

**Elektronischer Datenaustausch für Verwaltung, Wirtschaft
und Transport (EDIFACT)**Teil 5: Regeln zur Generierung von XML-Schema-Dateien (XSD) aus
EDI(FACT)-Anwendungsbeschreibungen (ISO/TS 20625:2002)**DIN****16557-5**

ICS 35.240.60

Electronic data interchange for administration, commerce and transport (EDIFACT) — Part 5: Rules for generation of XML scheme files (XSD) on the basis of EDI(FACT) implementation guidelines (ISO/TS 20625:2002)

Échange de données informatisé pour l'administration, le commerce, et le transport (EDIFACT) — Partie 5: Règles pour la génération de fichiers de schéma XML (XSD) basés sur les guides de mise en oeuvre d'EDI(FACT) (ISO/TS 20625:2002)

Die Internationale Technische Spezifikation ISO/TS 20625:2002 „Electronic data interchange for administration, commerce and transport (EDIFACT) – Part 5: Rules for generation of XML scheme files (XSD) on the basis of EDI(FACT) implementation guidelines“ ist unverändert in diese Deutsche Norm übernommen worden.

Nationales Vorwort

Diese Norm wurde vom Normenausschuss Bürowesen (NBü), Arbeitsausschuss 3 „EDI/EDIFACT“ (NBü-AA 3) erarbeitet. Der NBü ist u. a. für normative Festlegungen im Anwendungsbereich des elektronischen Datenaustausches im weiteren Sinn auch Electronic Commerce zuständig.

DIN 16557 *Elektronischer Datenaustausch für Verwaltung, Wirtschaft und Transport (EDIFACT) besteht aus:*

- Teil 2: *Wörterbuch (zz. Entwurf)*
- Teil 3: *Allgemeine Einführung für Einheitliche Nachrichtentypen (UNSMs)*
- Teil 4: *Regeln für die Auszeichnung von UN/EDIFACT-Übertragungsdateien mit der Extensible Markup Language (XML) unter Einsatz von Document Type Definitions (DTDs)*
- Teil 5: *Regeln zur Generierung von XML-Schema-Dateien (XSD) aus EDI(FACT)-Anwendungsbeschreibungen*

Fortsetzung Seite 2 bis 57

Normenausschuss Bürowesen (NBü) im DIN Deutsches Institut für Normung e.V.

Deutsche Übersetzung

Elektronischer Datenaustausch für Verwaltung, Wirtschaft und Transport (EDIFACT)

Regeln zur Generierung von XML-Schema-Dateien (XSD) aus EDI(FACT)-Anwendungsbeschreibungen

Einleitung

Traditionelle EDI-Normen stellen eine Syntax bereit für die Implementierung von Dateninhalten aus unterschiedlichen Geschäftsprozessen durch die Verwendung von Datenelementen, Segmenten und Nachrichtentypen. XML stellt zunächst lediglich die Syntax zur Verfügung. Die derzeit stattfindende Neuerfindung von EDI in der XML-Welt führt zu ausufernden neuen Aufwänden, die das von EDI bisher ungelöste Ziel konterkarieren: die Anbindung von kleinen und mittelständischen Unternehmen (KMU) an elektronische Geschäftsprozesse.

Diese Norm soll aufzeigen, wie bestehendes EDI-Know-how mit der XML-Syntax verbunden werden kann, um EDI-Daten für XML-Anwender schnell und konsistent mit vorhandenen Anwendungen nutzbar zu machen.

EDIFACT Message Implementation Guidelines (MIGs) beschreiben die Umsetzung der Norm in einen konkreten Geschäftsprozess. Deshalb sind MIGs die geeignete Quelle zur Ableitung von XML-Schemata. Diese Norm soll den Prozess der Umsetzung normieren.

1 Anwendungsbereich

Diese Norm legt Regeln zur Ableitung von Vorgaben für XML-Schemata aus EDI-MIGs fest. Damit wird ein konsistentes Verfahren zur Darstellung von semantischen Sachverhalten gegeben.

Primär wird von MIGs für UN/EDIFACT ausgegangen. Diese Norm gilt grundsätzlich aber auch für andere EDI-Standards.

Diese Norm gilt nicht für DTDs.

2 Normative Verweisungen

Diese Norm enthält durch datierte oder undatierte Verweisungen Festlegungen aus anderen Publikationen. Diese normativen Verweisungen sind an den jeweiligen Stellen im Text zitiert, und die Publikationen sind nachstehend aufgeführt. Bei datierten Verweisungen gehören spätere Änderungen oder Überarbeitungen dieser Publikationen nur zu dieser Norm, falls sie durch Änderung oder Überarbeitung eingearbeitet sind. Bei undatierten Verweisungen gilt die letzte Ausgabe der in Bezug genommenen Publikation (einschließlich Änderungen).

