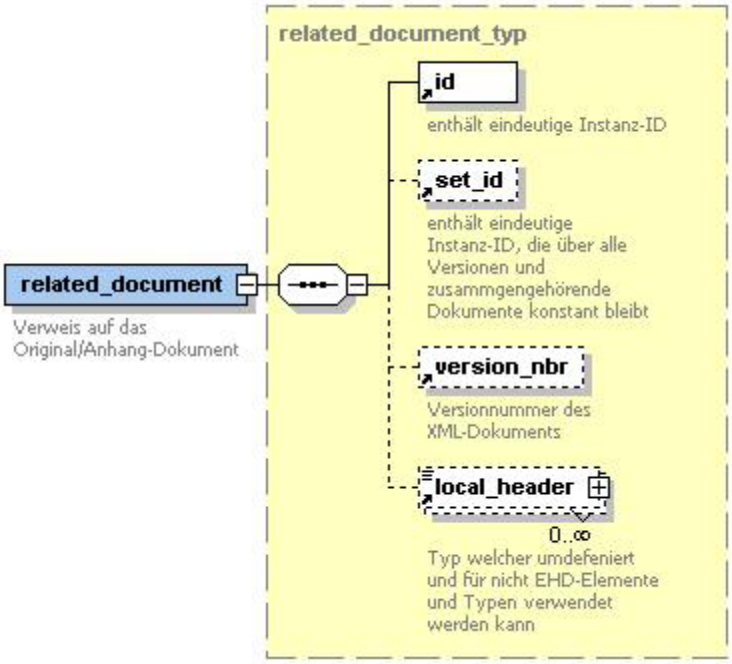


Abbildung 4 /ehd/header/document_relationship/related_document

Das Element hat folgende Attribute:

Kardinalität	1..1					
children	**id (1..1)** **set_id (0..1)** **version_nbr (0..1)** **local_header(0..n)**					
attributs	Name	Type	Use	Default	Fixed	Annotation

Die Identifikation des Verweisdokuments erfolgt über die Dokument-Identifikatoren:

<id>, siehe: id (Dokument-ID), eine GUID muss nicht gebildet werden.

<set_id>, siehe: set_id (Set-Kennung)

<version_nbr>, siehe: version_nbr (Versionsnummer)

<local_header> - Element wird in <local_header> erklärt.

```
<related_document>
    <id EX="0123A5Z7-89BB-2rt5-67er-0123zeuA7890" RT="123.345.66446.3"/>
    <set_id EX="34453" RT="St_kbv"/>
    <version_nbr V="1" />
</related_document>
```

XML-Code 11 /ehd/header/document_relationship/related_document

4.8 intended_recipient (Empfänger / Zielgruppe der Daten)

Das Element <intended_recipient> enthält Angaben zum Empfänger der Daten bzw. zur Zielgruppe.

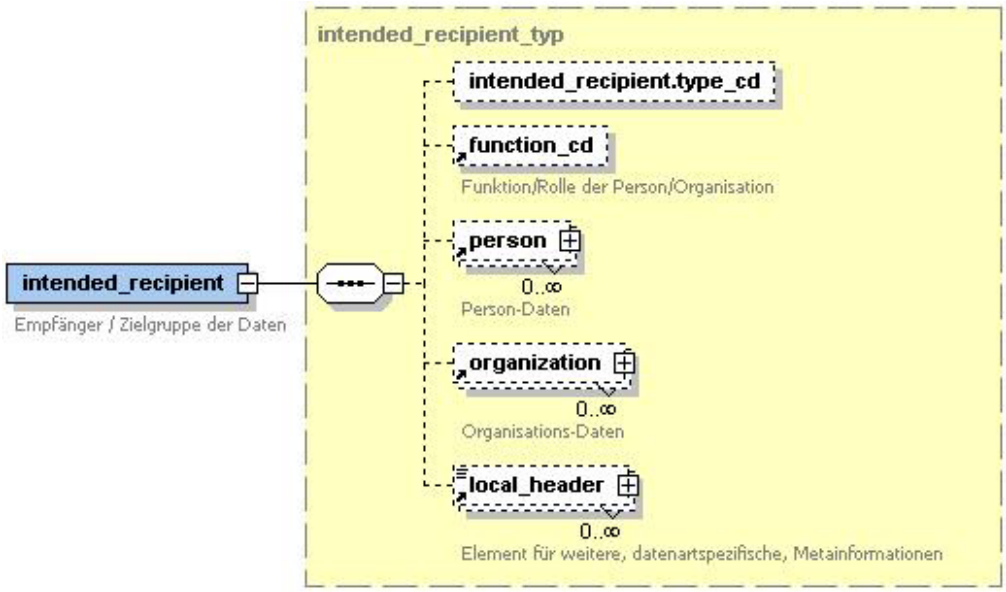


Abbildung 5 /ehd/header/intended_recipient

Das Element hat folgende Attribute:

Kardinalität	0..n
children	**intended_recipient.type_cd (0..1)** **function_cd (0..1)** **person (0..n)** **organization (0..n)** **local_header(0..n)**
attributs	Name Type Use Default Fixed Annotation

<intended_recipient.type_cd>, Der Empfänger bzw. die Zielgruppe kann einem Empfängertyp („Eigner“, „Empfänger“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

<function_cd>, Die genauere Rolle/Funktion des Empfängers („Dateneingang“) wird in diesem Kode-Element beschrieben.

<person>, der Empfänger kann eine Person sein. Mehrere Personen können zu einem Empfängertyp angegeben werden.

<organization>, der Empfänger kann eine Organisation sein. Mehrere Organisationen können zu einem Empfängertyp angegeben werden.

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<intended_recipient>
   <intended_recipient.type_cd V="OWNER" S="1.3.1.3.T.e.s.t" SV="1.0"/>
   <organization>
         <id EX="01" RT="12.3T.e.s.t.t.a.b.e.l.l.e" RTV="1.10"/>
         <organization.nm V="KV Schleswig-Holstein"/>
         <addr>
            <STR V="Teststrasse"/>
            <HNR V="12"/>
            <CTY V="Segeberg"/>
         </addr>
         <telecom V="tel:233212"/>
   </organization>
</intended_recipient>
```

XML-Code 12 /ehd/header/intended_recipient

4.8.1 intended_recipient.type_cd (Empfängertypen)

Der Empfänger bzw. die Zielgruppe kann einem Typ (z.B. Empfängertyp: „Empfänger“, „Eigner“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
<intended_recipient.type_cd V="OWNER" S="1.2.276.0.76.2.2.102" SV="1.0" DN="Eigner"/>

<intended_recipient.type_cd V="RECEIVER" S="1.2.276.0.76.2.2.102" SV="1.0" DN="Empfänger"/>
```

XML-Code 13 /ehd/header/intended_recipient/intended_recipient.type_cd

4.8.2 function_cd (Rolle/Funktion der Empfänger)

Die genauere Rolle/Funktion des Empfängers („Dateneingang“) wird in diesem Kode-Element beschrieben.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
<function_cd V="DtE" S="1.2.276.0.76.2.2.105" SV="1.0" DN="Dateneingang"/>
```

XML-Code 14 /ehd/header/intended_recipient/function_cd

4.8.3 person (Persondaten)

Generell können Persondaten in diesem Element untergebracht werden. Dieses Element wird an mehreren Stellen, wo Personangaben übermittelt werden sollen, wiederverwendet.

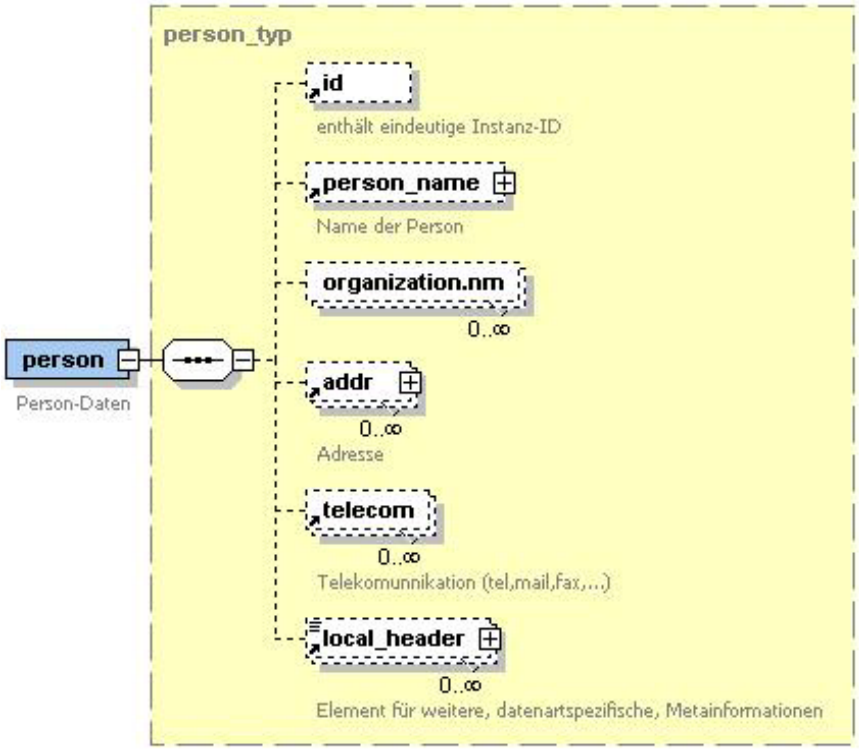


Abbildung 6 /ehd/header/.../person

Das Element hat folgende Attribute:

Kardinalität	0..n
children	**id (0..1)** **person_name (0..1)** **organization.nm (0..n)** **addr (0..n)** **telecom (0..n)** **local_header(0..n)**
attributs	Name Type Use Default Fixed Annotation

<id>, Die Identifikation einer Person.

<person_name>, Angabe zu dem Namen einer Person wird hier hinterlegt.

<organization.nm>, Angabe zu dem Organisationsnamen, zu der die Person zugehört, wird hier hinterlegt. Damit ist der direkter Bezug der Person zu der Organisation vorhanden und die Notwendigkeit die Organisation noch mal im ***<organization>*** - Element anzugeben entfällt.

<addr>, Angabe zu der Adresse, in der die Person erreichbar ist, wird hier hinterlegt.

<telecom>, Angabe zu den Telekommunikationskontakten (mail, fax, tel. usw...), unter den die Person erreichbar ist, wird hier hinterlegt.

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<person>
    <id EX="3223" RT="1.2.276.0.76.T.e.s.t" RTV="1.10"/>
    <person_name>
        <nm>
            <GIV V="Hans"/>
            <FAM V="Müller"/>
            <PFX V="Dr." QAUL="AC"/>
        </nm>
    </person_name>
    <organization.nm V="KVB"/>
    <addr>
        <STR V="Herbert-Lewin-Platz"/>
        <HNR V="2"/>
    </addr>
    <telecom V="tel:2314432"/>
</person>
```

XML-Code 15 /ehd/header/…/person

4.8.3.1 id (Personidentifikation)

Das id-Element wird genauso gebildet wie das Dokument-ID, mit dem Unterschied, dass keine GUID angegeben werden muss. Siehe: id (Dokument-ID). Im EX- und RT-Attribut kann eine beliebige Zeichenfolge verwendet werden, welche die weltweite Eindeutigkeit des Objekts gewährleisten muss.

Wenn als Identifikation ein Kodewert aus einer Schlüsseltabelle verwendet werden soll, so wird im EX-Attribut der Kodewert und im RT-Attribut die OID der Schlüsseltabelle eingetragen. Zusätzlich im RTV-Attribut kann die Version der Schlüsseltabelle angegeben werden.

Es ist empfehlenswert die Version der Schlüsseltabelle aus folgenden Gründen immer anzugeben:

a) Wenn neue Werte in die Schlüsseltabelle hinzugefügt werden, so kann anhand der Tabellen-Version erkannt werden, dass es sich nicht um Fehlerwerte handelt, wenn diese durch das Programm nicht erkannt werden. Die Fehlerursache kann damit geklärt werden.

b) Um nach mehreren Schlüsseltabellenänderungen noch klären zu können, wann ein Wert gültig war. Die Tabellen-Version ist für die Überprüfung von älteren Dateien mit alten Schlüsseltabellenwerten wichtig.

```
<id EX="3223" RT="2.3.5.3.T.e.s.t" RTV="1.10"/>
```

XML-Code 16 /ehd/header/…/person/id

4.8.3.2 person_name (Name der Person)

Die Namensbestandteile der Person werden im Unterelement ***<nm>*** angegeben.

