

# 1 GML-Codierungsregeln für INTERLIS

*Dieses Dokument ist ein Entwurf, hat noch offene Punkte und enthält zusätzlich auch nicht ausformulierte Ideen für Alternativen (Varianten).*

## 1.1 Einleitung

In diesem Dokument werden die Regeln definiert, um aus einem INTERLIS-Modell ein GML-Applikationsschema abzuleiten.

## 1.2 Voraussetzungen

Dieses Dokument setzt INTERLIS und GML Kenntnisse voraus. Der Text ist aber mit Beispielen ergänzt, so dass dem Inhalt auch vom Nicht-Experten gefolgt werden kann.

## 1.3 Allgemeine Ziele

- Es wird eine möglichst genaue Modellabbildung angestrebt
- Das resultierende GML-Applikationsschema lässt sich durch GML-konforme Software nutzen
- Das resultierende GML-Applikationsschema lässt sich auch im Kontext Web-Dienste (insbesondere WFS) nutzen
- Dieses Dokument verwendet INTERLIS 2.3 und GML 3.2.x. Da INTERLIS 1 mehr oder weniger 1:1 nach INTERLIS 2.3 übersetzt werden kann, gilt es aber entsprechend auch für INTERLIS 1.

## 1.4 Referenzdokumente

### INTERLIS

[http://www.interlis.ch/interlis2/docs23/ili2-refman\\_2006-04-13\\_d.zip](http://www.interlis.ch/interlis2/docs23/ili2-refman_2006-04-13_d.zip)

### GML

[http://portal.opengeospatial.org/files/?artifact\\_id=20509](http://portal.opengeospatial.org/files/?artifact_id=20509)

[http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO\\_19136\\_Schemas/](http://standards.iso.org/ittf/PubliclyAvailableStandards/ISO_19136_Schemas/)

<http://schemas.opengis.net/gml/3.2.1/>

## 1.5 Was ist INTERLIS?

Das INTERLIS-Referenzhandbuch besteht aus zwei Teilen:

- eine Datenbeschreibungssprache, und
- XML-Codierungsregeln.

Werden die XML-Codierungsregeln auf ein mit Hilfe der Datenbeschreibungssprache definiertes Datenmodell angewendet, entsteht ein zum Datenmodell passendes Transferformat.

Für die XML-Codierungsregeln wird im vorliegenden Dokument eine Alternative definiert. Im Anhang B ist ein Beispiel Datenmodell in INTERLIS inkl. einer Transferdatei gemäss diesem Datenmodell.

## 1.6 Was ist GML?

GML ist ein Satz von Basis XML-Schemas und ein Satz von Regeln, wie diese Schemas in eigenen Schemas (sog. Applikationsschemas) für die Definition von Transferformaten zu verwenden sind. Aus der Kenntnis der Basis-Schemas und der Regeln für Applikationsschemas lässt sich somit auch eine Datenbeschreibung herleiten.

### Anmerkung

Weil die Regeln für GML-Applikationsschemas definiert sind, kennt man nicht nur die zulässige Folge von XML-Elementen (aufgrund des XML-Schemas), sondern weiss, welche XML-Elemente Klassen darstellen und welche Elemente Attribute dieser Klassen sind. Das diese Beschreibung weniger genau ist als mit INTERLIS, ist dann eine andere Sache, um die es aber in diesem Dokument nicht primär geht.

Im Anhang B ist ein Beispiel Applikationsschema inkl. einer Transferdatei gemäss diesem Schema.

### Anmerkung

Die GML 3.1 Schemas validieren z.T. nicht! Das ist nicht nur ein GML Problem, s.a.

<http://schemas.opengis.net/gml/3.1.1/readme.txt>

In der GML 3.2 Spez. steht dazu:

### 21.2.6 Property Type Derivation

...

NOTE As derivation-by-restriction of property types has created problems with commonly used XML parsers in the past, all instances of such derivations have been removed from the GML schema. It is recommended to avoid derivation-by-restriction in property types in application schemas, too.

## 1.7 Was sind Codierungsregeln?

Codierungsregeln definieren, wie aus einem fachlichen Datenmodell durch die Anwendung von Regeln das Transferformat abgeleitet werden kann (modell-basierter Datentransfer). Die Regeln sind dabei so formuliert, dass sie auf ein beliebiges Datenmodell angewendet werden können.

## 1.8 Grundsätzliche Anwendungsfälle von XML

Es gibt grundsätzlich die folgenden Anwendungsfälle, die das Design einer XML Struktur stark beeinflussen:

- Texte/Dokumente (z.B. XHTML, DocBook). In diesem Fall stehen die Lesbarkeit für den Menschen und die korrekte Reihenfolge/Strukturierung der XML-Elemente im Vordergrund. Datentypen sind weniger wichtig.
- Meldungen (z.B. Liegenschaftsmutation oder im Kontext AJAX und Web-Services). In diesem Fall steht die einfache Verarbeitung im Vordergrund, oft nimmt man darum bewusst Redundanzen der Daten in Kauf.
- Massendatentransfer (z.B. Amtliche Vermessungsschnittstelle). In diesem Fall stehen die effiziente Verarbeitung und die Grösse der XML-Datei im Vordergrund.
- Riesige Datenmenge (z.B. Bilder, Messreihen). In diesem Fall steht die Grösse der XML-Datei so dominant im Vordergrund, dass die Markierung der einzelnen Informationsteile teilweise oder vollständig entfällt
- durch Mensch editierte "Daten" (z.B. XML Schema-Dokumente, ant build Skripts). In diesem Fall steht die Lesbarkeit für den Menschen im Vordergrund. Im Gegensatz zum Fall "Texte/Dokumente", sind hier auch Datentypen wichtig.

Weitere Aspekte sind

- Applikations-interner Gebrauch. Man kann die XML-Struktur genau nach den Bedürfnissen dieser Applikation ausrichten.
- Unterschiedliche Nutzungszusammenhänge, z.B. verschieden 'harte' Restriktionen, weil in frühen Verarbeitungszusammenhängen in einer Kette von Verarbeitungsvorgängen noch nicht alle Informationen vorhanden sind.

- Applikations-übergreifender Gebrauch. Man kann die XML-Struktur nicht nach den Bedürfnissen einer Applikation ausrichten, die verschiedenen Applikationen haben aber typischerweise doch eine ähnliche Vorstellung von der Realität.
- Fachbereichs-übergreifender Gebrauch. Die verschiedenen Applikationen haben typischerweise eine unterschiedliche Vorstellung der Realität. Beispiel: In der Steuerverwaltung stehen die Personen im Vordergrund, Grundstücke und Gebäude kommen auch vor. Im Grundbuch stehen die dinglichen Rechte im Vordergrund, Grundstücke und Personen kommen auch vor. In der Amtlichen Vermessung stehen Geometrien von Grundstücken und Gebäuden im Vordergrund, Personen kommen nicht vor.
- Existierendes Schema oder existierender Schemadesignstil
- Metadaten, insbesondere auch Schema, als Teil der Daten
- Generische, statt fachspezifische Struktur, z.B.:
 

```
<object class="Person">
  <attr name="Name" value="Eisenhut"/>
  <attr name="Vorname" value="Claude"/>
</object>
```
- Erweiterbarkeit: kann man weitere Eigenschaften (z.B. für Versionierung) einfach in die bestehenden Strukturen einbetten oder darauf verweisen?
- Änderungstoleranz: kann man die Datenstrukturen leicht ändern, ohne dass man die Programme vollständig neu entwickeln *muss*?