ISO 8601:2000-12, *Data elements and interchange formats — Information interchange — Representation of dates and times*.

ISO 9735-1:1998-10, *Electronic data interchange for administration, commerce and transport (EDIFACT) — Application level syntax rules (Syntax version number: 4, Syntax release number: 1) — Part 1: Syntax rules common to all parts*.

3 Begriffe, Symbole und Abkürzungen

Für die Anwendung dieser Norm gelten die folgenden Begriffe, Symbole und Abkürzungen:

3.1

BSR (en: Basic Semantics Register)

Verzeichnis semantischer Grundbegriffe

3.2

BSU (en: Basic Semantic Unit)

semantischer Grundbegriff

3.3

DTD (en: Document Type Definition)

Standard-Strukturbeschreibung eines SGML- und XML-Dokumentes

3.4

EDI (en: Electronic Data Interchange)

elektronischer Datenaustausch

3.5

EDIFACT (en: Electronic Data Interchange for Administration, Commerce and Transport)

elektronischer Datenaustausch für Verwaltung, Wirtschaft und Transport

3.6

ELEMENT

syntaktischer Baustein, der Daten enthält und/oder Attribute besitzt

3.7

HTML (en: Hyper Text Markup Language)

Layout-orientierte Seitenbeschreibungssprache

3.8

MIG (en: Message Implementation Guideline)

Nachrichten-Anwendungsbeschreibung ^{N1)}

3.9

NAME

Ein Name in XML beginnend mit einem Buchstaben oder einem erlaubten Interpunktionszeichen, woran sich Buchstaben, Ziffern, Bindestriche, Unterstriche, Doppelpunkte oder Punkte anschließen. Letztere sind als Name-Zeichen bekannt. Namen, die mit "xml" oder mit einer Zeichenkette beginnen, die zu ('X'l'x') ('M'l'm') ('L'l'l') passt, sind für die Normung in XML reserviert

3.10

SGML (en: Standard Generalised Mark-up Language) ^{N2)}

3.11

Tag

Formatierungsanweisung oder semantische Auszeichnung

Nationale Fußnote 1: Message Implementation Guideline steht hier stellvertretend für Subsets, Conventions, Profiles, Anwendungsdokumentationen und ähnliche Begriffe zur Bezeichnung von Spezifikationen eines EDI-Standards für bestimmte Anwendungsfälle.

Nationale Fußnote 2: Hochsprache zur Ableitung von Auszeichnungssprachen wie HTML und XML

3.12

Template

Schablone bzw. vorgegebenes Referenzmuster, das mit der gesamten zu erkennenden Entität oder einem ihrer Teile verglichen wird

3.13

XLL (en: Extensible Link Language)

Verknüpfungssprache zu XML

3.14

XML (en: Extensible Markup Language)

Auszeichnungssprache u. a. zur Strukturierung von Daten

3.15

XSD (en: Extensible Scheme Definition)

XML-Schema

3.16

XSL (en: Extensible Stylesheet Language)

Visualisierungssprache zu XML

3.17

W3C (en: World Wide Web Consortium)

Internationales Konsortium zur Entwicklung technischer Spezifikationen für das WWW

4 Typische Inhalte von Message Implementation Guidelines

4.1 Ebene: MIG

- a) Identifikation des MIG;
- b) Identifikation des zugrunde liegenden EDIFACT-Verzeichnisses;
- c) Identifikation des Nachrichtentyps und gegebenenfalls des Branchen-Subsets;
- d) sonstiger Text.

4.2 Ebene: Nachrichtentyp

- a) Struktur des Nachrichtentyps (Segmente und Segmentgruppen) sowie die davon benutzten Teilmengen;
- b) Status (Standard versus Anwendung) der benutzten Segmente und Segmentgruppen;
- c) positionsspezifische Namen und Beschreibungen der Segmente und Segmentgruppen;
- d) Beispiele;
- e) Abhängigkeiten zwischen Segmenten und Segmentgruppen;
- f) sonstiger Text, Kommentare auf Nachrichtentyp-Ebene.

4.3 Ebene: Segmente und Datenelementgruppen

- a) Struktur der Segmente und Datenelementgruppen sowie der benutzten Teilmengen daraus;
- b) Status (Standard versus Anwendung) der Datenelemente und Datenelementgruppen;
- c) Abhängigkeiten zwischen Datenelementen und Datenelementgruppen innerhalb eines Segmentes und innerhalb des Nachrichtentyps;

- d) positionsspezifische Namen und Beschreibungen;
- e) Beispiele;
- f) sonstiger Text, Kommentare.