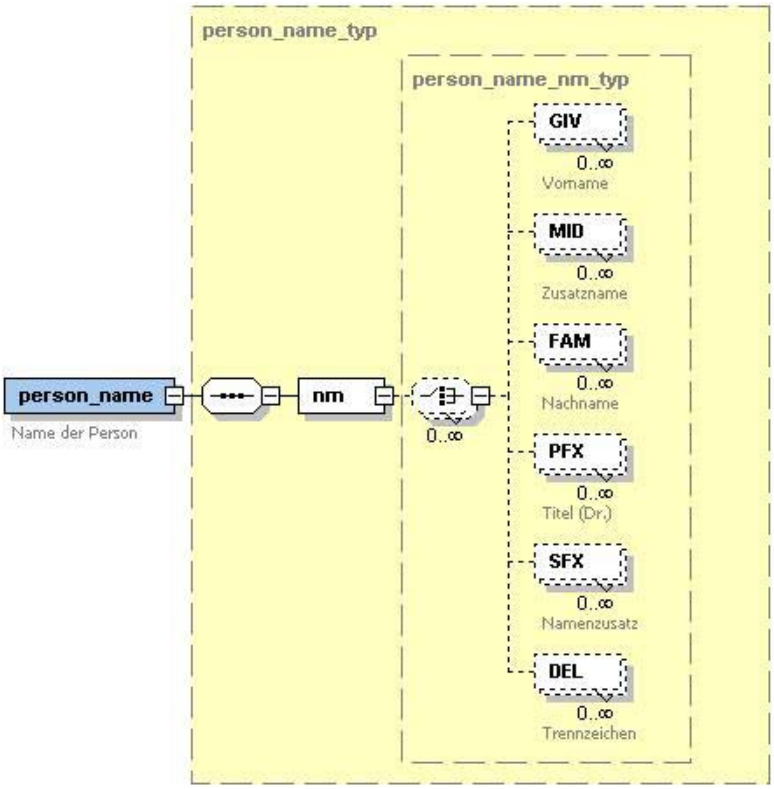


Abbildung 7 /ehd/header/.../person/person_name

Das Element hat folgende Attribute:

Kardinalität	0..1					
children	**nm (1..1)**					
attributs	Name	Type	Use	Default	Fixed	Annotation

<*nm*>, Hier werden die Namensbestandteile angegeben.

Das Element *nm* kann folgende Namensbestandteile enthalten:

<*GIV*>, Vorname

<*MID*>, Zusatzname

<*FAM*>, Nachname

<*PFX*>, Präfix, führender Namenszusatz, wie z.B. „Dr. med“, und Adelsbezeichnungen, wie z.B. „Freiherr“ oder „von“

<*SFX*>, Suffix, Ein folgender Namenszusatz hat eine starke Bindung zum vorhergehenden Teil eines Namens. Folgende Namenszusätze können nicht umgekehrt werden.

<*DEL*>, Ein Trennzeichen hat nur die Bedeutung, genau so gedruckt zu werden, wie es im Namen vorkommt. Ein Trennzeichen hat keine führenden und nachfolgenden Leerzeichen.

Werte für diese Elemente werden im V-Attribut angegeben. Jedes Namensteil-Element hat folgende Attribute:

	Name	Type	Use	Default	Fixed	Annotation
attributs	V	xs:string	required			
	QUAL	xs:NMTOKENS	optional			

V: hier wird der Wert eingetragen

QUAL: Ein kodierter Wert kann angegeben werden, um die Daten näher zu beschreiben. So zum Element ***<PFX>*** kann es zusätzliche Werte „AC“ und „NB“ geben:

Code	Definition	Ausprägung
AC	academic	Akademischer Grad, Zusatz beim Element PFX (mehrere Titel sind durch Leerzeichen getrennt)
NB	nobility	Adelszusatz zum Element PFX, z.B. „Gräfin“ und „von“ (mehrere Namenszusätze sind durch Leerzeichen getrennt)

Tabelle 4 Erläuterungen /person/person_name/pfx

Als Beispiel für den Namen des Arztes „Dr. med. Ernst August Graf von Oberberg“ ist hier folgender Code anzugeben:

```
<nm>
    <GIV V="Ernst August"/>
    <FAM V="Oberberg"/>
    <PFX V="Dr. med." QUAL="AC"/>
    <PFX V="Graf von" QUAL="NB"/>
</nm>
```

XML-Code 17 /ehd/header/…/person/person_name/nm

4.8.3.3 organization.nm (Name der Organisation)

Wenn eine Person zu einer Organisation gehört, kann hier der Name der Organisation angegeben werden. Damit ist der direkter Bezug der Person zu der Organisation vorhanden und die Notwendigkeit, die Organisation noch mal im ***<organization>*** - Element anzugeben, entfällt.

Das Element hat folgende Attribute:

Kardinalität	0..n					
	Name	Type	Use	Default	Fixed	Annotation
attributs	V	xs:string	required			

V: hier wird der Wert eingetragen

```
<organization.nm V="KBV"/>
```

XML-Code 18 /ehd/header/…/person/organization.nm

4.8.3.4 addr (Adresse)

Die Adresse wird in diesem Element erfasst.

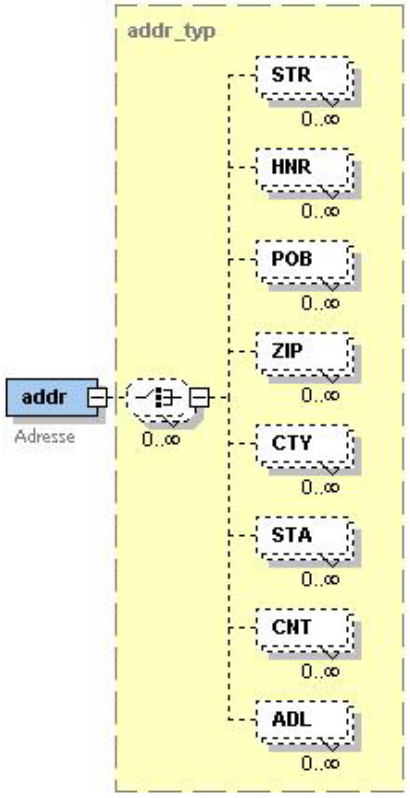


Abbildung 8 /ehd/header/.../person/addr

Dieses Element kann folgende Kindelemente enthalten:

<STR>,	Strasse
<HNR>,	Hausnummer
<POB>,	Postfach
<ZIP>,	Postleitzahl
<CTY>,	Stadt
<STA>,	Bundesland
<CNT>,	Staat
<ADL>,	zusätzliche Adressangabe (additional address locator)

Werte für diese Elemente werden im V-Attribut angegeben. Jedes Adressteil-Element hat folgende Attribute:

Das Element hat folgende Attribute:

Kardinalität	0..n					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:string	required			

V: hier wird der Wert eingetragen

```
<addr>
    <STR V="Herbert-Lewin-Platz"/>
    <HNR V="2"/>
    <ZIP V="10682"/>
    <CTY V="Berlin"/>
</addr>
```

XML-Code 19 /ehd/header/…/person/addr

4.8.3.5 telecom (Kommunikationsmöglichkeiten)

Dieses Element dient dazu, Telefon- und Faxnummern, Emailadressen und Homepages aufzunehmen. Werte werden im V-Attribut angegeben, der mit dem Wert „tel“, „fax“ „mailto“, „http“ oder „ftp“ beginnen muss.

Das Element hat folgende Attribute:

Kardinalität	0..n					
attributs	Name V	Type xs:string	Use required	Default	Fixed	Annotation

V: hier wird der Wert eingetragen

Die Erläuterung für die verschiedenen Werte sind in der Tabelle 5 aufgeführt.

Code	Definition	Beispiel
tel	Telefon/Mobiltelefon	<telecom V="tel:(0221)4449-0" USE="WP"/>
fax	Faxnummer	<telecom V="fax:(0221)4449-400" USE="WP"/>
mailto	Emailadresse	<telecom V="mailto:info@kbv.de" USE="WP"/>
http	Homepage	<telecom V="http://www.kbv.de" USE="WP"/>
ftp	FTP-Server	<telecom V="ftp://ftp.kbv.de" USE="WP"/>

Tabelle 5 Erläuterungen /person/person_name/telecom

4.8.4 organization (Organisationsdaten)

Generell können Organisationsdaten in diesem Element untergebracht werden. Dieses Element wird an mehreren Stellen, wo Organisationsangaben übermittelt werden sollen, wiederverwendet.

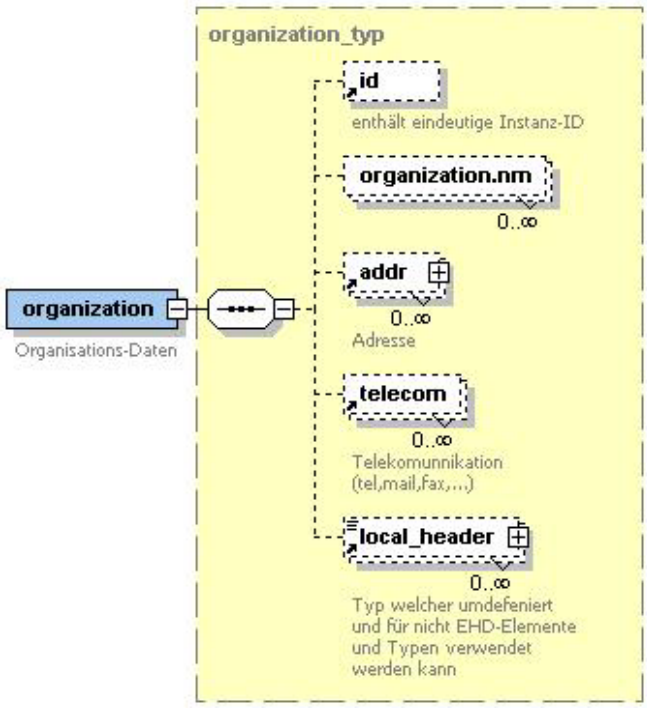


Abbildung 9 /ehd/header/intended_recipient/organization

Das Element hat folgende Attribute:

Kardinalität	0..n
children	**id (0..1)** **organization.nm (0..n)** **addr (0..n)** **telecom (0..n)** **local_header(0..n)**
attributs	Name Type Use Default Fixed Annotation

<id>, Die Identifikation einer Organisation.

<organization.nm>, Angabe zu dem Organisationsnamen

<addr>, Angabe zu der Adresse, in der die Organisation den Sitz hat.

<telecom>, Angabe zu den Telekommunikationskontakten (mail, fax, tel. usw...), unter den die Organisation erreichbar ist.

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<organization>
    <id EX="01" RT="12.3T.e.s.t.t.a.b.e.l.l.e" RTV="1.10"/>
    <organization.nm V="KV Schleswig-Holstein"/>
    <addr>
        <STR V="Teststrasse"/>
        <HNR V="12"/>
        <CTY V="Segeberg"/>
    </addr>
```

```
        <telecom V="tel:233212"/>
    </organization>
```

XML-Code 20 /ehd/header/intended_recipient/organization

4.8.4.1 id (Organisationsidentifikation)

Das id-Element wird genauso gebildet wie das Dokument-ID, mit dem Unterschied, dass keine GUID angegeben werden muss. Siehe: id (Dokument-ID). Im EX- und RT-Attribut kann eine beliebige Zeichenfolge verwendet werden.

Wenn als Identifikation ein Kodewert aus einer Schlüsseltabelle verwendet werden soll, so wird im EX-Attribut der Kodewert und im RT-Attribut die OID der Schlüsseltabelle eingetragen. Zusätzlich im RTV-Attribut kann die Version der Schlüsseltabelle angegeben werden. Gründe für die Angabe der Schlüsseltabellenversion siehe Seite 26.