## 1.9 Grundsätzliches zur Codierung von Objekten in XML

Es gibt ein paar grundsätzliche Entwurfsentscheidungen zu treffen, die meisten davon sind allerdings schon durch die GML-Spezifikation getroffen worden.

### 1.9.1 Objekteigenschaften (Attribute im INTERLIS-Modell)

Objekteigenschaften lassen sich als XML-Attribute oder XML-Subelemente codieren. Durch GML wird die Variante XML-Subelemente verlangt.

### 1.9.2 Assoziationen

Assoziationen lassen sich als eigenständige Objekte ("Zwischentabelle"), als einseitige Referenz, als gegenseitige Referenz oder als Objekteinbettung codieren. Durch GML wird keine bestimmte Variante verlangt.

### 1.9.3 Typdiskriminator

Der Typ eines Objektes kann als XML-Attribut (in der Regel xsi:type) oder XML-Element codiert werden.

Durch GML wird die Variante XML-Elemente verlangt. GML bezeichnet das als "Object-Property-Object Pattern".

### 1.9.4 Vererbung

Die Vererbungshierarchie kann im XML-Schema ausgedrückt werden oder nach verschiedenen Verfahren zu inhaltsgleichen Strukturen abgebildet werden (die aber die Vererbung nicht mehr zum Ausdruck bringen; ähnlich wie beim O/R-Mapping).

### 1.9.5 Semantische Transformation

Im Prinzip kann man auch semantische Transformationen durchführen, solange sie den Inhalt nicht ändern, nur anders strukturieren. Zum Beispiel könnte man zwei über eine 1:1 Assoziation verbundene Klassen zu einem einzigen XML-Element zusammenfassen.

## **1.10 Grundsätzlich Unterschiede zu den Codierungsregeln gemäss INTERLIS Referenzhandbuch**

### **1.10.1 Codierungsregeln**

Das INTERLIS-Referenzhandbuch definiert Instanz-Codierungsregeln. In diesem Dokument werden Schema-Codierungsregeln definiert und die Instanzen ergeben sich aus dem so erzeugten Schema.

### **1.10.2 Aufbau eines Transfers**

Das INTERLIS-Referenzhandbuch unterteilt einen Transfer in Vorspann und Datenbereich. Auf den Vorspann wird in diesem Dokument verzichtet, da dies durch eine GML-konforme-Software als Daten interpretiert würde (statt als Metadaten zum Transfer).

#### **Variante**

Angaben aus dem Vorspann als GML-Metadaten transferieren.

### **1.10.3 Lesen von erweiterten/übersetzten Modellen**

Das INTERLIS-Referenzhandbuch definiert als Teil des Vorspanns eine Abbildungstabelle, so dass ein Leseprogramm das für ein bestimmtes Datenmodell programmiert oder konfiguriert worden ist, Daten von Erweiterungen (oder Übersetzungen) dieses Datenmodells ohne Kenntnis der erweiterten Modelldefinitionen lesen kann. In diesem Dokument wird auf eine solche Abbildungstabelle verzichtet. Ein Leseprogramm muss also um Erweiterungen korrekt ignorieren zu können, das erweiterte Datenmodell kennen. Für übersetzte INTERLIS-Datenmodelle wird kein GML-Applikationsschema erzeugt, d.h. das Transferformat ändert sich bei einer Übersetzung nicht.

#### *Anmerkung*

Ist die Handhabung einer unbekanntem Modellerweiterung mit GML (ohne Spezialsoftware) überhaupt möglich?

Unbekannt im Sinne von Schema nicht verfügbar: nein.

Unbekannt im Sinne von Schema verfügbar, aber nicht implementiert: ja.

### **1.10.4 Zeichenvorrat**

Das INTERLIS-Referenzhandbuch definiert einen eingeschränkten Zeichenvorrat (Anhang B im Referenzhandbuch).

Dieses Dokument definiert standardmässig den Unicode Zeichenumfang. Ein eingeschränkter Zeichenvorrat muss im Rahmen einer Transfergemeinschaft definiert werden.

### **1.10.5 Transferarten**

Das INTERLIS-Referenzhandbuch definiert drei Transferarten: FULL, INITIAL oder UPDATE. Zusätzlich können Angaben zur Konsistenz übermittelt werden: COMPLETE, INCOMPLETE, INCONSISTENT, ADAPTED.

Dieses Dokument beschreibt wie im Rahmen von Web-Diensten einzelne Objekte transferiert werden. Optional wird der klassische Datei-basierte Transfer definiert (entspricht in INTERLIS: FULL und COMPLETE).

#### *Anmerkung*

Inkrementeller Transfer lässt sich mit GML Standardsoftware nicht realisieren, da GML keine solche Semantik definiert.

Konsistenzangaben könnte man als GML-Metadaten codieren.

## **1.11 Schema-Codierungsregeln**

### **1.11.1 Einleitung**

Für die Formalisierung der Transferformat-Ableitungsregeln wird die im INTERLIS-Referenzhandbuch in Kapitel 2.1 eingeführte EBNF-Notation benutzt.

### 1.11.2 Zeichencodierung

Es gelten die XML-Regeln (d.h. der Unicode Zeichensatz und UTF-8 (empfohlen) oder UTF-16 als Zeichencodierung).

### 1.11.3 Allgemeine Regeln

Das generierte XML-Schema kann beliebig viele Kommentare (<!-- -->) oder zusätzliche xsd:annotation Elemente enthalten.

### 1.11.4 Abbildung der INTERLIS-Modellelementnamen auf XML-Schema-Namen

Jedes INTERLIS-Datenmodell enthält einen eigenen XML-Schema-Namensraum. Für die XML-Element- und Typ-Definitionen werden die unqualifizierten INTERLIS-Namen verwendet. Bei Konflikten wird der qualifizierte Namen verwendet. Die Namen auf Stufe MODEL haben Vorrang gegenüber Namen aus der Stufe TOPIC.

### 1.11.5 Abbildung von Basistypen

Für Modell unabhängige Basistypen (z.B. HALIGNMENT, SurfacePropertyType) wird ein INTERLIS-Basischema definiert.

#### Variante

In jedem generierten Schema immer wieder neu definieren (=weniger Schema-Dateien).