4.4 Ebene: Datenelement

- a) Eigenschaften der EDI-Datenelemente (Typ, Länge) und deren MIG- und positionsbezogene Anwendungseinschränkungen;
- b) positionsspezifische Namen und Beschreibungen der Datenelemente, gegebenenfalls zusätzlich eindeutige Tags und Beschreibungen, z.B. aus Datenrepositories wie dem ISO-BSR (siehe ISO/TS 16668);
- c) Beispiele;
- d) sonstiger Text, Kommentare;
- e) zugelassene Werte;
- f) Konstanten;
- g) explizit aufgeführte Codes aus EDIFACT- oder (ISO-/UN-)Verweis-Codelisten;
- h) explizit aufgeführte anwenderdefinierte Codes;
- i) implizit aufgeführte Codes aus EDIFACT- oder (ISO-/UN-)Verweis-Codelisten;
- j) implizit aufgeführte anwenderdefinierte oder andere Codes, die auf externe, nicht im Regelwerk aufgeführte Codelisten verweisen;
- k) Regeln, denen die Werte eines Datenelementes genügen müssen;
- l) Zuordnung zu Feldern von Anwendungen bzw. Flatfiles.

5 Anforderungen an Ableitungsregeln für Schemata

- a) Die unter Abschnitt 4 aufgelisteten fachlichen Informationen des MIG müssen so vollständig wie nötig in Schemata übernommen werden.
- b) Die Struktur des zugrunde liegenden MIG muss nachvollziehbar sein (XML und traditioneller EDI-Guide müssen strukturkompatibel sein.).
- c) Die resultierenden XML-Nachrichten sollten möglichst schlank sein.
- d) Von den verschiedenen Varianten, mit denen semantische Sachverhalte in XML dargestellt werden können, sollte die Norm im Idealfall eine als die verbindliche festlegen.
- e) Der Entwickler des MIG entscheidet, welche Daten fachlich wichtig und welche Strukturen für seine Anwendung sinnvoll sind. Er bestimmt damit, welche Strukturelemente in das Schema übernommen werden.

6 Regeln zur Erzeugung von XML-Schemata aus EDI-MIGs

ANMERKUNG Das Kürzel 'din' in den Beispielen dieses Abschnittes dient lediglich zu demonstrativen Zwecken und kann entweder ausgelassen oder durch andere passende Kürzel ersetzt werden.

6.1 Regel 1: Tag-Benennung

6.1.1 Variante 1

Die Namen der XML-Zielstruktur werden aus den EDI-Tags gebildet, die je nach Strukturebene (Segmentgruppe, Segment, Datenelementgruppe oder Datenelement) einen festgelegten Präfix erhalten:

„M_“ + Nachrichtentyp + [Suffix]	Beispiel: M_ORDERS
„G_“ + Segmentgruppe + [Suffix]	Beispiel: G_SG36 oder G_LIN_ALC
„S_“ + Segment + [Suffix]	Beispiel: S_LIN
„C_“ + Datenelementgruppe + [Suffix]	Beispiel: C_C082_2
„D_“ + Datenelement + [Suffix]	Beispiel: D_3035 oder D_3035_10

Der Suffix ist optional und entsteht bei unterschiedlicher semantischer Verwendung von EDI-Elementen.

Wird die XML-Schema-Datei aus einem EDIFACT-MIG erstellt, ist nur der Prefix „D_“ notwendig. Die anderen Prefixe sind in Hinblick auf andere EDI-Standards, bei denen Datenelementgruppen und Datenelemente numerische Tags haben, zu verwenden.

Die zweite Notation für Segmentgruppen-Tags kann dort angewendet werden, wo der zugrunde liegende EDI-Standard keine expliziten Segmentgruppen ausweist oder aus anderen Gründen die Notation der jeweiligen Trigger-Segmente bevorzugt wird. In diesen Fall ist die Schachtelung der Segmentgruppen als Sequenz ihrer Trigger-Segmente anzugeben.

Der W3C-XML-Standard verlangt „selbsterklärende Tags“. EDI(FACT)-Tags erfüllen diese Bedingung eher als natürlichsprachliche Tags, weil sie eine eingeführte Lingua Franca für EDI-Spezialisten darstellen.

Beispiel:

```
<xsd:element name="M_ORDERS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:D_1004"/>
      <xsd:element ref="din:D_2380"/>
      <xsd:element ref="din:D_2380_2"/>
      <xsd:element ref="din:G_SG2"/>
      <xsd:element ref="din:G_SG2_2"/>
      <xsd:element ref="din:D_6345" minOccurs="0" maxOccurs="5"/>
      <xsd:element ref="din:G_SG25" minOccurs="1" maxOccurs="10"/>
      <xsd:element ref="din:D_5004_2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

6.1.2 Variante 2

Aus geeigneten Kommentaren werden, wenn gewünscht, „sprechende“ Tags generiert. Diese Methode wird alternativ zugelassen. Wird mit "sprechenden" Tags gearbeitet, so wird die EDI-Herkunft des entsprechenden Elementes entweder über ein Attribut-Wert (siehe auch 6.9) oder mit anderen Dokumentationsmitteln eindeutig dokumentiert.

Beispiel:

```

<xsd:element name ="M_ORDERS">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:Order_number"/>
      <xsd:element ref="din:Order_date"/>
      <xsd:element ref="din:Delivery_date"/>
      <xsd:element ref="din:Buyer"/>
      <xsd:element ref="din:Seller"/>
      <xsd:element ref="din:Currency" minOccurs="0" maxOccurs="5"/>
      <xsd:element ref="din:Line_item_details" minOccurs="1"
        maxOccurs="10"/>
      <xsd:element ref="din:Total_order_value"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
...
<xsd:element name ="Name">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base ="string1..10">
        <xsd:attribute name="EDIPath" type="xsd:string"
          fixed="ORDERS.SG2.NAD.C080.3036(0120:040:01)"/>
        <!-- The attribute EDIPath contains the reference to the original
          EDI standard -->
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

```

6.2 Regel 2: Struktur

6.2.1 Gleichartige Namen führen zu aggregierten Elementen (siehe 6.10).

6.2.2 Wenn eine Unterscheidung zwischen verschiedenen semantischen Verwendungen des gleichen Datencontainers gewünscht ist, müssen auch verschiedene Tags oder Namen zugewiesen werden, entweder durch Ergänzung eines Suffixes zum EDI-Tag oder durch Verwendung unterschiedlicher Namen.

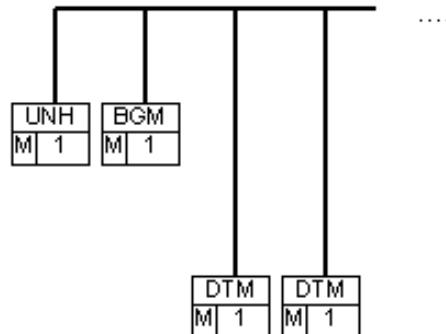
6.2.3 Das XML-Schema kann zusätzlich klammernde Elemente für Gruppen gleichartiger Nachrichten und die Übertragungsdatei selbst enthalten (analog UNG-UNE und UNB-UNZ in UN/EDIFACT).

6.2.4 Jede im MIG dokumentierte Verwendung eines EDI-Datencontainers (Nachrichtentyp, Segmentgruppe, Segment usw.) kann als eigenständiges XML-Element abgebildet werden. Die vorhandene EDI-Struktur ist Quelle der XML-Zielstruktur, somit muss das XML-Schema strukturkompatibel zum EDI-MIG sein. Die Menge der erzeugten XML-Elemente ist kleiner oder gleich der Menge der EDI-Elemente.

ANMERKUNG Die Art, wie der Anwender den MIG geschrieben hat, muss den Bedürfnissen des Geschäftsprozesses genügt haben. Dementsprechend muss auch das Schema strukturiert sein. Sind zum Beispiel im MIG "Dokumentendatum" und "Gefordertes Lieferdatum" in zwei getrennten Instanzen des DTM-Segmentes beschrieben, so entstehen daraus auch zwei getrennte XML-Elemente. Sind "Dokumentendatum" und "Gefordertes Lieferdatum" in einem DTM-Segment beschrieben, so entsteht auch nur ein XML-Element.

Beispiele für 6.2.1 und 6.2.2:**Variante 1:**

Der Beispiel-Guide enthält zwei DTM-Segmente (siehe Bild 1).



DTM (#1), Status M, Occurrence 1, Qualifier in DE 2005: 4, Name: Order_date
 DTM (#2), Status M, Occurrence 1, Qualifier in DE 2005: 2, Name: Delivery_date

Bild 1 — Nachrichtenaufbau-Diagramm des Beispiel-Guides mit zwei DTM-Segmenten

Die Überführung in ein XML-Schema nach 6.2.1 führt zu:

```
<xsd:element name="S_DTM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:D_2005"/>
      <xsd:element ref="din:D_2380"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="D_2005" type="din:D_2005">
  <xsd:annotation>
    <xsd:documentation>Type of date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="D_2380" type="xsd:decimal">
  <xsd:annotation>
    <xsd:documentation>Date/Time/Period</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

ANMERKUNG Das Element D_2005 ist ein "enumeration type" und enthält die zwei möglichen Werte '2' und '4'.