```
<id EX="3223" RT="2.3.5.3.T.e.s.t" RTV="1.01"/>
```

XML-Code 21 /ehd/header/intended_recipient/organizatio/id

4.8.4.2 organization.nm (Name der Organisation)

Siehe organization.nm (Name der Organisation)

4.8.4.3 addr (Adresse)

Siehe addr (Adresse)

4.8.4.4 telecom (Kommunikationsmöglichkeiten)

Siehe telecom (Kommunikationsmöglichkeiten)

4.9 originator (Urheber)

Das Element ***<originator>*** enthält Informationen zum Urheber/Ersteller der Daten. Urheber können Personen, oder Organisationen seien, die das Dokument/Daten erstellt haben, oder im hohen Maße am Entstehungsprozess beteiligt waren.

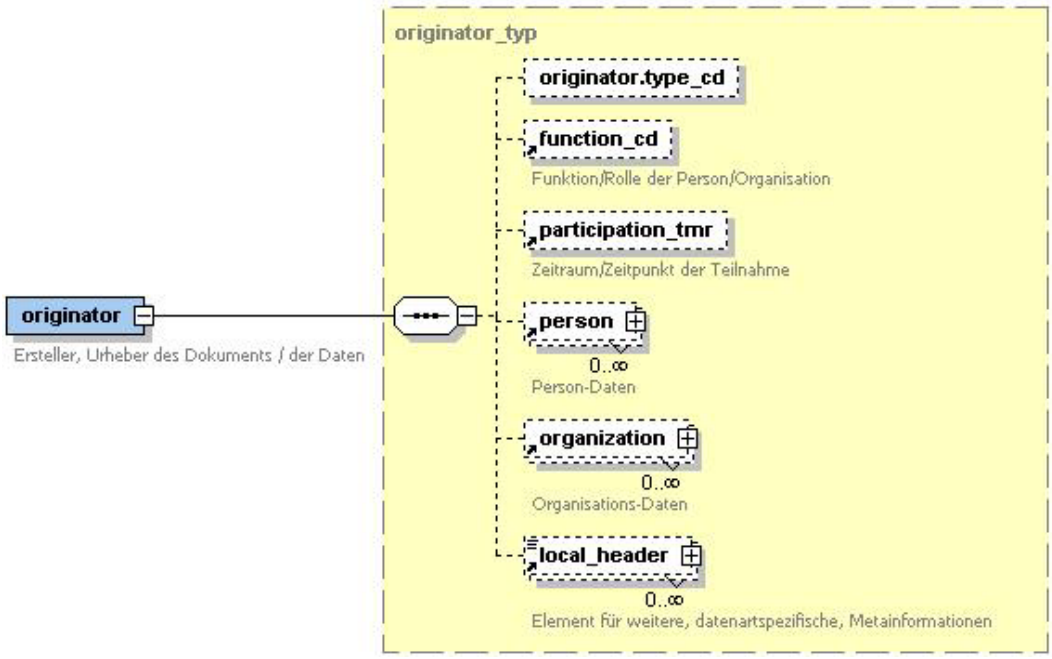


Abbildung 10 /ehd/header/originator

Das Element hat folgende Attribute:

Kardinalität	0..n
children	**originator.type_cd (0..1)** **function_cd (0..1)** **participation_tmr (0..1)** **person (0..n)** **organization (0..n)** **local_header(0..n)**
attributs	Name Type Use Default Fixed Annotation

<originator.type_cd>, Der Urheber kann einem Typ („Labore“, „Datenannahmestellen“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

<function_cd>, Die genauere Rolle/Funktion des Urhebers („Datenausgang“) wird in diesem Kode-Element beschrieben.

<participation_tmr>, Der Zeitraum/Zeitpunkt, wann der Urheber an der Entstehung des Dokuments/Daten beteiligt war.

<person>, der Urheber kann eine Person sein. Mehrere Personen können zu einem Urhebertyp angegeben werden.

<organization>, der Urheber kann eine Organisation sein. Mehrere Organisationen können zu einem Urhebertyp angegeben werden.

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<originator>
```

```
  <originator.type_cd V="KV" S="1.3.1.3.T.e.s.t" SV="1.0"/>
  <participation_tmr V="2003-09-30..2003-10-30"/>
  <organization>
        <id EX="01" RT="12.3T.e.s.t.t.a.b.e.l.l.e/>
        <organization.nm V="KV Schleswig-Holstein"/>
        <addr>
            <STR V="Teststrasse"/>
            <HNR V="12"/>
            <CTY V="Segeberg"/>
        </addr>
        <telecom V="tel:233212"/>
  </organization>
</originator>
```

XML-Code 22 /ehd/header/originator

4.9.1 originator.type_cd (Urhebertypen)

Der Urheber kann einem Typ („Softwarehäuser“, „Datenannahmestellen“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
  <originator.type_cd V="KV" S="1.2.276.0.76.2.2.103" SV="1.0"/>
```

XML-Code 23 /ehd/header/originator/originator.type_cd

4.9.2 function_cd (Rolle/Funktion der Urheber)

Die genauere Rolle/Funktion des Empfängers („Dateneingang“) wird in diesem Kode-Element beschrieben.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
  <function_cd V="ENTW" S="1.2.276.0.76.2.2.105" SV="1.0" DN=”Entwickler”/>
```

XML-Code 24 /ehd/header/originator/function_cd

4.9.3 participation_tmr (Zeitraum/Zeitpunkt der Teilnahme)

Der Zeitraum/Zeitpunkt, wann der Urheber an der Entstehung des Dokuments/Daten beteiligt war. Der Zeitraum wird so angegeben, wie der Gültigzeitraum des Elements <service_tmr>. Siehe service_tmr (Gültigkeitszeitraum).

```
<participation_tmr V="2003-09-30..2003-10-30"/>
```

XML-Code 25 /ehd/header/originator/participation_tmr

4.9.4 person (Persondaten)

Siehe person (Persondaten)

4.9.5 organization (Organisationsdaten)

Siehe organization (Organisationsdaten)

4.10 provider (Lieferant/Sender)

Eine Person oder Organisation, die das Dokument/Daten liefert/sendet oder weiterleitet. Damit ist **<*provider*>** der Sender der Daten. Das Element **<*provider*>** ist ein Pflichtelement, damit die Herkunft der Daten ermittelt werden kann. Generell wird davon ausgegangen, dass der provider auch der Urheber ist, wenn zum Sender das Element **<*originator*>** (Urheber) fehlt. Erst wenn der Urheber vom Absender unterscheidet, wird der Urheber mitangegeben.

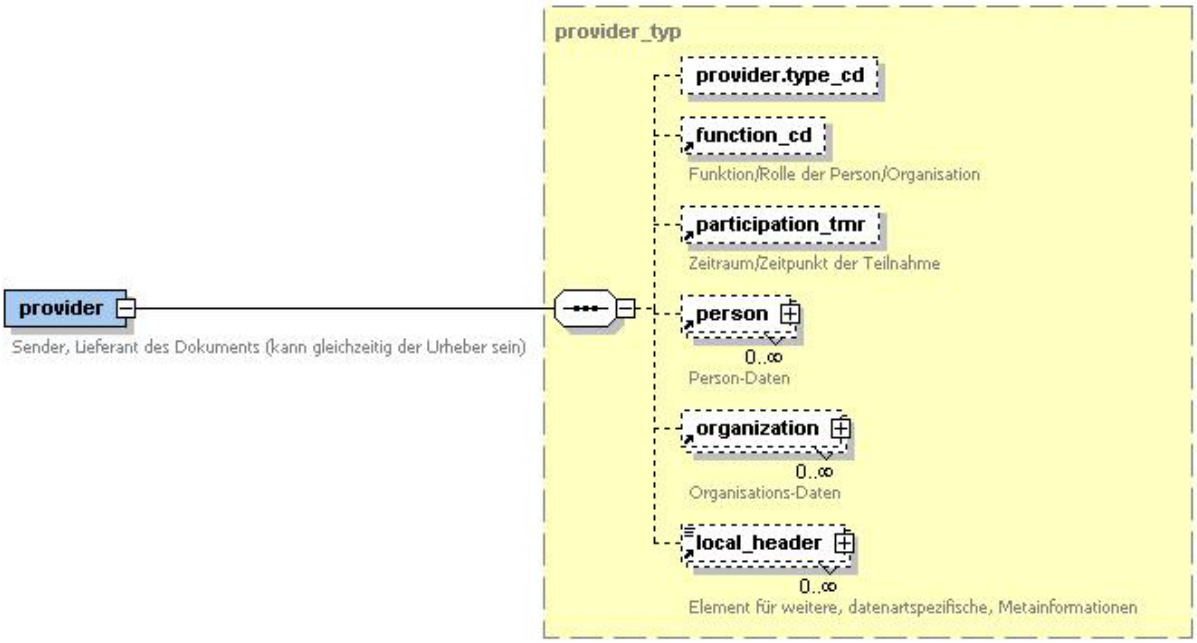


Abbildung 11 /ehd/header/provider

Das Element hat folgende Attribute:

Kardinalität	1..n					
children	**provider.type_cd (0..1)** **function_cd (0..1)** **participation_tmr (0..1)** **person (0..n)** **organization (0..n)** **local_header(0..n)**					
attributs	Name	Type	Use	Default	Fixed	Annotation

<provider.type_cd>, Der Sender kann einem Typ („Labore“, „Datenannahmestellen“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

<function_cd>, Die genauere Rolle/Funktion des Senders („Datenausgang“) wird in diesem Kode-Element beschrieben.

<participation_tmr>, Der Zeitraum/Zeitpunkt, wann der Sender an der Entstehung/Sendung des Dokuments/Daten beteiligt war.

<person>, der Sender kann eine Person sein. Mehrere Personen können zu einem Liefertyp angegeben werden.

<organization>, der Sender kann eine Organisation sein. Mehrere Organisationen können zu einem Liefertyp angegeben werden.

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<provider>
  <provider.type_cd V="KV" S="1.2.276.0.76.2.2.104" SV="1.0"/>
  <participation_tmr V="2003-09-30..2003-10-30"/>
  <organization>
      <id EX="01" RT="12.3T.e.s.t.t.a.b.e.l.l.e" RTV="1.01"/>
      <organization.nm V="KV Schleswig-Holstein"/>
      <addr>
          <STR V="Teststrasse"/>
          <HNR V="12"/>
          <CTY V="Segeberg"/>
      </addr>
      <telecom V="tel:233212"/>
  </organization>
</provider>
```

XML-Code 26 /ehd/header/provider

4.10.1 provider.type_cd (Sendertypen)

Der Sender kann einem Typ („Softwarehäuser“, „Datenannahmestellen“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
<provider.type_cd V="KV" S="1.2.276.0.76.2.2.104" SV="1.0"/>
```

XML-Code 27 /ehd/headerprovider/provider.type_cd

4.10.2 function_cd (Rolle/Funktion des Senders)

Die genauere Rolle/Funktion des Senders („Datenausgang“) wird in diesem Kode-Element beschrieben.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
<function_cd V="DTA" S="1.2.276.0.76.2.2.105" SV="1.0" DN="Datenausgang"/>
```

XML-Code 28 /ehd/header/provider/function_cd

4.10.3 participation_tmr (Zeitraum/Zeitpunkt der Teilnahme)

Der Zeitraum/Zeitpunkt, wann der Sender an der Entstehung/Sendung des Dokuments/Daten beteiligt war. Der Zeitraum wird so angegeben, wie der Gültigzeitraum des Elements <service_tmr>. Siehe service_tmr (Gültigkeitszeitraum).

```
<participation_tmr V="2003-09-30..2003-10-30"/>
```

XML-Code 29 /ehd/header/provider/participation_tmr

4.10.4 person (Persondaten)

Siehe person (Persondaten)

4.10.5 organization (Organisationsdaten)

Siehe organization (Organisationsdaten)

4.11 scope

Das optionale Element ***<scope>*** beschreibt den Geltungsbereich der Daten. ***<scope>*** ist in erster Linie bei Stammdateien relevant, die nur einen begrenzten Geltungsbereich haben, wenn z.B. von der KBV Daten verteilt werden, die nur in bestimmten KV-Regionen gelten. Die näheren Angaben stehen in den Kindelementen.

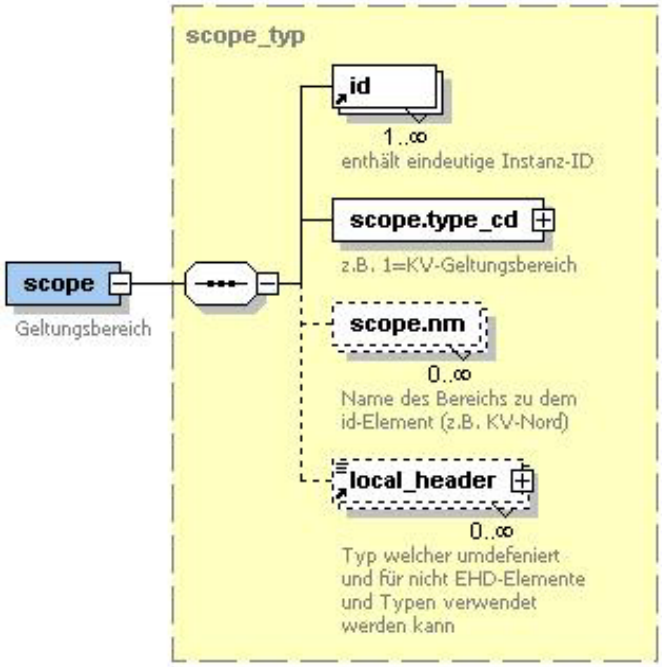


Abbildung 12 /ehd/header/scope

Das Element hat folgende Attribute:

Kardinalität	0..n
children	**id (1..n)** **scope.type_cd (1..1)** **scope.nm (0..n)** **local_header (0..n)**
attributs	Name Type Use Default Fixed Annotation

<id>, Die Identifikation des Geltungsbereiches.

<scope.type_cd>, Das Geltungsbereich kann einem Typ („KV-Geltungsbereich“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

<scope.nm>, Klartextname des Geltungsbereiches

<local_header> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<scope>
    <id EX="74" RT="1.2.276.0.76.2.2.106" RTV="1.0" />
    <scope.type_cd V="KVG" S="1.2.276.0.76.2.2.107" DN="KV-Geltungsbereich"/>
    <scope.nm V="KBV (bundesweit)"/>
</scope>
```

XML-Code 30 /ehd/header/scope

4.11.1 id (Geltungsbereichidentifikation)

Das id-Element wird genauso gebildet wie das Dokument-ID, mit dem Unterschied, dass keine GUID angegeben werden muss. Siehe: id (Dokument-ID). Im EX- und RT-Attribut kann eine beliebige Zeichenfolge verwendet werden.