### 1.11.6 Allgemeiner Aufbau des XML-Schemas

Jedes INTERLIS-Modell wird in ein XML-Schema, in einem Schemadokument, abgebildet.

```
ModelDef = '<xsd:schema
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="%ModelNamespaceIdentifier%"
            targetNamespace="%ModelNamespaceIdentifier%"
            elementFormDefault="qualified"
            attributeFormDefault="unqualified"
            xmlns:gml="http://www.opengis.net/gml/3.2"
            { 'xmlns:%nsprefix% namespace=
              "%ImportedModelNamespaceIdentifier%" ' }
            '>'
IliModelInfo
'<xsd:import
  namespace="http://www.opengis.net/gml/3.2"/>'
{ ModelImport }
{ ClassDef | AssociationDef | DomainDef }
{ TopicDef }
'</xsd:schema>'.
IliModelInfo = '<xsd:annotation>
  <xsd:appinfo source="http://www.interlis.ch/ili2c">
    <ili2c:model>%ModelName%</ili2c:model>
    <ili2c:modelVersion>%ModelVersion%
      </ili2c:modelVersion>
    <ili2c:modelAt>%ModelAt%</ili2c:modelAt>
  </xsd:appinfo>
</xsd:annotation>'.
ModelImport = '<xsd:import
  namespace="%ImportedModelNamespaceIdentifier%"/>'.

```

#### Beispiel

INTERLIS	INTERLIS 2.3;  MODEL ModelA (de) AT "mailto:ceis@localhost"
----------	--

	<pre> VERSION "2008-03-31" =   IMPORTS Units;  END ModelA. </pre>
GML	<pre> &lt;xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"   xmlns="http://www.interlis.ch/INTERLIS2.3/GML32/ModelA"   targetNamespace=     "http://www.interlis.ch/INTERLIS2.3/GML32/ModelA"   elementFormDefault="qualified"   attributeFormDefault="unqualified"   xmlns:gml="http://www.opengis.net/gml/3.2"   xmlns:ns1=     "http://www.interlis.ch/INTERLIS2.3/GML32/Units"&gt; &lt;xsd:annotation&gt;   &lt;xsd:appinfo source="http://www.interlis.ch/ili2c"&gt;     &lt;ili2c:model&gt;ModelA&lt;/ili2c:model&gt;     &lt;ili2c:modelVersion&gt;2008-03-31&lt;/ili2c:modelVersion&gt;     &lt;ili2c:modelAt&gt;mailto:ceis@localhost&lt;/ili2c:modelAt&gt;   &lt;/xsd:appinfo&gt; &lt;/xsd:annotation&gt; &lt;xsd:import namespace="http://www.opengis.net/gml/3.2"/&gt; &lt;xsd:import namespace   =" http://www.interlis.ch/INTERLIS2.3/GML32/Units"/&gt; &lt;/xsd:schema&gt; </pre>

### 1.11.7 Codierung von Wertebereichen

Wertebereichsdefinitionen werden wie folgt gebildet:

DomainDef = DomainDefComplex | DomainDefSimple.

```

DomainDefComplex = '<xsd:complexType name="%DomainName%">
  <xsd:complexContent>
    <xsd:restriction base="%BaseType%">
      <!-- any applicable xsd restrictions -->
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>'.

```

```

DomainDefSimple = '<xsd:simpleType name="%DomainName%">
  <xsd:restriction base="%BaseType%">
    <!-- any applicable xsd restrictions -->
  </xsd:restriction>
</xsd:simpleType>'.

```

Je nach Abbildungsregel des entsprechenden INTERLIS-Datentyps (siehe Kapitel 1.4.13) wird die Regel DomainDefComplex oder DomainDefSimple angewendet.

#### Beispiel

INTERLIS	<pre> DOMAIN   LKoord = COORD 480000.00 .. 850000.00 [m] {CHLV03[1]},           60000.00 .. 320000.00 [m] {CHLV03[2]},           ROTATION 2 -&gt; 1;   Horizontbezeichnung = TEXT*20; </pre>
GML	<pre> &lt;xsd:complexType name="LKoord"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:restriction base="gml:DirectPositionType"&gt;     &lt;/xsd:restriction&gt;   &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; </pre>

	<pre> &lt;xsd:simpleType name="Horizontbezeichnung"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;     &lt;xsd:maxLength value="10"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; </pre>
--	---

### 1.11.8 Codierung von Themen

Themen haben in INTERLIS zwei Funktionen: einerseits definieren sie einen eigenen Namensraum (getrennt vom Namensraum des Modells), andererseits definieren sie einen Behälter. Die Eigenschaft als Namensraum wird nicht abgebildet. Die Eigenschaft als Behälterbeschreibung wird wie folgt abgebildet.

```

TopicDef = '<xsd:complexType name="%TopicName%MemberType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureMemberType">
      <xsd:sequence>
        <xsd:choice>'
          { '<xsd:element ref="%ClassName%"/>' }
        '</xsd:choice>'
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="%TopicName%" type="%TopicName%Type"
  substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="%TopicName%Type">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="member"
          type="%TopicName%MemberType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attributeGroup
        ref="gml:AggregationAttributeGroup"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>'.

```

Beispiel	
INTERLIS	<pre> TOPIC Bodenbedeckung =    CLASS BoFlaechen =   END BoFlaechen;    CLASS Gebaeude =   END Gebaeude;  END Bodenbedeckung; </pre>
GML	<pre> &lt;xsd:complexType name="BodenbedeckungMemberType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureMemberType"&gt;       &lt;xsd:sequence&gt;         &lt;xsd:choice&gt;           &lt;xsd:element ref="BoFlaechen"/&gt;           &lt;xsd:element ref="Gebaeude"/&gt;         &lt;/xsd:choice&gt;       &lt;/xsd:sequence&gt;     &lt;/xsd:extension&gt;   &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; </pre>

	<pre> &lt;/xsd:sequence&gt; &lt;/xsd:extension&gt; &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; &lt;xsd:element name="Bodenbedeckung"   type="BodenbedeckungType"   substitutionGroup="gml:AbstractFeature"/&gt; &lt;xsd:complexType name="BodenbedeckungType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;xsd:element name="member"           type="BodenbedeckungMemberType"           minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;/xsd:sequence&gt;       &lt;xsd:attributeGroup         ref="gml:AggregationAttributeGroup"/&gt;     &lt;/xsd:extension&gt;   &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; </pre>
--	---

## Variante

Jedes Thema als getrenntes XML-Schema abbilden.

### 1.11.9 Codierung von Klassen, Strukturen und Assoziationen

Klassen und Strukturen werden wie folgt abgebildet:

```

ClassDef = '<xsd:element name="%ClassName%"
  type="%ClassName%Type"
  substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="%ClassName%Type">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        { AttributeDef | EmbeddedRoleDef }
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>'.

```

Im XML-Schema lässt sich somit nicht mehr erkennen ob es sich um eine CLASS oder STRUCTURE handelt.

Eigenständige Assoziationen (siehe auch Kapitel 1.4.10) werden wie folgt abgebildet:

```

AssociationDef = '<xsd:element name="%ClassName%"
  type="%ClassName%Type"
  substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="%ClassName%Type">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        { RoleDef } { AttributeDef | EmbeddedRoleDef }
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>'.

```

Für die Reihenfolge der Attribute, Rollen, Eingebetten-Rolle innerhalb der Klasse/Assoziation gilt: Zuerst werden alle Rollen der Basisklasse, dann alle Attribute



der Basisklasse, dann alle Eingebettete-Rollen der Basisklasse, dann alle Attribute der Erweiterung, dann alle Eingebetteten der Erweiterung codiert, etc. (Zwiebelprinzip). Innerhalb der gleichen Erweiterungsstufe werden die Attribute und Rollen gemäss ihrer Definitionsreihenfolge in der Modelldatei codiert. Die Eingebetteten-Rollen werden innerhalb der gleichen Erweiterungsstufe alphabetisch aufsteigend sortiert. Parameter werden mit einer Ausnahme, wie sie im Kapitel 1.4.11 Codierung von Grafikdefinitionen angegeben ist, nicht übertragen.