Alternativ würde die Anwendung der Regel 6.2.2 resultieren in:

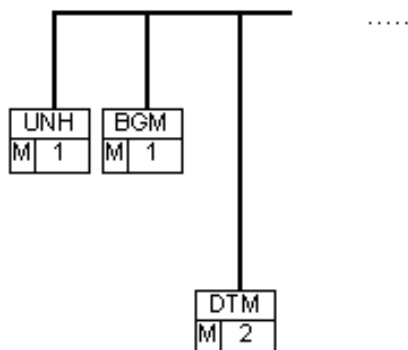
```
<xsd:element name="D_2380" type="xsd:decimal">
  <xsd:annotation>
    <xsd:documentation>Order date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="D_2380_2" type="xsd:decimal">
  <xsd:annotation>
    <xsd:documentation>Delivery date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

oder

```
<xsd:element name="Order_date" type="xsd:decimal"/>
<xsd:element name="Delivery_date" type="xsd:decimal"/>
```

Variante 2:

Eine implizite Dokumentation im Guide mit nur einem DTM-Segment (siehe Bild 2)



DTM, Status M, Occurrence 2, Qualifier in DE 2005: 2 und 4

Bild 2 — Nachrichtenaufbau-Diagramm des Beispiel-Guides mit einem DTM-Segment

Die Überführung in ein XML-Schema analog zum Beispiel in 6.2.1 führt zu:

```
<xsd:element name="S_DTM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:D_2005"/>
      <xsd:element ref="din:D_2380"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="D_2005" type="din:D_2005">
  <xsd:annotation>
    <xsd:documentation>Type of date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="D_2380" type="xsd:decimal">
  <xsd:annotation>
    <xsd:documentation>Order date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

Beispiel für 6.2.3:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="S_UNB">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="D_0004" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="D_0010" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="D_0017" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="D_0020" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="M_ORDERS" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

...

6.3 Regel 3: Struktur-Optimierung

Wenn es erwünscht ist, dass die XML-Strukturen möglichst flach sind, können die folgenden Regeln angewendet werden. Für die Integration in bestehende EDI-Systeme sollten jedoch die aus den Systemanforderungen abgeleiteten Mindestanforderungen für die Datenstruktur erhalten bleiben.

6.3.1 Zusammenfassende EDI-Objekte müssen nur insoweit nach XML überführt werden, als sie eine zusammenfassende Funktion erfüllen. Wenn das Segment nur ein Datenelement mit Geschäftsdaten enthält, gibt es keine zusammenfassende Funktion auf Segmentebene. Bei der Transformation in ein XSD-Schema kann diese Segmentebene daher weggelassen werden.

6.3.2 Im MIG nicht benutzte Elemente des Primär-Standards sind wegzulassen.

6.3.3 Konstante Qualifier oder konstante Codes sind nicht in die XML-Struktur zu übertragen (für ein bestimmtes Datenelement wurde nur ein Code dokumentiert). Die entsprechenden Datenelemente können in XML wegfallen.

Beispiel:

Aus:

```
<xsd:element name ="S_DTM">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:C_C507"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name ="C_C507">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:D_2005"/>
      <xsd:element ref="din:D_2380"/>
      <xsd:element ref="din:D_2379"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name ="D_2379" fixed ="102"/>
<xsd:element name ="D_2005" fixed ="4"/>
<xsd:element name ="D_2380" type ="xsd:decimal">
  <xsd:annotation>
    <xsd:documentation>Order date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

wird in Anwendung dieser Regel:

```
<xsd:element name ="D_2380" type ="xsd:decimal">
  <xsd:annotation>
    <xsd:documentation>Order date</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

Die Ebenen Segment und Datenelementgruppe werden nicht benötigt (konstante Qualifier) und entfallen daher.

6.4 Regel 4: Status

EDI-Status und Anwendungsstatus sind im MIG zu einem XML-Status zusammenzufassen. Der enger gefasste Status ist zu übernehmen.

Der Status „Muss“ (M) wird durch eine minimale Wiederholung von 1 ausgedrückt, der Status „Kann“ (K) durch eine minimale Wiederholung von 0. Die Angabe erfolgt im Attribut **minOccurs**.

Beispiele:

Kann:

Segmentgruppe	<code><xsd:element ref="din:G_SG7" minOccurs="0" maxOccurs="5"/></code>
Segment	<code><xsd:element ref="din:S_IMD" minOccurs="0" maxOccurs="1"/></code>
Datenelementgruppe	<code><xsd:element ref="din:C_C059" minOccurs="0" maxOccurs="1"/></code>
Datenelement	<code><xsd:element ref="din:D_4022" minOccurs="0" maxOccurs="1"/></code>

Muss:

Segmentgruppe	<code><xsd:element ref="din:G_LIN" minOccurs="1" maxOccurs="10"/></code>
Segment	<code><xsd:element ref="din:S_LIN" minOccurs="1" maxOccurs="1"/></code>
Datenelementgruppe	<code><xsd:element ref="din:C_C516" minOccurs="1" maxOccurs="1"/></code>
Datenelement	<code><xsd:element ref="din:D_0065" minOccurs="1" maxOccurs="1"/></code>

6.5 Regel 5: Maximale Häufigkeiten

Anwendungshäufigkeit im MIG wird zu XML-Häufigkeit. Der Wert wird in das XSD-Attribut **maxOccurs** übernommen.