Wenn als Identifikation ein Kodewert aus einer Schlüsseltabelle verwendet werden soll, so wird im EX-Attribut der Kodewert und im RT-Attribut die OID der Schlüsseltabelle eingetragen. Zusätzlich im RTV-Attribut kann die Version der Schlüsseltabelle angegeben werden. Gründe für die Angabe der Schlüsseltabellenversion siehe Seite:26.

```
<id EX="74" RT="1.2.276.0.76.2.2.106" RTV="1.0" />
```

XML-Code 31 /ehd/header/scope/id

4.11.2 scope.type_cd (Geltungsbereichtypen)

Das Geltungsbereich kann einem Typ („KBV-Geltungsbereich“) zugeordnet werden. Der Typ wird in diesem Kode-Element festgelegt.

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

```
<scope.type_cd V="KVG " S="1.2.276.0.76.2.2.107" DN="KV-Geltungsbereich"/>
```

XML-Code 32 /ehd/header/scope/scope.type_cd

4.11.3 scope.nm (Name des Geltungsbereiches)

Hier wird der Klartextname des Geltungsbereiches (z.B. „Bezirksstelle-XYZ“) angegeben.

Das Element hat folgende Attribute:

Kardinalität	0..n					
attributs	Name V	Type xs:string	Use required	Default	Fixed	Annotation

V: hier wird der Wert eingetragen

```
<scope.nm V="Bezirksstelle -XYZ"/>
```

XML-Code 33 /ehd/header/scope/scope.nm

4.12 state (Bearbeitungszustand)

Das optionale Element **<*state*>** beschreibt in seinem V-Attribut den Bearbeitungszustand. Es wird damit ermöglicht, die XML-Dateien zu kennzeichnen, die einen Bearbeitungsschritt durchlaufen haben (z.B. Prüfung, Filterung).

Bei diesem Element handelt es sich um ein Kode-Element, mit dem Kodewerte angegeben werden können. Der Datentyp ist **v_s_string_typ**, in dem Attribute und Elemente definiert sind, die für Kode-Elemente üblich sind.

Die Werte für das V-Attribut sind je nach konkreter Schnittstelle frei wählbar. Im S-Attribut steht die OID der jeweiligen Schlüsseltabelle.

```
<state V="E01002" S="1.2.276.0.76.2.2.108" DN="EingangspruefungOK"/>
```

XML-Code 34 /ehd/header/state

4.13 interface (Beschreibung der Schnittstelle)

Die Daten der XML-Datei entsprechen einer Schnittstellenbeschreibung, welche normalerweise aus Strukturinformation und ggf. Kontextregeln besteht. Im Element **<*interface*>** wird auf die Beschreibung der zu den Daten gehörenden Schnittstelle verwiesen.

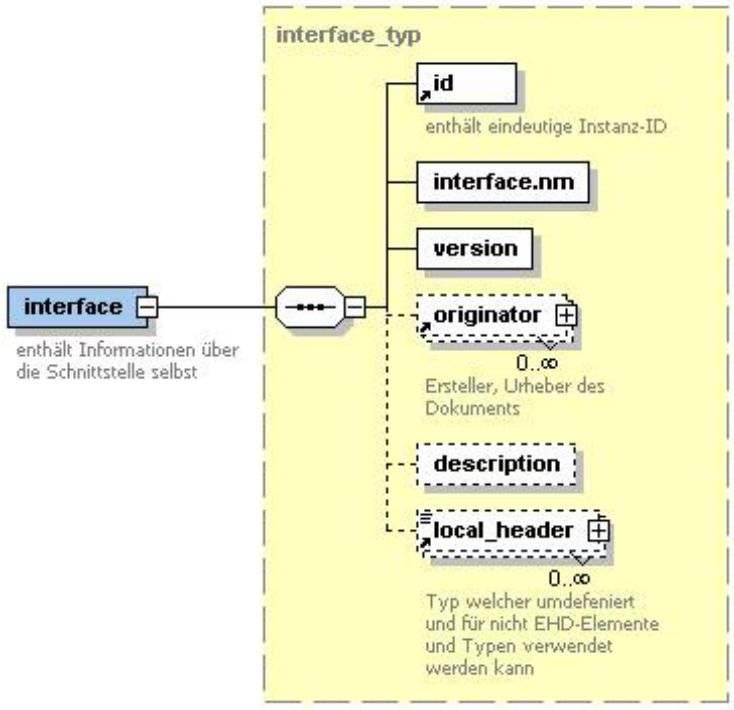


Abbildung 13 /ehd/header/interface

Das Element hat folgende Attribute:

Kardinalität	1..1
children	**id (1..1)** **interface.nm(1..1)** **version (1..1)** **originator (0..n)** **description (0..1)** **local_header (0..n)**
attributs	Name Type Use Default Fixed Annotation

<*id*>, Die Identifikation der Schnittstelle.

<*interface.nm*>, Angabe zu dem Namen der Schnittstelle

<*version*>, Versionsnummer der Schnittstelle

<*originator*>, Angabe zu dem Ersteller/Urheber, alle relevanten Informationen zu der die Schnittstelle definierenden Instanz

<*description* >, Kurzbeschreibung der Schnittstelle oder aber, im günstigsten Fall, eine URL auf ein online zugängliches Dokument der vollständigen Schnittstellenbezeichnung.

<*local_header*> - Element wird in local_header (Platz für lokale Elementdefinitionen) erklärt.

```
<interface>
    <id EX="SDKT" RT="1.2.276.0.76.2.2.109" RTV="1.11"/>
    <interface.nm V="Stammdaten Kostenträger"/>
    <version V="1.00"/>
    <originator>
      <originator.type_cd V="KV" S="1.2.276.0.76.2.2.103" SV="1.0"/>
       <participation_tmr V="2003-09-30..2003-10-30"/>
       <organization>
           <id EX="74" RT="12.3T.e.s.t.t.a.b.e.l.l.e/>
           <organization.nm V="KBV"/>
           <addr>
               <STR V="Herbert-Lewin-Platz"/>
               <HNR V="2"/>
               <CTY V="Berlin"/>
           </addr>
           <telecom V="tel:(030)4005-0"/>
       </organization>
    </originator>
    <description V="http://daris.kbv.de/daris/link.asp?ID=1003734142">
</interface>
```

XML-Code 35 /ehd/header/interface

4.13.1 id (Identifikation der Schnittstelle)

Das id-Element wird genauso gebildet wie das Dokument-ID, mit dem Unterschied, dass keine GUID angegeben werden muss. Siehe: id (Dokument-ID). Im EX- und RT-Attribut kann eine beliebige Zeichenfolge verwendet werden.

Wenn als Identifikation ein Kodewert aus einer Schlüsseltabelle verwendet werden soll, so wird im EX-Attribut der Kodewert und im RT-Attribut die OID der Schlüsseltabelle eingetragen. Zusätzlich im RTV-Attribut kann die Version der Schlüsseltabelle angegeben werden. Gründe für die Angabe der Schlüsseltabellenversion siehe Seite:26.

```
<id EX="SDKT" RT="1.2.276.0.76.2.2.109" RTV="1.11"/>
```

XML-Code 36 /ehd/header/interface/id

4.13.2 interface.nm (Name der Schnittstelle)

Das Element ***<interface.nm>*** enthält die Bezeichnung der Schnittstelle als Text, wie er auf der entsprechenden Schnittstellenbezeichnung steht. Nahliegend ist, dass dieser Text Teile des DN (Displayname) von document_type_cd enthält.

Das Element hat folgende Attribute:

Kardinalität	1..1					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:string	required			

V: hier wird der Wert eingetragen

```
<interface.nm V="Stammdaten Kostenträger "/>
```

XML-Code 37 /ehd/header/interface/interface.nm

4.13.3 version (Versionsnummer der Schnittstelle)

Das Element ***<version>*** enthält die Versionsnummer der Schnittstellenbezeichnung. Da sich von Version zu Version auch Struktur und Inhaltsbedeutung der Daten ändern kann, sind zur Kennzeichnung einer Schnittstelle immer Bezeichnung und Versionsnummer erforderlich.

Das Element hat folgende Attribute:

Kardinalität	1..1					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:nonNegativeInteger	required			

V: hier wird der Wert eingetragen

```
<version V="2.03"/>
```

XML-Code 38 /ehd/header/interface/version

4.13.4 originator (Urheber)

Im Element **<originator>** sind alle relevanten Informationen zu der die Schnittstelle definierenden Instanz angegeben. Die Struktur dieses Elements ist bei originator (Urheber) beschrieben.

4.13.5 description (Kurzbeschreibung)

Das Element **<description>** kann Text mit einer Kurzbeschreibung der Schnittstelle oder aber, im günstigsten Fall, einen URL auf ein online zugängliches Dokument der vollständigen Schnittstellenbeschreibung.

Das Element hat folgende Attribute:

Kardinalität	0..1					
Attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:string	required			
	URL	xs:anyURI	optional			

V: hier wird die Kurzbeschreibung eingegeben

URL: hier wird der Link auf eine physikalische Adresse (Internet-Seite) angegeben.

```
<description V="http://daris.kbv.de/daris/link.asp?ID=1003734142">
```

XML-Code 39 /ehd/header/interface/description

4.14 local_header (Platz für lokale Elementdefinitionen)

In diesem Element können lokale Elemente, abhängig von der konkret zu definierenden Schnittstelle definiert werden. Mit der Zeit und ändernden Anforderungen werden immer neue Elemente benötigt, die nicht in der ehd-Richtlinie berücksichtigt wurden. An dieser Stelle können Erweiterungen am Header vorgenommen werden, und der Schnittstellen-Entwickler kann beliebige neue Elemente hinzufügen. D.h. der Schnittstellenerfinder kann hier Schnittstellenspezifische Metadaten unterbringen.

Das Element hat folgende Attribute:

Kardinalität	0..n					
attributs	Name	Type	Use	Default	Fixed	Annotation
	Ignore	ignore_typ	optional	markup		
	descriptor	xs:string	optional			
	Render	xs:render	optional			

ignore: Dieses Attribut kann einer Anwendung (z.B. stylesheet) Hinweis geben, ob Daten innerhalb des **<local_header>**-Elements ignoriert werden können. Mit Ausprägung „markup“ soll nur das **<local_header>**-Element ignoriert werden. Mit Ausprägung „all“ sollen auch die Elemente und Daten innerhalb des **<local_header>**-Element ignoriert werden.

descriptor: Eine Kurzbeschreibung des Elements bzw. der Daten.

render: In diesem Attribut wird vorgegeben, wie die Inhalte durch Anwendungen (z.B. stylesheets) dargestellt werden sollen.

Die Grundstruktur des **<*local_header*>**-Elements:

Abbildung 14 /ehd/header/local_header

Anstatt des **<*any*>** Elements kann in der konkreten ehd-Schnittstellen-Implementierung, jedes beliebige Element verwendet werden. Dazu wird der Basistyp: local_header-cont.model mit **<*xs:restriction*>** eingeschränkt und die schnittstellenspezifische Elemente hinzugefügt.