### Beispiel

INTERLIS	<pre>CLASS Horizont =     Horizontbezeichnung : MANDATORY TEXT*10; END Horizont;</pre>
GML	<pre>&lt;xsd:element name="Horizont" type="HorizontType"     substitutionGroup="gml:AbstractFeature"/&gt; &lt;xsd:complexType name="HorizontType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;xsd:element name="Horizontbezeichnung"&gt;           &lt;xsd:simpleType&gt;             &lt;xsd:restriction base="xsd:normalizedString"&gt;               &lt;xsd:maxLength value="10"/&gt;             &lt;/xsd:restriction&gt;           &lt;/xsd:simpleType&gt;         &lt;/xsd:element&gt;       &lt;/xsd:sequence&gt;     &lt;/xsd:extension&gt;   &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt;</pre>

### Variante

STRUCTURE basierend auf gml:AbstractObject

#### 1.11.10 Codierung von Sichten

Zur Codierung von Sichten vgl. Kapitel 3.2.4 Transferierbare Objekte im Referenzhandbuch. Als Attribute des Sichtobjekts werden nur diejenigen Attribute übertragen, welche in der Sicht explizit unter ATTRIBUTE bzw. implizit mit ALL OF angegeben wurden.

#### 1.11.11 Codierung von Beziehungen

Beziehungen werden auf zwei Arten codiert: eingebettet oder nicht eingebettet. Eine eingebettete Beziehung wird als Sub-Element von einer, an der Assoziation beteiligten, Klasse codiert. Die Instanz einer nicht eingebetteten Beziehung (Link) wird wie eine Instanz einer Klasse codiert.

Beziehungen werden immer eingebettet, ausser

- wenn sie mehr als zwei Rollen haben oder
- wenn bei beiden (Basis-)Rollen die maximale Kardinalität grösser 1 ist oder
- wenn für die Beziehung eine OID gefordert wird oder
- bei gewissen themenübergreifenden Beziehungen (s. unten).

Falls bei einer der beiden (Basis-)Rollen die maximale Kardinalität grösser 1 ist, wird bei der Ziel-Klasse dieser Rolle eingebettet. Wenn diese Ziel-Klasse in einem anderen Topic definiert ist als die (Basis-)Assoziation, kann nicht eingebettet werden.

Falls bei beiden (Basis-)Rollen die maximale Kardinalität kleiner gleich 1 ist, wird bei der Ziel-Klasse der zweiten Rolle eingebettet. Wenn diese Ziel-Klasse in einem anderen

Topic definiert ist als die (Basis-)Assoziation und die Ziel-Klasse der ersten Rolle im selben Topic definiert ist wie die (Basis-)Assoziation, wird bei der Ziel-Klasse der ersten Rolle eingebettet (d.h., wenn die Ziel-Klassen der beiden Rollen in einem anderen Topic definiert sind als die (Basis-)Assoziation, kann nicht eingebettet werden).

**Variante**

Referenz so einbetten, dass Kardinalität im XML-Schema sichtbar ist

**Variante**

Bei Komposition, Ziel-Klasse einbetten

**Variante**

Rolle so abbilden, dass als Instanz Referenz oder eingebettetes Objekt codiert werden kann

1.11.11.1 *Eingebettete Beziehungen*

Eingebettete Beziehungen werden wie ein Strukturattribut der Klasse, bei der die Beziehung eingebettet wird, übertragen.

Die Unterstruktur hat folgenden Aufbau:

```
EmbeddedRoleDef =
  '<xsd:element name="%RoleName%" type="gml:ReferenceType">
    <xsd:annotation>
      <xsd:appinfo>
        <gml:targetElement>%ClassName%</gml:targetElement>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:element>'
```

**OPEN: ORDER\_POS, EmbeddedLinkStruct**

Für %RoleName% muss der Name der Rolle angegeben werden, welche auf das gegenüberliegende Objekt verweist (die andere Rolle wird nicht codiert).

**Beispiel**

<p>INTERLIS</p>	<pre>CLASS Liegenschaft = END Liegenschaft;  CLASS Person = END Person;  ASSOCIATION =   Eigentum -- {0..*} Liegenschaft;   Eigentuermer -- {1} Person; END;</pre>
<p>GML</p>	<pre>&lt;xsd:element name="Liegenschaft" type="LiegenschaftType"   substitutionGroup="gml:AbstractFeature"/&gt; &lt;xsd:complexType name="LiegenschaftType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;xsd:element name="Eigentuermer"           type="gml:ReferenceType"&gt;           &lt;xsd:annotation&gt;             &lt;xsd:appinfo&gt;               &lt;gml:targetElement&gt;Person&lt;/gml:targetElement&gt;             &lt;/xsd:appinfo&gt;           &lt;/xsd:annotation&gt;         &lt;/xsd:element&gt;       &lt;/xsd:sequence&gt;</pre>

	<pre> &lt;/xsd:extension&gt; &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; &lt;xsd:element name="Person" type="PersonType"               substitutionGroup="gml:AbstractFeature"/&gt; &lt;xsd:complexType name="PersonType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;/xsd:sequence&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; </pre>
--	--

### 1.11.11.2 Nicht eingebettete Beziehungen

Nicht eingebettete Beziehungen werden wie Objektinstanzen von Klassen übertragen. Hinweis: Für Beziehungen ohne expliziten Namen wird der (Klassen)Name durch zusammenhängen der einzelnen Rollennamen gebildet (d.h. z.B. %RoleName1RoleName2%).

Rollen werden wie Attribute behandelt. Die Rollen selbst werden wie folgt codiert:

```

RoleDef =
  '<xsd:element name="%RoleName%" type="gml:ReferenceType">
    <xsd:annotation>
      <xsd:appinfo>
        <gml:targetElement>%ClassName%</gml:targetElement>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:element>'.

```

#### OPEN: ORDER\_POS

Ob die Referenz auf ein Objekt in der gleichen Transferdatei zeigt, lässt sich nicht immer erkennen (nur wenn die Referenz ein Dokument relativer XML-Fragment-Identifikator ist).