Beispiel:

Segmentgruppe	<code><xsd:element ref="din:G_SG25" minOccurs="1" maxOccurs="10"/></code>
Segment	<code><xsd:element ref="din:S_LIN" minOccurs="1" maxOccurs="1"/></code>

Ab Syntax-Version 4 (ISO 9735-1) und der Implementierung entsprechender Directories trifft das analog auf Datenelementgruppen und Datenelemente zu.

6.6 Regel 6: Datenelement-Formate

6.6.1 Aus „an“ und „a“ wird „string“, aus „n“ wird „decimal“. Für die im Guide festgehaltenen Längen von alphanumerischen Datenelementen sind entsprechende abgeleitete simpleTypes zu generieren.

6.6.2 Datumsformate können in die XML-Datentypen "date", "timeInstant" und "time" überführt werden. Dabei ist jedoch eine Konvertierung der Daten notwendig. Die Repräsentation dieser Formate in XML ist:

```
date:      1999-05-31 (nach ISO 8601)
time:      13:20:00
dateTime:  1999-05-31T13:20:00
```

Beispiel:

```
<xsd:simpleType name="string1..70">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1"/>
    <xsd:maxLength value="70"/>
  </xsd:restriction>
</xsd:simpleType>
```

6.7 Regel 7: Codelisten und benutzerdefinierte Codes

6.7.1 Codierte Datenelemente werden als complexType definiert. Im Guide einzeln dokumentierte Codes bedeuten, nur diese Codes dürfen in der Anwendung genutzt werden. Es werden auch nur diese Codes in die XML-Struktur übernommen.

6.7.2 Sind im Guide keine Codes dokumentiert, gilt die gesamte Codeliste als zugelassen. Es wird auch die gesamte Codeliste in die XML-Struktur übertragen.

6.7.3 Wiederholt verwendete Codelisten können in ausgelagerten Dateien gespeichert werden.

6.7.4 Die Codenamen werden (optional) als Annotation zum Code gespeichert.

6.7.5 Nach Regel 3 (siehe 6.3) werden konstante Qualifier oder konstante Codes nicht in die XML-Struktur übertragen (für ein bestimmtes Datenelement wurde nur ein Code dokumentiert). Die entsprechenden Datenelemente fallen in XML weg, es sei denn, der Nutzer legt ausdrücklich fest, dass dieses Datenelement inklusive Code mit in die XML-Struktur übernommen wird (z. B. bei der Währung, Datenelement 6345 im Segment MOA).

Beispiele:

(1)

```
<xsd:element name="D_6345" type="D_6345"/>
<xsd:simpleType name="D_6345">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="DEM">
      <xsd:annotation>
        <xsd:documentation>Deutsche Mark</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="EUR">
      <xsd:annotation>
        <xsd:documentation>Euro</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="GBP">
      <xsd:annotation>
        <xsd:documentation>Pfund Sterling</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
```

(2)

```
<xsd:simpleType name="D_6347">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="1"/>
    <xsd:enumeration value="2"/>
    <xsd:enumeration value="3"/>
    <xsd:enumeration value="4"/>
    <xsd:enumeration value="5"/>
    <xsd:enumeration value="6"/>
    <xsd:enumeration value="7"/>
    ... etc. listing of the complete code list
  </xsd:restriction>
</xsd:simpleType>
```

(3)

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.din.de/example/orders"
  xmlns:din="http://www.din.de/example/orders"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
```

```

<include schemaLocation="CL_6411.xsd"/>
<xsd:element name="D_6411" type="din:CL_6411">
  <xsd:annotation>
    <xsd:documentation>Measure unit</xsd:documentation>
  </xsd:annotation>
</xsd:element>
...

```

External file with codes:

```

<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.din.de/ example/orders"
  xmlns:din="http://www.din.de/ example/orders"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:simpleType name="CL_6411">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="ACR"/>
    <xsd:enumeration value="AMH"/>
    .....

```

(4)

```

<xsd:simpleType name="CL_6411">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="TNE">
      <xsd:annotation>
        <xsd:documentation>Tonne (1000 kg) *</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="KGM">
      <xsd:annotation>
        <xsd:documentation>Kilogram *</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="GRM">
      <xsd:annotation>
        <xsd:documentation>Gram *</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="DZN">
      <xsd:annotation>
        <xsd:documentation>Dozen</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>

```

(5)