Ein Beispiel für die Verwendung des **<*local_header*>**-Elements mit zwei erfundenen Elementen, die Kurzbeschreibung (descriptor) kennzeichnet, dass die Elemete von „KBV“ stammen.

```
<local_header ignore="markup" descriptor="KBV">
    <my_element_A>ein Test</my_element_A>
    <my_element_B V="T2">auch ein Test</my_element_B>
</local_header>
```

XML-Code 40 /ehd/header/local_header

5 Inhaltsdaten (body)

Im Bereich **<*body*>** liegen die eigentlichen Inhalte der Datenlieferung. In diesem Bereich kann der Schnittstellenerfinder seine eigenen Strukturen definieren, wobei die im Abschnitt 8 befindlichen Designregeln beachtet werden müssen. Hier ist auch beschrieben, wie eine ehd-Schnittstelle am besten zu definieren ist, damit sie der ehd-Richtlinie zu 100% entspricht.

Das **<*body*>** Element hat das Kindelement **<*any*>** , welches 0 oder mehrmals vorkommen kann. Die Grundstruktur ist in der folgenden Abbildung beschrieben:

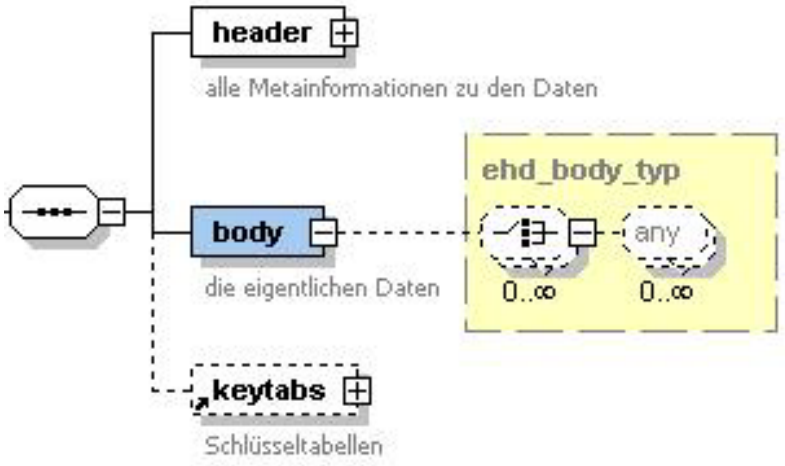


Abbildung 15 /ehd/body

Anstatt des **<*any*>** Elements kann in der konkreten ehd-Schnittstellen-Implementierung, jedes beliebige Element verwendet werden. Dazu wird das ehd_body_typ mit <restriction> eingeschränkt und die Schnittstellenspezifische Elemente hinzugefügt.

5.1 Möglichkeiten body zu verschlüsseln mit XML-Encryption

Diese Möglichkeit wird noch geprüft und steht in der vorliegenden Version der ehd-Richtlinie nicht zur Verfügung.

5.2 Möglichkeiten body zu signieren mit XML-Signatur

Diese Möglichkeit wird noch geprüft und steht in der vorliegenden Version der ehd-Richtlinie nicht zur Verfügung.

6 keytabs (Schlüsseltabellen)

Das (optionale) Element **<*keytabs*>** beinhaltet Schlüsseltabellen, welche für die Referenzierung innerhalb der XML-Datei verwendet werden. Bei der Entwicklung einer konkreten Schnittstelle muss das Element **<*keytabs*>** nicht mit angegeben werden.

Wenn eine Übertragung von internen Schlüsseltabellen innerhalb einer XML-Instanz erlaubt werden soll, weil z.B. im body-Bereich auf diese Code-Wert-Paare referenziert wird, so kann dieses Element verwendet werden. Durch dieses Element können interne Schlüsseltabellen übertragen werden, die nicht öffentlich vorliegen.

Die allgemeine Struktur stellt die folgende Abbildung dar.

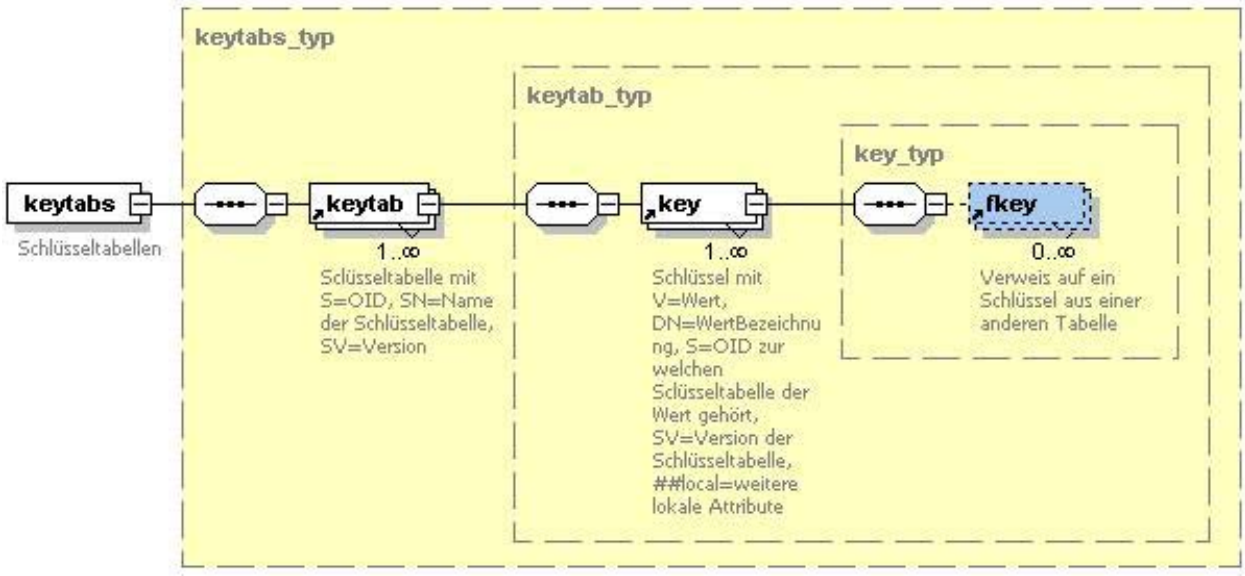


Abbildung 9: /ehd/keytabs

Die dem Element **<*keytabs*>** untergeordneten (Pflicht-) Kindelemente (hier **<*keytab*>** und **<*key*>**) stellen jeweils eine Schlüsseltabelle dar und sind in ihrer Benennung frei wählbar. Die Benennung erfolgt in Attributen.

Das folgende XMLCode-Beispiel erläutert beispielhaft den Aufbau einer **<*keytabs*>** -Sektion.

```
<keytabs>
    <keytab S="13.54.24.5.TEST" SN="Kassenärztliche Vereinigungen" SV="1.0">
        <key V="01" DN="KV Schleswig-Holstein" S="13.54.24.5.TEST" SV="1.0"/>
        <!--..weitere Schlüssel ...-->
        <key V="98" DN="Sachsen" S="13.54.24.5.TEST" SV="1.0"/>
    </keytab>
    <!--..weitere Tabellen ...-->
</keytabs>
```

XML-Code 41 /ehd/keytabs

6.1 keytab (Schlüsseltabelle)

Die Elemente **<*keytab*>** beinhalten jeweils eine einzelne Schlüsseltabelle, welche in Attributen: S, SN, und SV näher beschrieben wird.

Das Element hat folgende Attribute:

Kardinalität	1..n					
children	**key (1..n)**					
attributs	Name	Type	Use	Default	Fixed	Annotation
	S	xs:string	required			
	SN	xs:string	required			

	SV	xs:string	required

S: OID der Schlüsseltabelle, in der kodierte Werte verwaltet werden

SN: menschenlesbarer Klartextname der Schlüsseltabelle

SV: Version der Schlüsseltabelle; Wenn die Schlüsseltabelle geändert bzw. ergänzt wird, wird die Version hochgezählt.

```
<keytab S="13.54.24.5.TEST" SN="Kassenärztliche Vereinigungen" SV="1.0">
        <key V="01" DN="KV Schleswig-Holstein" S="13.54.24.5.TEST" SV="1.0"/>
        <!--..weitere Schlüssel ...-->
        <key V="98" DN="KV Sachsen" S="13.54.24.5.TEST" SV="1.0"/>
</keytab>
```

XML-Code 42 /ehd/keytabs/keytab

6.1.1 key (Schlüssel)

Eine Schlüsseltabelle enthält mehrere Schlüsseln (**<*key*>**). Hier werden die Kodewerte in Attributen eingetragen. Zur besseren Referenzierung von Kodewerten innerhalb der **<*body*>**-Sektion und der Bildung von „Constraints“ (Referenzintegrität) wird die Benennung der Schlüsseltabelle noch einmal in Attributen angegeben.

Das Element hat folgende Attribute:

Kardinalität	1..n					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:string	required			
	DN	xs:string	required			
	S	xs:string	required			
	SV	xs:string	required			
	##local					

V: Kürzel, kodierter Wert

DN: menschenlesbarer Klartextname des Wertes

S: OID der Schlüsseltabelle

SV: Version der Schlüsseltabelle

##local: weitere Attribute, das können weitere Spalten der Tabelle sein.

```
<key V="01" DN="KV Schleswig-Holstein" S="13.54.24.5.TEST" SV="1.0"/>
```

XML-Code 43 /ehd/keytabs/keytab/key

6.1.1.1 fkey (Fremdschlüssel)

Eine Schlüsseltabelle kann einen Verweis auf einen Schlüssel aus einer anderen Tabelle haben. Dieser Verweis (Fremdschlüssel) wird mit Hilfe dieses Elements abgebildet.

Das Element hat folgende Attribute:

Kardinalität	1..n					
attributs	Name	Type	Use	Default	Fixed	Annotation
	V	xs:string	required			
	S	xs:string	required			
	SV	xs:string	required			
	##local					

V: Kürzel, kodierter Wert

S: OID der Schlüsseltabelle

SV: Version der Schlüsseltabelle

##local: weitere Attribute, das können weitere Spalten der Tabelle sein.

```
<fkey V="01" S="13.54.24.5.TEST" SV="1.0"/>
```

XML-Code 44 /ehd/keytabs/keytab/key

6.2 constraint (Referenzintegrität)

Mit Hilfe der Referenzintegrität kann sichergestellt werden, dass z.B. innerhalb der **<*body*>**-Sektion in betroffenen Elementen bzw. Attributen nur Werte aus der passenden Schlüsseltabelle aus der **<*keytabs*>**-Sektion verwendet werden.

Dazu müssen die Schlüsselwerte aus der Schlüsseltabelle als *keys* im Sinne der XML-Schema-Definition definiert werden, was an folgendem Beispiel illustriert wird:

> In der **<*body*>**-Sektion wird das Element <kv> definiert, welches Kodewerte aus der Schlüsseltabelle „kv_bereiche“ enthält. Die Schlüsseltabelle wird im **<*keytab*>**-Element angelegt und die Schlüsseln werden im **<*key*>**-Element angegeben. Das <kv>-Element enthält Attribute (V, S, DN, SN, SV), die für Kode-Elemente üblich sind. So könnte das Schema aussehen:

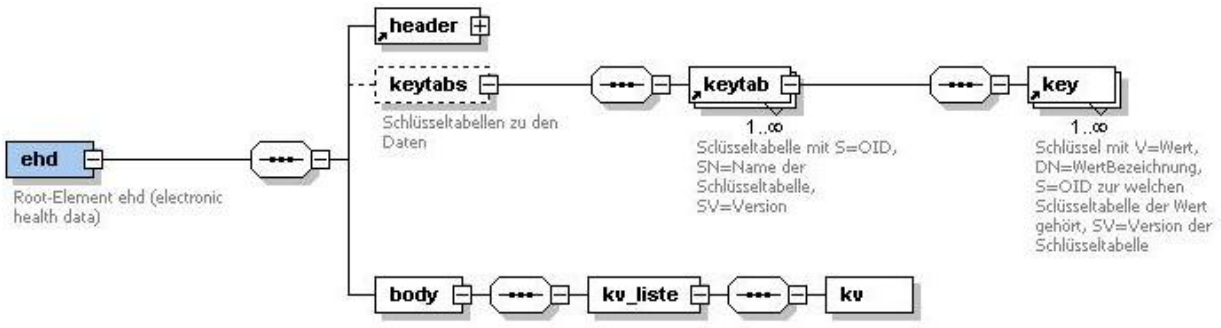


Abbildung 10: Beispiel-XML-Schema zur Referenzintegrität

Um die Werte im V-Attribut der **<*key*>**-Elemente als *key* im Sinne von XML-Schema zu definieren, wird im Schema, dem Root-Element **<*ehd*>** folgender Code hinzugefügt:

```
<xs:element name="ehd">
    . . .
        <!-- constraints -->
        <xs:key name="k_key_all">
            <xs:selector xpath="ehd:keytabs/ehd:keytab/ehd:key"/>
            <xs:field xpath="@V"/>
            <xs:field xpath="@S"/>
            <xs:field xpath="@SV"/>
        </xs:key>
    . . .
```

XML-Code 18: Definition von keys

Um die Werte im V-Attribut des **<*kv*>**-Elements als Verweise auf die entsprechenden *keys* zu definieren, wird auch im Schema, im Root-Element **<*ehd*>** folgender Code hinzugefügt:

```
    . . .
        <xs:keyref name="kref_kv_bereiche" refer="k_key_all">
            <xs:selector xpath="ehd:body/ehd:kv_liste/ehd:kv"/>
            <xs:field xpath="@V"/>
            <xs:field xpath="@S"/>
            <xs:field xpath="@SV"/>
        </xs:keyref>
    . . .
</xs:element>
```

XML-Code 19: Definition von keyrefs

Durch diese beiden Definitionen ist sichergestellt, dass in der konkreten XML-Datei in kv_liste/kv (@V,@S,@SV) nur Werte aus keytab/key (@V,@S,@SV) vorkommen. Der Schlüssel wird aus drei Attributen gebildet, damit eine genaue Identifikation möglich ist.