Beispiel	
<b>INTERLIS</b>	<pre> CLASS Person = END Person;  CLASS Verein = END Verein;  ASSOCIATION =   DauerhafteVereinigung -- {0..*} Verein;   Mitglied -- {1..*} Person; END; </pre>
<b>GML</b>	<pre> &lt;xsd:element name="Verein" type="VereinType"               substitutionGroup="gml:AbstractFeature"/&gt; &lt;xsd:complexType name="VereinType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;/xsd:sequence&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; </pre>

	<pre> &lt;/xsd:complexType&gt; &lt;xsd:element name="Person" type="PersonType"               substitutionGroup="gml:AbstractFeature" /&gt; &lt;xsd:complexType name="PersonType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;/xsd:sequence&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; &lt;xsd:element name="DauerhafteVereinigungMitglied"               type="DauerhafteVereinigungMitgliedType"               substitutionGroup="gml:AbstractFeature" /&gt; &lt;xsd:complexType name="DauerhafteVereinigungMitgliedType"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:extension base="gml:AbstractFeatureType"&gt;       &lt;xsd:sequence&gt;         &lt;xsd:element name="DauerhafteVereinigung"                       type="gml:ReferenceType"&gt;           &lt;xsd:annotation&gt;             &lt;xsd:appinfo&gt;               &lt;gml:targetElement&gt;Verein&lt;/gml:targetElement&gt;             &lt;/xsd:appinfo&gt;           &lt;/xsd:annotation&gt;         &lt;/xsd:element&gt;         &lt;xsd:element name="Mitglied"                       type="gml:ReferenceType"&gt;           &lt;xsd:annotation&gt;             &lt;xsd:appinfo&gt;               &lt;gml:targetElement&gt;Person&lt;/gml:targetElement&gt;             &lt;/xsd:appinfo&gt;           &lt;/xsd:annotation&gt;         &lt;/xsd:element&gt;       &lt;/xsd:sequence&gt;     &lt;/xsd:extension&gt;   &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; </pre>
--	---

### 1.11.12 Codierung von Grafikdefinitionen

Für jede Grafikdefinition werden im Transfer die von der Grafikdefinition referenzierten Signaturklassen (Sign-ClassRef) übertragen. Die Objektinstanzen der Signaturklassen werden durch das Ausführen der Grafikdefinitionen auf einem konkreten Inputdatensatz erzeugt. Parameter werden dabei wie Attribute codiert.

### 1.11.13 Codierung von Attributen

#### 1.11.13.1 Allgemeine Regeln für die Codierung von Attributen

Jedes Attribut einer Objektinstanz (einschliesslich komplexer Attribute wie COORD, POLYLINE, SURFACE, AREA, STRUCTURE, LIST OF, BAG OF, etc.) wird wie folgt codiert:

```

AttributeDef = ExplicitTypeAttribute
              | ImplicitTypeAttribute
              | StructAttribute
              | ReferenceAttribute.

```

```

ExplicitTypeAttribute =
    '<xsd:element name="%AttributeName%" type="%DataType%"
     [ 'minOccurs="0"' ] '/>'.

ImplicitTypeAttribute =
    '<xsd:element name="%AttributeName%"
     [ 'minOccurs="0"' ] '>'
    '<xsd:simpleType>
     <xsd:restriction base="%BaseType%">
     <!-- any applicable xsd restrictions -->
     </xsd:restriction>
     </xsd:simpleType>
     </xsd:element>'.

StructAttribute = '<xsd:element name="%AttributeName%"
     [ 'minOccurs="0"' ] '>'
    '<xsd:sequence>
     <xsd:element ref="%StructureName%"
     [ 'maxOccurs="%maxCardinality%"' ] '/>
     </xsd:sequence>
     </xsd:element>'.

ReferenceAttribute =
    '<xsd:element name="%AttributeName%" type="gml:ReferenceType"
     [ 'minOccurs="0"' ] '>'
    '<xsd:annotation>
     <xsd:appinfo>
     <gml:targetElement>%ClassName%</gml:targetElement>
     </xsd:appinfo>
     </xsd:annotation>
     </xsd:element>'.

```

### Beispiel

INTERLIS	<pre> DOMAIN   Prozent100 = 0 .. 100;  STRUCTURE Koernung =   Ton : Prozent100;   Schluff : Prozent100;   Sand : Prozent100; END Koernung;  CLASS Profil =   Lage : COORD 480000.00 .. 850000.00,          60000.00 .. 320000.00;   KoernungsklasseOberboden : Koernung;   KoernungsklasseUnterboden : Koernung;   Bodenpunktzahl : 0 .. 100; END Profil; </pre>
GML	<pre> &lt;xsd:simpleType name="Prozent100"&gt;   &lt;xsd:restriction base="xsd:integer"&gt;     &lt;xsd:minInclusive value="0"/&gt;     &lt;xsd:maxInclusive value="100"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:element name="Koernung" type="KoernungType" </pre>

```

substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="KoernungType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="Ton" type="Prozent100"
          minOccurs="0"/>
        <xsd:element name="Schluff" type="Prozent100"
          minOccurs="0"/>
        <xsd:element name="Sand" type="Prozent100"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Profil" type="ProfilType"
  substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="ProfilType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="Lage" minOccurs="0"
          type="gml:PointPropertyType">
        </xsd:element>
        <xsd:element name="KoernungsklasseOberboden">
          <xsd:sequence>
            <xsd:element ref="Koernung"/>
          </xsd:sequence>
        </xsd:element>
        <xsd:element name="KoernungsklasseUnterboden">
          <xsd:sequence>
            <xsd:element ref="Koernung"/>
          </xsd:sequence>
        </xsd:element>
        <xsd:element name="Bodenpunktzahl" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minInclusive value="0"/>
              <xsd:maxInclusive value="100"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

### 1.11.14 Abbildung der INTERLIS-Typen auf XML-Schema-Typen

#### 1.11.14.1 Übersicht

<i>INTERLIS</i>	<i>Abbildung</i>	<i>Complex/Simple-Type</i>
TextType	xsd:string	Simple
EnumerationType	xsd:string / gml:CodeWithAuthorityType	Simple/Complex
EnumTreeValueType	xsd:string /	Simple/Complex

	<code>gml:CodeWithAuthorityType</code>	
<code>AlignmentType</code>	<code>xsd:string</code>	Simple
<code>BooleanType</code>	<code>xsd:boolean</code>	Simple
<code>NumericType</code>	<code>xsd:integer / xsd:decimal / xsd:double</code>	Simple
<code>FormattedType</code>	<code>xsd:string / xsd:date / xsd:time / xsd:dateTime</code>	Simple
<code>CoordinateType</code>	<code>gml:DirectPositionType</code>	Complex
<code>OIDType</code>	<code>gml:CodeWithAuthorityType</code>	Complex
<code>BlackboxType</code>	<code>xsd:anyType / xsd:base64Binary</code>	Complex/Simple
<code>ClassType</code>	<code>xsd:string</code>	Simple
<code>AttributePathType</code>	<code>xsd:string</code>	Simple
<code>LineType</code>	<code>gml:CurveType / gml:PolygonType</code>	Complex

#### 1.11.14.2 Codierung von Zeichenketten

Attribute/Wertebereiche vom Basistyp TEXT werden als `xsd:normalizeString`, MTEXT als `xsd:string`, NAME als `xsd:string` und URI als `xsd:anyURI` codiert. Falls aus dem Modell ableitbar, wird `xsd:restriction` codiert.

#### Beispiel

INTERLIS	<pre> DOMAIN TestText = TEXT; TestTextLen = TEXT*10; TestMText = MTEXT; TestMTextLen = MTEXT*10; TestURI = URI; TestName = NAME; </pre>
GML	<pre> &lt;xsd:simpleType name="TestText"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestTextLen"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;     &lt;xsd:maxLength value="10"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestMText"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestMTextLen"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;     &lt;xsd:maxLength value="10"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestURI"&gt;   &lt;xsd:restriction base="xsd:anyURI"/&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestName"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;     &lt;xsd:maxLength value="255"/&gt;     &lt;xsd:pattern value="OPEN"/&gt;   &lt;/xsd:restriction&gt; </pre>

	<pre>&lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt;</pre>
--	---

### 1.11.14.3 Codierung von Aufzählungen

In INTERLIS können Aufzählungen verfeinert und/oder erweitert werden. Beispiele:

```
Farbe = (rot, gelb, gruen);
FarbeVerfeinert EXTENDS Farbe = (rot (dunkelrot, orange, karmin));
FarbeErweitert EXTENDS Farbe = (schwarz, blau);
```

Da sich mit XML-Schema Aufzählungen weder verfeinern noch erweitern lassen, wird eine INTERLIS-Aufzählung nur dann in eine XML-Schema Aufzählung abgebildet, wenn der Wertebereich oder das Attribut als FINAL markiert ist. Ansonsten wird die Aufzählung als gml:CodeWithAuthority abgebildet.

#### Beispiel

INTERLIS	<pre>DOMAIN   FarbeFinal (FINAL) = (rot, gelb, gruen);   Farbe = (rot, gelb, gruen);</pre>
GML	<pre>&lt;xsd:simpleType name="FarbeFinal"&gt;   &lt;xsd:restriction base="xsd:string"&gt;     &lt;xsd:enumeration value="rot"/&gt;     &lt;xsd:enumeration value="gelb"/&gt;     &lt;xsd:enumeration value="gruen"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:complexType name="Farbe"&gt;   &lt;xsd:restriction     base="gml:CodeWithAuthorityType"/&gt; &lt;/xsd:complexType&gt;</pre> <p>OPEN: Dictionary</p>

Die Aufzählungswerte werden in beiden Fällen wie folgt codiert:

```
EnumValue = ( EnumElement-Name { '.' EnumElement-Name } ) .
```

Für die Codierung von Aufzählungswerten (unabhängig davon, ob der Wertebereich nur die Blätter oder auch die Knoten umfasst) wird die Syntax für Aufzählungskonstanten angewendet (Regel EnumValue). Das Zeichen # wird dabei weggelassen.

Die vordefinierten Textausrichtungstypen HALIGNMENT und VALIGNMENT werden als Teil eines im Anhang gegebenen Basis-Schemas definiert. Für den Typ BOOLEAN wird der entsprechende XML-Schema Typ xsd:boolean verwendet.

#### Variante

DomainDef/AttributeDef ohne FINAL als xsd:string ohne xsd:enumeration abbilden

#### Variante

DomainDef/AttributeDef ohne FINAL als xsd:union von xsd:string und xsd:enumeration abbilden

#### Variante

DomainDef/AttributeDef ohne FINAL als xsd:string und kompliziertes xsd:pattern abbilden

### 1.11.14.4 Codierung von numerischen Datentypen

Numerische Datentypen werden je nach Unter- und Obergrenzwert als xsd:integer, xsd:decimal oder xsd:double codiert. Falls aus dem Modell ableitbar, wird xsd:restriction codiert.



Beispiel	
INTERLIS	<pre> DOMAIN   TestInt = 1 .. 10;   TestDec = 1.0 .. 10.0;   TestDouble = 0.123 .. 0.234; </pre>
GML	<pre> &lt;xsd:simpleType name=" TestInt"&gt;   &lt;xsd:restriction base="xsd:integer"&gt;     &lt;xsd:minInclusive value="1"/&gt;     &lt;xsd:maxInclusive value="10"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestDec"&gt;   &lt;xsd:restriction base="xsd:decimal"&gt;     &lt;xsd:minInclusive value="1.0"/&gt;     &lt;xsd:maxInclusive value="10.0"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="TestDouble"&gt;   &lt;xsd:restriction base="xsd:double"&gt;     &lt;xsd:minInclusive value="0.123"/&gt;     &lt;xsd:maxInclusive value="0.234"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; </pre>

### Variante

als gml:MeasureType falls im Modell eine Angabe zur Einheit vorhanden ist.

#### Anmerkung

Man kann die Einheit im Schema nicht angeben. Mit MeasureType erreicht man nur, dass in den Daten die Einheit codiert werden kann.

#### 1.11.14.5 Codierung von formatierten Wertebereichen

Formatierte Wertebereiche, die INTERLIS.XMLDate, INTERLIS.XMLTime oder INTERLIS.XMLDateTime referenzieren, werden in die entsprechenden XML-Schema Datumsstypen (xsd:date, xsd:time, xsd:dateTime) abgebildet. Alle anderen formatierten Wertebereiche werden in ein xsd:string mit einem dem Format entsprechenden xsd:pattern abgebildet.

Beispiel	
INTERLIS	<pre> STRUCTURE GregorianDate =   Year: 1582 .. 2999;   SUBDIVISION Month: 1 .. 12;   SUBDIVISION Day: 1 .. 31; END GregorianDate;  DOMAIN   BuchungsDatum = FORMAT INTERLIS.XMLDate     "2002-01-01" ..     "2007-12-31";   StartZeit = FORMAT INTERLIS.XMLTime     "00:00:00.000" ..     "23:59:59.999";   MessZeitpunkt = FORMAT INTERLIS.XMLDateTime     "2002-01-01T00:00:00.000" ..     "2007-12-31T23:59:59.999";   EigenesDatum = FORMAT BASED ON GregorianDate     ( Year "Y" Month "M" Day "D"); </pre>
GML	<pre> &lt;xsd:simpleType name="BuchungsDatum"&gt; </pre>

	<pre> &lt;xsd:restriction base="xsd:date"/&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="StartZeit"&gt;   &lt;xsd:restriction base="xsd:time"/&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="MessZeitpunkt"&gt;   &lt;xsd:restriction base="xsd:dateTime"/&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="EigenesDatum"&gt;   &lt;xsd:restriction base="xsd:string"&gt;     &lt;xsd:pattern value="OPEN"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; </pre>
--	---

#### 1.11.14.6 Codierung von Gefässen

Die XML-Variante des Typs BLACKBOX wird als xsd:any codiert, die binäre Variante als xsd:base64Binary.

##### Beispiel

INTERLIS	<pre> DOMAIN   BlackboxXml = BLACKBOX XML;   BlackboxBinary = BLACKBOX BINARY; </pre>
GML	<pre> &lt;xsd:complexType name="BlackboxXml"&gt;   &lt;xsd:sequence&gt;     &lt;xsd:any namespace="##any"       minOccurs="0" maxOccurs="unbounded"       processContents="lax"/&gt;   &lt;/xsd:sequence&gt; &lt;/xsd:complexType&gt; &lt;xsd:simpleType name="BlackboxBinary"&gt;   &lt;xsd:restriction base="xsd:base64Binary"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; </pre>

#### 1.11.14.7 Codierung von Klassentypen

Der Klassentyp wird als xsd:normalizedString codiert.

##### Beispiel

INTERLIS	<pre> DOMAIN   InterlisClassRef = CLASS; </pre>
GML	<pre> &lt;xsd:simpleType name="InterlisClassRef"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; </pre>

Der Wert enthält den vollständig qualifizierten Klassen-, Struktur- oder Assoziationsnamen (z.B. DM01AVCH24D.FixpunkteKategorie1.LFP1).

#### 1.11.14.8 Codierung von Attributpfadtypen

Der Attributpfadtyp wird als xsd:normalizedString codiert.

##### Beispiel

INTERLIS	<pre> DOMAIN   InterlisAttributeRef = ATTRIBUTE; </pre>
GML	<pre> &lt;xsd:simpleType name="InterlisAttributeRef"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; </pre>

Der Wert enthält den vollständig qualifizierten Klassennamen gefolgt vom durch einen Punkt abgetrennten Attributnamen (z.B. Grunddatensatz.Fixpunkte.LFP.Nummer).

#### 1.11.14.9 Codierung von Koordinaten

Koordinaten werden als gml:DirectPositionType codiert.

Beispiel	
INTERLIS	DOMAIN LKoord = COORD 480000.00 .. 850000.00 [m] {CHLV03[1]}, 60000.00 .. 320000.00 [m] {CHLV03[2]}, ROTATION 2 -> 1;
GML	<xsd:complexType name="LKoord"> <xsd:complexContent> <xsd:restriction base="gml:DirectPositionType"> </xsd:restriction> </xsd:complexContent> </xsd:complexType>

Angaben zu Achsen-Wertebereich, Einheit, Referenzsystem und Richtungssinn lassen sich im GML-Applikationsschema nicht festhalten.

#### 1.11.14.10 Codierung von Linienzügen

Linienzüge werden als gml:CurvePropertyType codiert.

Beispiel	
INTERLIS	DOMAIN LKoord = COORD 480000.00 .. 850000.00 [m] {CHLV03[1]}, 60000.00 .. 320000.00 [m] {CHLV03[2]}, ROTATION 2 -> 1; Strassenachse : POLYLINE WITH (ARCS,STRAIGHTS) VERTEX LKoord WITHOUT OVERLAPS>0.10;
GML	<xsd:complexType name="LKoord"> <xsd:complexContent> <xsd:restriction base="gml:DirectPositionType"> </xsd:restriction> </xsd:complexContent> </xsd:complexType> <xsd:complexType name="Strassenachse"> <xsd:complexContent> <xsd:restriction base="gml:CurvePropertyType"> </xsd:restriction> </xsd:complexContent> </xsd:complexType>

Angaben zu Stützpunkt-Wertebereich und zulässigen Kurvenstück-Formen lassen sich im GML-Applikationsschema nicht festhalten.

**OPEN: DIRECTED**

**OPEN: OVERLAPS**

#### Variante

Optimieren zu LineString , falls im Linienwertebereich nur Geraden zulässig sind

#### 1.11.14.11 Codierung von Einzelflächen und Gebietseinteilungen

Einzelflächen und Gebietseinteilungen werden als gml:Polygon codiert. Da es gml:PolygonPropertyType nicht gibt, muss die entsprechende XML-Struktur als Teil des Applikationsschemas definiert werden, um das vorgeschriebene Object-Property-Object Muster einzuhalten.

<b>Beispiel</b>	
<b>INTERLIS</b>	<pre>DOMAIN   LKoord = COORD 480000.00 .. 850000.00 [m] {CHLV03[1]},            60000.00 .. 320000.00 [m] {CHLV03[2]},            ROTATION 2 -&gt; 1;   GebaeudeFlaeche : SURFACE WITH (ARCS,STRAIGHTS) VERTEX                     LKoord WITHOUT OVERLAPS&gt;0.10;</pre>
<b>GML</b>	<pre>&lt;xsd:complexType name="LKoord"&gt;   &lt;xsd:complexContent&gt;     &lt;xsd:restriction base="gml:DirectPositionType"&gt;     &lt;/xsd:restriction&gt;   &lt;/xsd:complexContent&gt; &lt;/xsd:complexType&gt; &lt;xsd:complexType name="GebaeudeFlaeche"&gt;   &lt;xsd:sequence&gt;     &lt;xsd:element ref="gml:Polygon"/&gt;   &lt;/xsd:sequence&gt;   &lt;xsd:attributeGroup ref="gml:OwnershipAttributeGroup"/&gt; &lt;/xsd:complexType&gt;</pre>

Angaben zu Stützpunkt-Wertebereich und zulässigen Kurvenstück-Formen lassen sich im GML-Applikationsschema nicht festhalten, ebenso wenig dass es sich um eine Gebietseinteilung handelt.

#### OPEN: LINE ATTRIBUTES

#### OPEN: OVERLAPS

#### 1.11.14.12 Codierung von Typen für Objektidentifikation

Typen für Objektidentifikationen werden als `xsd:normalizedString` abgebildet.

<b>Beispiel</b>	
<b>INTERLIS</b>	<pre>DOMAIN   BFS_EGID = OID 1 .. 999999999;   UUID = OID TEXT*36;</pre>
<b>GML</b>	<pre>&lt;xsd:simpleType name="BFS_EGID"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;xsd:simpleType name="UUID"&gt;   &lt;xsd:restriction base="xsd:normalizedString"&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt;</pre>

#### Variante

Als `gml:CodeWithAuthorityType`

#### Variante

Als `xsd:anyURI`

#### 1.11.14.13 Codierung von zusätzlichen Kurvenstück-Formen

#### OPEN

### 1.12 Instanz-Codierungsregeln

Zusätzlich zu den Schema-Codierungsregeln gelten die in diesem Abschnitt definierten Instanz-Regeln.

#### **1.12.1 Codierung von mehreren Behältern (TOPICs) in einer Transferdatei**

Werden in einer Transferdatei mehrere Behälter transferiert (typischerweise der Fall) wird als Wurzel-XML-Element "TRANSFER" verwendet. Dieses Element und die zugehörigen Typen sind im Anhang A definiert.

#### **1.12.2 Codierung von Objekt und Behälter-Identifikationen**

Die Transferidentifikation eines Objektes oder Behälters wird als gml:id codiert.  
Die stabile Identifikation des Objektes oder Behälters wird als gml:identifizier codiert.

#### **1.12.3 Codierung von Geometrien**

Alle Geometrien müssen inline codiert werden, auch wenn das GML-Basischema Referenzen zulassen würde.

## Anhang A

### Vordefiniertes Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.interlis.ch/INTERLIS2.3/GML32/INTERLIS"

xmlns:ili2c="http://www.interlis.ch/INTERLIS2.3/GML32/INTERLIS/ili2c"
  targetNamespace=
    "http://www.interlis.ch/INTERLIS2.3/GML32/INTERLIS"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xmlns:gml="http://www.opengis.net/gml/3.2"
>
<xsd:annotation>
  <xsd:appinfo source="http://www.interlis.ch/ili2c">
    <ili2c:model>INTERLIS</ili2c:model>
    <ili2c:modelVersion>2005-06-16</ili2c:modelVersion>
    <ili2c:modelAt>http://www.interlis.ch</ili2c:modelAt>
  </xsd:appinfo>
</xsd:annotation>
<xsd:import namespace="http://www.opengis.net/gml/3.2"/>
  <xsd:simpleType name="HALIGNMENT">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Left"/>
      <xsd:enumeration value="Center"/>
      <xsd:enumeration value="Right"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="VALIGNMENT">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Top"/>
      <xsd:enumeration value="Cap"/>
      <xsd:enumeration value="Half"/>
      <xsd:enumeration value="Base"/>
      <xsd:enumeration value="Bottom"/>
    </xsd:restriction>
  </xsd:simpleType>

<xsd:complexType name="TRANSFERMemberType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureMemberType">
      <xsd:sequence>
        <xsd:choice>
          <xsd:element ref="gml:AbstractFeature"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="TRANSFER" type="TRANSFERType"
  substitutionGroup="gml:AbstractFeature"/>
```

```
<xsd:complexType name="TRANSFERType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="baskets"
                      type="TRANSFERMemberType"
                      minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="gml:AggregationAttributeGroup"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:schema>
```