```

<xsd:element name="D_6345" type="din:D_6345"/>
<xsd:simpleType name="D_6345">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="DEM">
      <xsd:annotation>
        <xsd:documentation>Deutsche Mark</xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>

```

6.8 Regel 8: Namen von EDI-Objekten

6.8.1 Standard- oder nutzerdefinierte Namen von Segmentgruppen, Segmenten, Datenelementgruppen und Datenelementen können als Attribut „annotation“ ins Schema übernommen werden. Je XML-Element ist nur ein EDI-Name als Attribut zulässig.

6.8.2 Gibt es für ein EDI-Datenobjekt sowohl Standard- als auch nutzerdefinierte Namen, so sollte nur der nutzerdefinierte Name übernommen werden.

ANMERKUNG Letzter Satz bezieht sich auf die Nutzdatennamen, die unter der Bezeichnung Nutzdatenname, BSU o. Ä. in MIGs zu finden sind. Auf diese Weise bleibt die XML-Datei schlank und über den Parser ist eine einfache fachliche Zuordnung möglich.

Beispiele:

(1)

```
<xsd:element name = "S_DTM">
  <xsd:annotation>
    <xsd:documentation>Date/Time/Period</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:D_2005"/> ...
```

(2)

```
<xsd:element name = "S_DTM">
  <xsd:annotation>
    <xsd:documentation>Order or delivery date </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="din:D_2005"/> ...
```

6.9 Regel 9: Mappinginformationen

6.9.1 Enthält der Guide Mappinginformationen, so können "Ankerpunkte" als Attribut gebildet werden, die eine einfache Implementierung des XML-Austauschformates in EDI-Subsysteme ermöglichen.

6.9.2 Die EDI(FACT)-Quelle wird in einem Attribut "EDISource" angegeben. Die Notation vereint in sich die Besonderheit der Anwendungsdokumentation und die originalen Angaben aus dem jeweiligen Release wie z. B. dem EDIFACT-Directory.

Dabei gelten folgende Regeln:

- a) Der Pfad wird angegeben in Form von "SegmentGruppe\Segment\DatenelementGruppe\Datenelement" oder "SegmentGruppe\Segment\Datenelement".
- b) Die Segmentgruppe kann mehrfach vorkommen, um die Ebenen der EDI(FACT)-Struktur abzubilden.
- c) Bei mehreren semantischen Varianten von Segmentgruppen kann in eckigen Klammern das qualifizierende Segment, der Qualifier und der entsprechende Wert des Qualifiers angegeben werden.

Am Ende wird in Klammern die Sequenznummer des Segmentes im originalen EDIFACT-Nachrichtentyp wie im Directory angegeben sowie nach Doppelpunkt die Sequenznummer des Datenelementes (Datenelementgruppe oder einfaches Datenelement) innerhalb des betreffenden Segmentes sowie gegebenenfalls des Gruppendatenelementes innerhalb der Datenelementgruppe.

Beispielsweise liest sich die Notation (0120:020:02) wie folgt: Zähler des Segments im Standard : Zähler der Datenelementgruppe oder des Datenelementes : gegebenenfalls Zähler des Gruppendatenelementes in der Datenelementgruppe.

Beispiele:

(1)

```
<xsd:element name="D_3433">
  <xsd:annotation>
    <xsd:documentation>BIC of buyer's bank</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="Mapping_anchor" type="xsd:string" use="fixed"
          value="BIC-BB"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