XPath vom Schema ist die **eingeschränkte** Version von XSL-XPath. Deshalb müssen bei der Definition von *key* und *keyref* folgende Regeln beachtet werden:

1. Den vollständiger Pfad immer mit zugehörigem **Namensraum** (z.B. xmlns:ehd) angeben. Ohne Namensraum werden Elemente nicht gefunden. Konstrukte wie “.//“ und “*“ können verwendet werden.
2. *key* und *keyref* müssen **innerhalb eines** Elements definiert werden, sonst ist die Referenz außerhalb des Referenzierungsbereichs und der Parser meldet Fehlermeldung („... out of scope“).
3. Auf Vaterelemente kann nicht mit „..“ referenziert werden.
4. Zusätzliche Einschränkungen mit [..] dürfen nicht verwendet werden.

Nähere Informationen zur Verwendung von *key* und *keyref* und Schema-XPath gibt es auf der W3C-Seite: [1]

7 Namensgebung für ehd-Dateien

Für konkrete Instanzen einer ehd-Schnittstelle ist eine verbindliche Namensgebung vorgegeben. Der Name spiegelt dabei einen Teil der im Header befindlichen Metainformationen über die Daten wieder. Diese Redundanz ist u.a. notwendig, wenn verschlüsselte Daten geliefert werden und diese automatisiert weiterverarbeitet werden sollen.

Die hier vorgeschriebene Namensgebung betrifft XML-Dateien von ehd-Schnittstellen. Es ist nicht sichergestellt, dass daraus abgeleitete Dateinamen für Nicht-XML-Dateien mit der gleichen Namensgebung funktionieren. In einem solchen Fall muss das „+“-Zeichen aus dem Dateinamen entfernt, oder durch ein anderes ersetzt werden.

Der Dateiname besteht aus Elementen, genannt „Nameparts“. Es gibt drei obligate und mehrere optionale Nameparts.

Aufbau des Namens einer ehd-Datei:

[ehd.]datatyp_vv.vv_sender[_x1+val][_x2+val][.._xn+val].xml[.zip][.kry]

[ ] ..... bedeutet allgemein, dass das Namenselement (der "Namepart") optional ist
_ ..... Trennungszeichen zwischen den Namenselementen
_xx+val .... optionales Namenselement mit Wert, [_Namepart-ID+value]. Als Trennzeichen wird zwischen Namepart und dem Wert das „+“-Zeichen verwendet.

Obligate Nameparts:
Die obligaten Nameparts haben keine Namepart-ID sondern sie sind an der Position erkennbar – Die Reihenfolge der ersten drei Nameparts ist also festgelegt.

datatyp ..... Datentyp, "Satzart", "ehd." ist optional als Vorsatz erlaubt;
Entspricht dem Header-Element <*document_type_cd*>.

vv.vv ..... VersionsNr. der Datentypbeschreibung;
Entspricht dem Element <*version*> des Header-Elements <*interface*>.

sender ..... Absender der Lieferung, (nicht immer mit Erzeuger bzw. Erstlieferanten der Daten identisch) bzw. wer hat die Daten geliefert;
Entspricht dem Element <*person*> oder dem Element <*organization*> des Header-Elements <*provider*>.

Optionale Nameparts:
Die optionalen Nameparts werden – im Gegensatz zu den obligaten Nameparts - durch ihre Namepart-ID gekennzeichnet, sie können in beliebiger Reihenfolge dem festen Namensteil bzw. den obligaten Nameparts folgen. Erlaubte optionale Nameparts sind:

re+ ..... receiver - Empfänger der Lieferung (nicht immer mit Nutzer bzw. Endempfänger der Daten identisch) bzw. an wen werden die Daten geliefert;

Entspricht dem Element <person> oder dem Element <organization> des Header-Elements <intended_recipient>. Wenn der Typecode den Wert für „Empfänger“ hat.

tf+ ..... timeframe - Zeitraum auf den sich die Daten beziehen, Folgende Notation ist erlaubt:
YYYY oder
YYYYqQ oder
YYYYmMM oder
YYYYwWW oder
YYYYmMMdDD oder
YYYYmMMdDD-YYYYmMMdDD oder
-YYYYmMMdDD oder
YYYYmMMdDD-

Y.. Jahreswert, M.. Monatswert, W.. Wochenwert, D.. Tageswert , Q.. Quartalswert
q.. Quartal, m.. month, w.. week, d.. day

Entspricht inhaltlich dem Header-Element <service_tmr>.

id+ ..... identification - einmalige Kennung der Datei;
Entspricht inhaltlich dem Header-Element <id>.

nr+ ..... number - Nummer der Lieferung, falls zu einem Zeitraum mehrere Lieferungen erfolgen;
Entspricht inhaltlich dem Header-Element <version_nbr>.

co+ ..... consignor – eigentlicher (ursprünglicher) Absender, "Eigentümer" ;
Entspricht inhaltlich dem Header-Element <originator>.

be+ ..... beneficiary - Bezugsberechtigter ("Nutzer", eigentlicher Empfänger);
Entspricht dem Element <person> oder dem Element <organization> des Header-Elements <intended_recipient>, Wenn der Typecode den Wert für den „Nutzer“ hat.

td+ ..... typ of delivery - Art der Lieferung (z.B. Erst-, Korrektur-, Ersatz-Lieferung);
Entspricht inhaltlich dem Element <document_relationship.type_cd> des Header-Elements <document_relationship> .

st+ ..... state - Zustand (falls die Daten gefiltert werden kann man hier den Bearbeitungszustand vermerken);
Entspricht inhaltlich dem Header-Element <state >.

du+ ..... dummy - Platzhalter z.B. für Tests, kann auch mehrmals verwendet werden

Um die Konsistenz zwischen den Informationen im Header und im Dateinamen zu gewährleisten, ist es angebracht, den Namen aus dem Header zu generieren.

Es ist vorgesehen, bei Bedarf in einer späteren Version der ehd-Richtlinie neue optionale Nameparts hinzuzufügen.

Die Extension .zip ist zulässig, wenn die ehd-Datei mit einem zip-Algorithmus komprimiert wurde. Wenn es Einschränkungen bei der Art und Weise des „zippens“ gibt, muss das in der konkreten Schnittstellenbeschreibung der ehd-Schnittstelle erläutert werden.

Die Extension .kry ist zulässig, wenn die ehd-Datei mit einem Verschlüsselungs-Algorithmus verschlüsselt wurde. Wenn es Einschränkungen bei der Art und Weise des verschlüsseln

gibt, muss das in der konkreten Schnittstellenbeschreibung der ehd-Schnittstelle erläutert werden.

Fiktive Beispiele von ehd-Dateinamen:

12345_01.05_kv04.xml
12345_01.05_kv04_ tf+2004q2.xml
ehd.sdkt_01.12_kbv_ re+allkv_tf+2004q4.xml.zip.kry
33445_03.33_kv05_ re+kbv_ tf+2004m05_st+eingangspruefungOK.xml
54545_02.01_kv06_ re+ks12345678_ tf+2003m04d01-2003m05d15.xml.zip
10203_01.09_kbv_ re+kv07_ tf+2004_co+kv08_td+add.xml.kry
ehd.sdebm2000plus_01.02_kbv_ re+allkv_tf+2005q1.xml.zip

Die vorgegebene Namensgebung betrifft ehd-Dateien, gezippte Archive können einer anderen Konvention entsprechen (KV-DTA Richtlinie).

8 Designregeln

8.1 Zeichensatz

Als Zeichensatz wird ISO 8859-1 vorgeschrieben.

8.2 Bezeichner für Elemente, Typen, Attribute und Schemadateien

Neben den allgemeinen Regeln für XML-Namen, welche sich aus den Spezifikationen des W3C ergeben, gelten folgende Einschränkungen:

- Für Elemente, Typen und Attribute sollen landeseigene Namen verwendet werden. Die Namen werden vollständig klein geschrieben. Zur Trennung von Wörtern (zwecks besserer Lesbarkeit) sind ausschließlich Unterstriche (_) erlaubt (z.B. sonstige_leistungserbringer).
- Alle Bezeichner dürfen nur aus den Buchstaben a-z und den Ziffern 0-9 sowie dem Unterstrich _ bestehen.

- Bezeichner für Typen, Elemente und Attribute, welche aus externen Schemata (z.B. CDA oder SCIPHOX) entlehnt werden, behalten zur besseren Identifizierung ihre dortige originale Schreibweise (z.B. document_type_cd aus CDA, KostentraegerAbrechnungsbereich aus einer SCIPHOX-SSU). Auf die Entlehnung ist innerhalb der <xs:annotation> -Tags hinzuweisen.
- Bezeichner für Typen haben die Endung „_typ“.
- Bezeichner für Gruppen haben die Endung „_gruppe“.
- Bezeichner für Listen haben die Endung „_liste“ (siehe Kapitel 8.4.3).
- Die Verwendung von Abkürzungen ist zu vermeiden und nur für allgemein bekannte Begriffe (z.B. dmp) erlaubt.
- Aus der Erfahrung mit der Schemataversionierung, sollten alle Schematadateien in der Dateibezeichnung die Version der Schemata enthalten: **schemata-datei[_Vx.xx].xsd** z.B. ehd-Richtlinie V1.40 hat eine Schematadatei **ehd_header_V1.40.xsd**.
- Die Schemata, von unterschiedlichen Versionen, sollten in verschiedenen Ordnern liegen. Die Ordner erhalten in diesem Fall im Ordnernamen die Version der Schemata: **schema[_Vx.xx]** z.B. die Schemata der ehd-Richtlinie V1.40 liegen im Ornder: **schema_V1.40**

8.3 Wiederverwendung von Typen

8.3.1 Allgemeines

Implizite Typdefinitionen sind bei einfachen Typen dann sinnvoll, wenn davon auszugehen ist, dass dieser Typ nicht mehrmals benötigt wird, da sie das XML-Schema kürzer machen.

Durch die Verwendung von expliziten Typdefinitionen (benannte Typen) können diese Typen wiederverwendet werden, indem neue Schemadefinitionen aus vorhandenen Typen zusammengesetzt werden.

Zu diesem Zweck werden Typen, die auch in anderen Projekten im Gesundheitsbereich verwendet werden können, nicht im speziellen Schema selbst definiert, sondern in die ehd-Typbibliothek ausgelagert. Diese wird dann durch eine <xs:import> - Anweisung eingebunden.

Die ehd-Richtlinie ist bewusst aus Datentypen zusammengebaut, damit spezifischen ehd-Schnittstellen-Schemas für spezielle Zwecke abgeleitet und eingeschränkt werden können.

8.3.2 Wiederverwendung von Typen aus CDA und SCIPHOX

Beim Entwurf von XML-Schnittstellen ist die weitestgehende Wiederverwendung von in CDA und SCIPHOX definierten Typen bzw. Elementen anzustreben. Dies wird durch die Einbindung von vorhandenen Basistypen mittels <xs:import>, <xs:include/> oder <xs:redefine/> erreicht. Ableitungen von Basistypen werden mit <xs:restriction> (Einschränkung) oder <xs:extension> (Erweiterung) vorgenommen. Nur mit diesen Mechanismen kann die Qualität der erstellten Schemas sichergestellt werden.