(2)

```
<xsd:element name="D_3433">
  <xsd:annotation>
    <xsd:documentation>BIC of buyer's bank</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="EDIPath" type="xsd:string" use="fixed"
          value="SG2 [NAD.3035=BY] .FII.C088.3433 (0140:030:01)"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

6.10 Regel 10: Aggregation gleichnamiger Datencontainer

Für den Fall, dass Anwendungsszenarien mit unterschiedlichen Nachrichtentypen bestehen und der Anwender gleichnamige Datencontainer zusammenfassen und gegebenenfalls nachrichtenübergreifend einheitlich beschreiben will, gelten folgende Regeln zur Aggregation:

6.10.1 Struktur

Aggregierte Datencontainer enthalten mindestens die Unterelemente, die im MIG verwendet und dokumentiert wurden. Die Sequenz dieser Elemente muss mit der Reihenfolge im EDI-Standard übereinstimmen.

6.10.2 Status

Im aggregierten Datencontainer wird dem Unterelement der Status optional zugewiesen, sofern er in einer Stelle im MIG als optional/konditional notiert wurde.

Beispiel: ORDERS DTM 2379 Status: R, IFTMIN DTM 2379 Status: O
→ XML-Status: O

6.10.3 Format

Es wird das am weitesten gefasste Format innerhalb der Nachricht oder des Nachrichtensets verwendet.

Beispiel: ORDERS DTM 2380 Format: an..12, IFTMIN DTM 2380 Format: an..35
→ XML-Format: string..35

6.10.4 Codeliste

Für codierte Datenelemente wird jeweils eine aggregierte Codeliste gebildet, die alle innerhalb der Nachricht oder des Nachrichtensets für dieses Datenelement verwendbaren Codes beinhaltet.

Beispiel: ORDERS DTM 2380 Codeliste: 102;103, IFTMIN DTM 2380 Codeliste: 103;203
→ XML-Codeliste: 102;103;203

Anhang A (informativ)

Beispiel für die Umsetzung von EDIFACT nach XML

ANMERKUNG Das in diesem Anhang aufgeführte Beispiel bedient sich lediglich aus praktischen Gründen deutschsprachiger Tags. Die Verwendung fremdsprachiger Tags wird damit nicht ausgeschlossen. Der Status R bedeutet "erforderlich" (required) und der Status O bedeutet "optional". Bei beiden handelt es sich um Anwendungsstatus-Informationen; sie haben die gleiche Bedeutung wie M (Muss) und C (Konditional). Der Status N bedeutet "nicht verwendet" (en: not used).

A.1 Zugrunde liegende EDIFACT-Struktur

A.1.1 Allgemeines

Grundlage für die zu generierende XML-Struktur bildet eine Implementierung des EDIFACT-Nachrichtentyps ORDERS (Bestellung) mit folgendem Aufbau.

A.1.2 Nachrichtenstruktur

A.1.2.1 Tabellarische Segmentübersicht

Tabelle A.1 — Tabellarische Segmentübersicht der zugrunde liegenden EDIFACT ORDERS

Nr.	Tag	St	Wdh.	Inhalt
01	UNH	M	1	Kopf der Nachricht
02	BGM	M	1	Dokumentenart und -nummer
03	DTM	M	1	Bestelldatum
04	DTM	M	1	Lieferdatum
	SG2	R	1	Käufer
05	NAD	M	1	Identifikation des Käufers
06	FII	O	1	Bankverbindung des Käufers
	SG3	O	1	Umsatzsteuernummer des Käufers
07	RFF	M	1	Umsatzsteuernummer
	SG5	O	1	Ansprechpartner beim Käufer
08	CTA	M	1	Verantwortlicher Angestellter beim Käufer
09	COM	O	1	Telefonnummer
10	COM	O	1	Telefaxnummer
	SG2	R	1	Verkäufer
11	NAD	M	1	Identifikation des Verkäufers
	SG7	O	1	Währung
12	CUX	M	1	Bestellwährung
	SG25	R	10	Positionsdaten
13	LIN	M	1	Artikelnummer des Lieferanten
14	IMD	O	1	Artikelbeschreibung

Tabelle A.1 (fortgesetzt)

Nr.	Tag	St	Wdh.	Inhalt
15	QTY	O	1	Artikelmenge
16	MOA	O	1	Gesamtbetrag
	SG27	O	1	Positionspreis
17	PRI	M	1	Einzelpreis
18	UNS	M	1	Trennung Positions- und Summenteil
19	MOA	R	1	Gesamtsumme
20	UNT	M	1	Abschluss der Nachricht

A.1.2.2 Nachrichtenaufbau-Diagramm

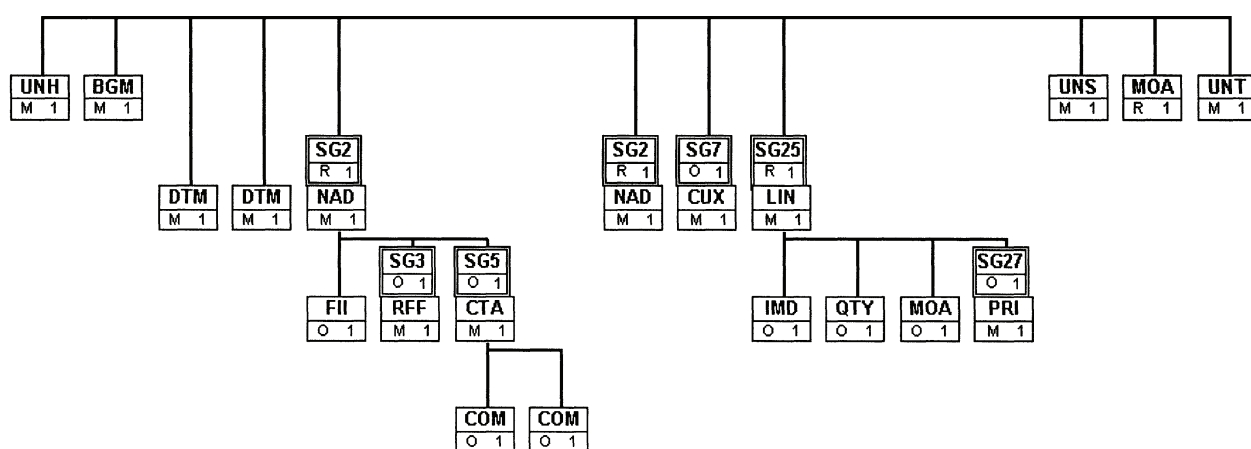


Bild A.1 — Nachrichtenaufbau-Diagramm der zugrunde liegenden EDIFACT ORDERS