Im Header können Datentypen nur mit Option <xs:restriction> (Einschränkung) wiederverwendet werden. Es ist nicht zulässig den Header zu erweitern, das wird vom Parser als Fehler

gemeldet. Damit soll sichergestellt werden, dass mögliche Header-Implementierung nur eine Untermenge des Headers darstellen.

Im ***<body>*** -Sektion können beliebige Elemente und Datentypen die Basisdatentypen mit Option ***<xs:extension>*** (Erweiterung) nutzen.

8.4 Aufzählungen

8.4.1 Verwendung von Enumerationen

Aufzählungstypen (***<xs:enumeration>***) sind der Verwendung von Stringtypen mit Pattern-Einschränkung vorzuziehen.

Beispiel:

```
<xs:simpleType name="gebuehrenordnung_typ">
    <xs:restriction base="xs:string">
        <xs:enumeration value="01"/>
        <xs:enumeration value="02"/>
        <xs:enumeration value="03"/>
    </xs:restriction>
</xs:simpleType>
```

anstatt

```
<xs:simpleType name="go_mit_string_typ">
    <xs:restriction base="xs:string">
        <xs:pattern value="0[1-3]"/>
    </xs:restriction>
</xs:simpleType>
```

Wenn davon auszugehen ist, dass solche Enumerationen ständigem Wandel unterliegen, so ist besser keine Enumerationen vorzuschreiben, sonder auf Kode-Elemente mit Schlüsseltabellen auszuweichen. Dazu werden Kode-Elemente mit den Verweisen auf Schlüsseltabellen definiert, der Änderungsauswand kann damit verringert werden. Bei Änderungen werden lediglich geänderte Schlüsseltabellen veröffentlicht.

8.4.2 Schlüsseltabellen

Als Schlüsseltabellen sind Listen von Code-Wert-Paaren anzusehen, welche über einen längeren Zeitraum normalerweise unverändert bleiben. Solche Schlüsseltabellen können entweder direkt in die Stammdateien eingebunden werden (siehe Kapitel 8.4.2.2) und dann intern referenziert werden oder als externe Datei referenziert werden (siehe Kapitel 8.4.2.1).

Welche Art der Referenzierung gewählt wird, sollte unter Beachtung des Umfanges und der Änderungshäufigkeit der Schlüsseltabelle entschieden werden.

8.4.2.1 Verweise auf bestehende externe Schlüsseltabellen

Wenn Verweise auf bestehenden KBV-Schlüsseltabellen in XML-Dateien verwendet werden sollen, ist der jeweilige Code in das V- bzw. value-Attribut zu übernehmen. Im S-Attribut wird mit Hilfe einer OID/URN auf die entsprechende Schlüsseltabelle verwiesen. Zusätzliche An-

gabe der Schlüsseltabellenversion (SV-Attribut) erleichtert die Historisierung und das wieder finden von gültigen Werten.

Beispiele:

```
<ktgruppe V="36" S="urn:ehd/keytab/kbv/kts/ktgruppe" SV="1.10"/>
<kv V="01" S="12.3.1..4.2.4.4553" SV="1.0"/>
```

Die Schlüsseltabellen der KBV, ihre URIs und die zulässigen Code-Wert-Paare werden in Zukunft im Internet dokumentiert und öffentlich zugänglich. Sie sollten daher in der Schnittstellenbeschreibung nicht nochmals aufgeführt werden.

8.4.2.2 Eingebettete Schlüsseltabellen

Sollen Schlüsseltabellen in eine Stammdatei eingebettet werden, so sollte im Schema die Codierung und Referenzierung mittels ***<xs:key>*** und ***<xs:keyref>*** vorgenommen werden, um damit die Integrität der Werte sicherzustellen.

Die Verwendung von ID und IDREF sollte vermieden werden.

8.4.3 Listen und Kollektionen

8.4.3.1 Listendatentypen

Die Verwendung von vordefinierten Listentypen (NMTOKENS, IDREFS etc.) sowie die Erstellung eigener Listentypen mittels ***<xs:list>*** ist nur in begründeten Ausnahmefällen erlaubt, da nicht alle Programmiersprachen die automatische Auflösung in einzelne Token unterstützen und diese daher die Verarbeitung erschweren. Stattdessen ist eine Kollektion (siehe Kapitel 8.4.3.2) zu verwenden.

Nicht zulässig:

im Schema:

```
<xs:element  name="notallowed" type="xs:NMTOKENS"/>
```

in der XML-Instanz:

```
<notallowed>01 02 03 44 50</notallowed>
```

korrekte Kodierung:

besser ist, im Schema das Element mit maxOccurs zu definieren:

```
<xs:element name="element" maxoccurs="unbounded"/>
```

Es besteht zwar nicht die Notwendigkeit zusätzlich ein Klammerungselement (..._liste) zu definieren, wenn jedoch die Schnittstelleanforderungen dies erfordern, kann der Container (..._liste) definiert werden. Die Verwendung sollte gut begründet sein, weil ein Container mehr Platz in der XML-Instanz verbraucht und nicht unbedingt zu mehr Strukturierung und Lesbarkeit führt:

```
<xs:element name="element_liste">
    <xs:complexType>
        <xs:sequence>
```

```
            <xs:element name="element" maxoccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

in der XML-Instanz:

```
<element_liste>
    <element value="01"/>
    <element value="02"/>
    <element value="03"/>
    <element value="44"/>
    <element value="50"/>
</element_liste>
```

8.4.3.2 Kollektionen

Als Kollektionen werden Konstrukte in XML-Dateien bezeichnet, in denen Elemente vom gleichen Typ mehrfach vorkommen.

Beispiel:

```
<stammdatei>
    <header>...</header>
    <body>
        <stammsatz></stammsatz>
        <stammsatz></stammsatz>
        ...
        <stammsatz></stammsatz>
    </body>
</stammdatei>
```

Diese Kollektionen können mit einem umschließenden Element versehen werden, welches die Endung „_liste“ trägt.

Beispiel:

```
<stammdatei>
    <header></header>
    <body>
        <stammsatz_liste>
            <stammsatz></stammsatz>
            <stammsatz></stammsatz>
            ...
            <stammsatz></stammsatz>
        </stammsatz_liste>
    </body>
</stammdatei>
```

8.5 Aufteilung auf mehrere Dokumente

Ein XML-Schema kann in mehrere Dokumente (*.xsd - Dateien) aufgeteilt werden. In diesem Fall müssen die Teildokumente denselben Namespace-URI als Targetnamespace verwenden. Die einzelnen Teildokumente können mittels ***<xs:include>*** zusammengefügt werden. Bei

Aufteilung in mehrere Schemata mit unterschiedlichen Namespace-URI, können diese mittels **<xs:import>** zusammengefügt werden.

8.6 Namespaces

8.6.1 Verwendung

In allen XML-Schemata ist die Verwendung von Namespaces Pflicht.

Jeder Namespace muss innerhalb der ehd-Namespacehierarchie eindeutig sein. Die bereits verwendeten Namespaces werden im Internet veröffentlicht. Bei der Registrierung einer neuen ehd-Schnittstelle muss der Namespace angegeben werden.

8.6.2 Aufbau der Namespacehierarchie

Alle Namespace-URIs beginnen mit **urn:ehd**. Namespace-URIs werden nach dem Schema **urn:ehd/datenart/[subdatenart/]versionsnummer** gebildet (z.B. **urn:ehd/12345/sle/001** für die erste Version der SLE-Schnittstelle).

8.6.3 Präfixe

Die im XML-Schema verwendeten Namespacekurzbezeichner (Präfixe) sind frei definierbar. Bei der Auswahl eines Präfixes für einen Namespace ist eine sinnvolle Bezeichnung zu wählen, die nicht länger als 6 Zeichen sein darf.

Beispiel :

Namespace-URI:	urn:ehd/kts/ktstamm/001
Präfix :	kts
Qualified Name:	<kts:vknr></kts:vknr>

9 Anleitung zum Erstellen einer ehd-Schnittstelle

Die ehd-Richtlinie bildet den Rahmen für die Entwicklung von konkreten XML-Schnittstellen. In verschiedenen Projekten werden unterschiedliche Schnittstellen definiert. Zur Erstellung von konkreten Schnittstellen wird der ehd-Rahmen d.h. das ehd_root.xsd und ehd_header-Schema für diese Zwecke angepasst. Bei sehr strikten Schemas wird der ehd-Header soweit eingeschränkt, bis nur die Elemente und Daten erlaubt sind, die für die spezielle Schnittstelle relevant sind.

Das Ziel dieses Rahmens ist, bei der großen Anzahl von verschiedenen angepassten Schnittstellen die Obermenge vorzugeben, mit der Standardtools und Werkzeuge zum Lesen und Validieren der XML-Dateien entwickelt werden können.

Vorgehensweise bei der Erstellung der xy-ehd-Schnittstellen:

1. xy_root-Schema definieren, welches von der ehd_root.xsd mit **<xs:restriction>** abgeleitet ist.

2. xy_header-Schema definieren, welches von der ehd_header.xsd mit **<xs:restriction>** abgeleitet ist.
3. xy_body-Schema definieren.

- alle Schemas können eigene Typebibliotheken verwenden.

Mit dieser Vorgehensweise kann mit dem XML-Parser (Xerces) sichergestellt werden, dass keine (versehentliche) Abweichungen von der ehd-Richtlinie gibt.

Detaillierte Anmerkungen zum Erstellen der Bereiche Root, Header und Body:

- Im Root-Element ist die Anpassung nur mit **<xs:restriction>** (Einschränkung) vorzunehmen. Das Root-Element kann auf einen beliebigen Datentyp verweisen, der jedoch vom Basisdatentyp abgeleitet ist. Dazu wird neues Schema erstellt und das xy_root Element vom Typ: ehd_root_typ abgeleitet.
- Im Header ist die Anpassung auch nur mit **<xs:restriction>** (Einschränkung) erlaubt. Das Header-Element kann auf einen beliebigen Datentyp verweisen, der jedoch vom Basisdatentyp abgeleitet ist. Damit können alle Header-Elemente auf die spezielle xy-Schnittstelle eingeschränkt werden. Dazu wird neues Schema erstellt und das xy_header Element vom Typ: ehd_header_typ abgeleitet. Schnittstellenspezifische Elemente können nur an vorgesehen Stellen **<local_header>** hinzugefügt werden. Wie im Kapitel 4.14 zu <local_header> beschrieben ist, wird der Basistyp: local_header-cont.model mit **<xs:restriction>** eingeschränkt und die schnittstellenspezifische Elemente können hinzugefügt werden.
- Im Body-Bereich wird der Basistyp: ehd_body_typ ebenfalls mit **<xs:restriction>** eingeschränkt. Analog zu <local_header> (Kapitel 4.14) wird das **<any>** Element mit eigentlichen schnittstellenspezifischen Elemente ersetzt. Für den Body-Bereich kann auch neues Schema erstellt werden.
- Das **<keytabs >**-Element ist unverändert zu übernehmen.

Folgende Abbildung stellt die Beziehungen zwischen den xy- und ehd-Schemas dar:

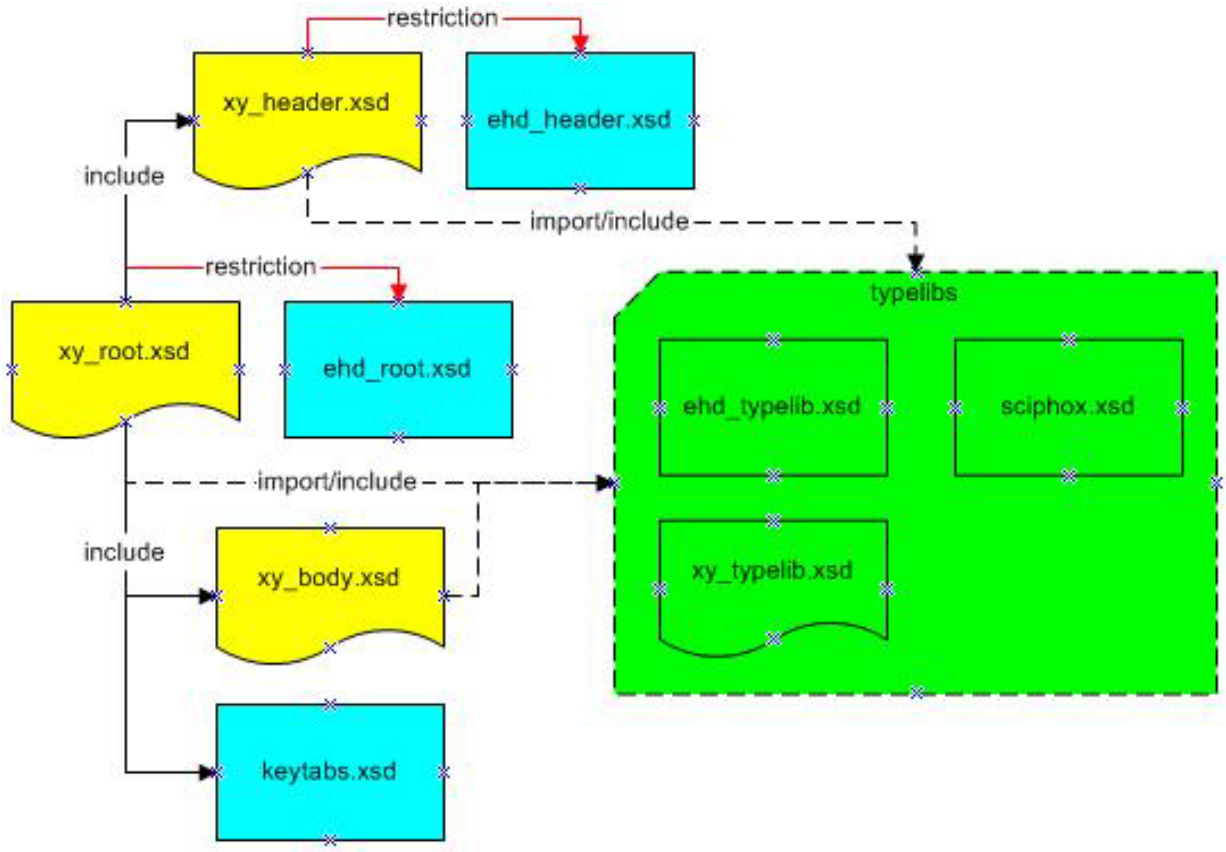


Abbildung 16 Aufteilung der Schemas für die ehd-Schnittstelle XY

In der Abbildung sind Schemas in Grün markiert, die nicht verändert werden können (Type-bibliotheken). Eigendefinierte Schemas sind in Cyan dargestellt. XY-Schemas, die Basis- und ehd-Typen einschränken, sind in Gelb dargestellt.

Das Schema „xy_root.xsd“ schränkt das „ehd_root.xsd“ ein. Gleichzeitig verweist/importiert es das Schema „xy_header.xsd“ und „xy_body.xsd“. Optional kann auch das ehd-Schema „keytabs.xsd“ importiert werden. Das schnittstellenspezifische Schema „xy_header.xsd“ enthält eine Ableitung des „ehd_header_typ“ aus der importierten Bibliothek „ehd_header.xsd“. Weitere Bibliotheken z.B. von „SCIPHOX“ oder eigendefinierte Schemas können zusätzlich importiert werden.

Es werden im Folgendem drei Beispiele für die Ableitungen der Bereiche Root, Header und Body vorgestellt:

Der Code für die ***<xs:restriction>***-Ableitung des Roots sieht im xy_root.xsd wie folgt aus:

```
<xs:schema targetNamespace="urn:ehd/001" ...
        ... blockDefault="substitution">

    <xs:include schemaLocation="xy_header.xsd"/>
    <xs:include schemaLocation="keytabs.xsd"/>

    <xs:element name="ehd" type="xy_root_typ">
    <xs:complexType name="xy_root_typ">
        <xs:complexContent>
```

```
            <xs:restriction base="ehd_root_typ">
                <!-- hier Elemente übernommen -->
                <xs:element name="header" type="xy_header_typ">
                <xs:element name="body" type="xy_body_typ">
                <xs:element ref="keytabs" minOccurs="0"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
```

XML-Code 45 xy_root.xsd

Es wird ein neuer Datentyp („xy_root _typ“) definiert, der eine Einschränkung des Basisdatentyps (hier „ehd_root_typ“) darstellt. Die Einschränkung wird mit ***<xs:restriction>*** erreicht. Jetzt können weitere Typen verwendet werden, die auch einen eingeschränkten Basisdatentyp darstellen. Z.B. xy_header.typ und xy_body_typ.

Das blockDefault-Attribut ist hier bewusst angegeben, weil Substituierung von Kindelementen im Root-Bereich nicht erlaubt ist. Das Weglassen dieser XML-Prüfung könnte den einen oder anderen Schnittstellenentwickler dazu veranlassen, diese Lücke für seine Schnittstelle auszunutzen.

Der Code für die ***<xs:restriction>***-Ableitung des Headers sieht im xy_header.xsd wie folgt aus:

```
<xs:schema targetNamespace="urn:ehd/001" ...
          ... blockDefault="substitution">

    <xs:complexType name="xy_header_typ">
        <xs:complexContent>
            <xs:restriction base="ehd_header_typ">
                <!-- hier Elemente übernommen -->
                <xs:element ref="id"/>
                <xs:element ref="set_id" minOccurs="0"/>
                <xs:element name="document_type_cd" type="xy_document_type_cd_typ"/>
                ...
                <!-- weitere Elemente können übernommen werden -->
                ...
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="xy_document_type_cd_typ">
        <xs:complexContent>
            <xs:restriction base="document_type_cd_typ">
                <xs:attribute name="V" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="SDKT"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
```

XML-Code 46 xy_header.xsd

Hier wird auch neuer Datentyp („xy_header_typ“) definiert, der eine Einschränkung des Basisdatentyps (hier „ehd_header_typ“) darstellt. Die Einschränkung wird mit **<xs:restriction>** erreicht. Innerhalb des Headers können weitere Elemente schnittstellenspezifisch angepasst werden: Z.B. das Element <document_type_cd> erlaubt nur einen bestimmten Wert (z.B. V=“SDKT“). Dazu wird ein Datentyp („xy_document_type_cd_typ“) mit Ableitung vom Basisdatentyp („document_type_cd_typ“) definiert. Nur so meldet der Parser keine Fehler und es kann sichergestellt werden, dass der neue „xy_header_typ“ Datentyp nur eine Untermenge des Basisdatentyps „ehd_header_typ“ darstellt.

Die spezielle Einschränkung für den „xy_document_type_cd_typ“ Typ erfolgt über das V-Attribut mit **<xs:enumeration >** :“SDKT“.

Auch hier ist das blockDefault-Attribut bewusst angegeben, weil Substituierung von allen Elementen im Header-Bereich nicht erlaubt ist. Das Weglassen dieser XML-Prüfung könnte den einen oder anderen Schnittstellenentwickler dazu veranlassen, diese Lücke für seine Schnittstelle auszunutzen.

Der Code für die **<xs:restriction>**-Ableitung des Body sieht im xy_body.xsd wie folgt aus:

```
<xs:complexType name="xy_body_typ">
    <xs:complexContent>
        <xs:restriction base="ehd_body_typ">
            <xs:sequence>
                <xs:element name="my_element_1"/>
                <xs:element name="my_element_2"/>
                <!-- weitere Elemente möglich -->
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
```

XML-Code 47 xy_body.xsd

Die Einschränkung für den Body-Bereich erfolgt über das **<xs:restriction>** des Basistyps: „ehd_body_typ“. Innerhalb der **<xs:sequence >** können die eigentlichen schnittstellenspezifischen Elemente hinzugefügt werden.

Mit diesem Verfahren ist es dem Parser möglich komplette Schemas auf die Einhaltung der ehd-Kriterien zu überprüfen. Nachteil dieser Vorgehensweise ist, dass beim Headerdefinition alle schnittstellenspezifischen Datentypen explizit von Basisdatentypen abgeleitet werden müssen. Ein Datentyp, welcher nicht aus einem ehd-Basisdatentyp hervorgeht, wird als Fehler ausgewiesen. Der Schnittstellenentwickler ist damit auf die konsequente Ableitung von Basisdatentypen angewiesen. Das neudefinierte Schema wird durch die zahlreichen Ableitungen schnell unübersichtlich und schwernachvollziehbar.

Auf der anderen Seite werden mögliche Fehler im Schema und die Nichteinhaltung der ehd-Kriterien sofort vom Parser erkannt. Dieses Vorgehen trägt viel zur Qualitätssicherung der Schemas bei. Beim ersten Entwurf von eigenen Schemas wird der Parser sicherlich viele Fehler melden, weil nicht alle Datentypen konsequent abgeleitet wurden. Aber mit der Zeit werden alle Flüchtigkeitsfehler beseitigt und der Nutzen der automatischen Sicherstellung der Schemaqualität wird die Entwicklungsschwierigkeiten überwiegen.

10 Dokumentation

Für jede definierte ehd-Schnittstelle muss es eine Schnittstellenbeschreibung geben, die die Schnittstelle vollständig, eindeutig und widerspruchsfrei beschreibt.

Es ist erwünscht, dass diese Schnittstellenbeschreibung über das Internet zugänglich ist.

11 Versionierung

11.1 Versionierung der Schnittstelle

Bei jeglicher Änderung an einer Schnittstelle ist vor der Freigabe die Versionsnummer hochzuzählen. Bei größeren Änderungen kann auch ein Nummernbereich übersprungen werden, z.B. bei einer großen Revision von 1.34 auf 2.00.

Änderungen ausschließlich an der Dokumentation ohne Änderung der Schnittstelle führen zu keiner neuen Schnittstellen-Version wohl aber zu einer Beschreibungsversion.

12 Registrierung einer Schnittstelle

Um zu vermeiden, dass eine Satzart zufällig mehrfach definiert wird, bzw. eine Bezeichnung versehentlich schon verwendet wurde, muss jede Satzart bzw. der Name jeder Satzart registriert werden.

Dazu muss lediglich eine formlose eMail mit folgendem Inhalt an die emailadresse

ehd-register@kbv.de geschickt werden:

- Name der Satzart, bzw. der Schnittstelle
- Definierende Instanz (Organisation, Person),
- Kurzbeschreibung,
- Namespace
- evtl. Link auf Schnittstellendefinition (empfohlen)

13 Weiterentwicklung

Die ehd-Richtlinie ist bewusst so gestaltet, dass sie nicht an eine Institution gebunden ist.

Bei Anregungen zur Erweiterung, Änderung, Korrektur dieser Richtlinie senden Sie bitte eine email an: ehd@kbv.de

14 Anhang

14.1 Verweise

[1] XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001,
URL: http://www.w3.org/TR/xmlschema-0/

[2] XML Schema Part 1: Structures, W3C Recommendation 2 May 2001,
URL:http://www.w3.org/TR/xmlschema-1/

[3] XML Schema Part 2: Datatypes, W3C Recommendation 2 May 2001,
URL:http://www.w3.org/TR/xmlschema-2/

[4] SCIPHOX - Standardisation of Communication between Information Systems in Physician's Offices and Hospitals using XML, Working Draft 15, Oktober 2002, URL: http://www.sciphox.de/

[5] *XML Path Language*, James Clark and Steve DeRose, eds., W3C, 16 November 1999. See http://www.w3.org/TR/1999/REC-xpath-19991116

[6] Object Identifier (OID) Konzept für das deutsche Gesundheitswesen (wird in Kürze vom DIMDI http://www.dimdi.de/ veröffentlicht)

14.2 Schlüsseltabellen

Grundstruktur eines OID-Baumes

Gesundheitswesen Deutschland: 1.2.276.0.76

Experimental: 1.2.276.0.76.0

Organisationen:1.2.276.0.76.1

Dienste: 1.2.276.0.76.2

OID	Tabelle	Element
1.2.276.0.76.5.100	Dokumenttyp	document_type_cd
1.2.276.0.76.5.101	Dokumentbeziehungstyp	document_relationship.type_cd
1.2.276.0.76.5.102	Datenempfängertyp	intended_recipient.type_cd
1.2.276.0.76.5.103	Datenerzeugertyp	originator.type_cd
1.2.276.0.76.5.104	Datensendertyp	provider.type_cd
1.2.276.0.76.5.105	Funktionstyp	function_cd
1.2.276.0.76.5.106	Geltungsbereich	scope
1.2.276.0.76.5.107	Geltungsbereichstyp	scope.type_cd
1.2.276.0.76.5.108	Bearbeitungszustand	state
1.2.276.0.76.5.109	Schnittstelle	interface/id

Tabelle 6 Schlüsseltabellen