## Anhang B

### Kleines Beispiel

#### **INTERLIS 2-Modell**

INTERLIS 2.3;

MODEL Beispiel (de)  
AT "mailto:ceis@localhost"  
VERSION "2008-03-31" =

DOMAIN

LKoord = COORD 100.00 .. 300.00, 100.00 .. 300.00;

TOPIC Bodenbedeckung =

CLASS BoFlaechen =

Art : MANDATORY (  
Gebaeude,  
befestigt,  
humusiert,  
Gewaesser,  
bestockt,  
vegetationslos);

Form : MANDATORY AREA WITH (ARCS,STRAIGHTS) VERTEX Beispiel.LKoord  
WITHOUT OVERLAPS>0.10;

END BoFlaechen;

CLASS Gebaeude =

AssNr : MANDATORY TEXT\*6;  
UNIQUE AssNr;  
END Gebaeude;

ASSOCIATION GebaeudeFlaechen =

Gebaeude -- {0..\*} Gebaeude;  
Flaechen -- {1} BoFlaechen;  
END GebaeudeFlaechen;

END Bodenbedeckung;

END Beispiel.

#### **INTERLIS 1-Modell**

TRANSFER Beispiel;

DOMAIN

LKoord = COORD2 100.00 100.00



```

300.00 300.00;
MODEL Beispiel
  TOPIC Bodenbedeckung =
    TABLE BoFlaechen =
      Art: (Gebaeude,
           befestigt,
           humusiert,
           Gewaesser,
           bestockt,
           vegetationslos);
      Form: AREA WITH (STRAIGHTS, ARCS) VERTEX LKoord
            WITHOUT OVERLAPS > 0.10;
      NO IDENT
    END BoFlaechen;
    TABLE Gebaeude =
      AssNr: TEXT*6;
      Flaechen: -> BoFlaechen // Art = Gebaeude //;
      IDENT
      AssNr; !! Annahme AssNr sei eindeutig.
      Flaechen; !! Dem Gebaeude ist genau eine
                !! Flaechen zugeordnet
    END Gebaeude;

  END Bodenbedeckung.
END Beispiel.
FORMAT FREE;
CODE BLANK = DEFAULT, UNDEFINED = DEFAULT, CONTINUE = DEFAULT;
TID = ANY;
END.

```

### GML-Applikationsschema

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.interlis.ch/INTERLIS2.3/GML32/Beispiel"
  targetNamespace="http://www.interlis.ch/INTERLIS2.3/GML32/Beispiel"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xmlns:gml="http://www.opengis.net/gml/3.2">
<xsd:import namespace="http://www.opengis.net/gml/3.2"/>
<xsd:complexType name="LKoord">
  <xsd:complexContent>
    <xsd:restriction base="gml:DirectPositionType">
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="BoFlaechen" type="BoFlaechenType"
  substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="BoFlaechenType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">

```

```

<xsd:sequence>
  <xsd:element name="Art">
    <xsd:simpleType>
      <xsd:restriction base="xsd:normalizedString">
        <xsd:enumeration value="Gebaeude"/>
        <xsd:enumeration value="befestigt"/>
        <xsd:enumeration value="humusiert"/>
        <xsd:enumeration value="Gewaesser"/>
        <xsd:enumeration value="bestockt"/>
        <xsd:enumeration value="vegetationslos"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="Form">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="gml:Polygon"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
<xsd:element name="Gebaeude" type="GebaeudeType"
  substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType name="GebaeudeType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="AssNr">
          <xsd:simpleType>
            <xsd:restriction base="xsd:normalizedString">
              <xsd:maxLength value="6"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Flaeche" type="gml:ReferenceType">
          <xsd:annotation>
            <xsd:appinfo>
              <gml:targetElement>BoFlaechen</gml:targetElement>
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:complexType name="BodenbedeckungMemberType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureMemberType">
      <xsd:sequence>
        <xsd:choice>
          <xsd:element ref="BoFlaechen"/>
          <xsd:element ref="Gebaeude"/>
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="Bodenbedeckung" type="BodenbedeckungType"
  substitutionGroup="gml:AbstractFeature"/>

<xsd:complexType name="BodenbedeckungType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="features"
          type="BodenbedeckungMemberType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attributeGroup ref="gml:AggregationAttributeGroup"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
</xsd:schema>

```

## INTERLIS 1-Daten

SCNT

Beispiel Transfer-File

////

MTID Beispiel

MODL Beispiel

TOPI Bodenbedeckung

TABL BoFlaechen\_Form

OBJE 1

STPT 146.92 174.98

LIPT 138.68 187.51

LIPT 147.04 193.00

LIPT 149.79 188.82

LIPT 158.15 194.31

LIPT 163.64 185.96

LIPT 146.92 174.98

ELIN

OBJE 2

STPT 140.69 156.63

LIPT 118.19 179.82

```

LIPT 113.00 219.97
LIPT 148.30 228.97
LIPT 186.38 206.82
ELIN
OBJE 3
STPT 186.38 206.82
ARCP 183.26 188.19
LIPT 170.18 176.00
LIPT 140.69 156.63
ELIN
OBJE 4
STPT 186.38 206.82
LIPT 194.26 208.19
ARCP 190.75 185.21
LIPT 174.10 169.00
LIPT 145.08 149.94
LIPT 140.69 156.63
ELIN
ETAB
TABL BoFlaechen
OBJE 10 0 148.20 183.48
OBJE 20 1 168.27 170.85
OBJE 30 2 133.95 206.06
ETAB
TABL Gebaeude
OBJE 40 958 10
ETAB
ETOP
EMOD
ENDE

```

## INTERLIS 2-Daten

```

<?xml version='1.0' encoding='UTF-8'?>
<TRANSFER xmlns="http://www.interlis.ch/INTERLIS2.3">
<HEADERSECTION SENDER="ceis" VERSION="2.3">
  <MODELS>
    <MODEL NAME="Beispiel" VERSION="2008-03-31"
      URI="mailto:ceis@localhost" />
  </MODELS>
</HEADERSECTION>
<DATASECTION>
<Beispiel.Bodenbedeckung BID="itf0">
<Beispiel.Bodenbedeckung.Gebaeude TID="40">
  <AssNr>958</AssNr><Flaeche REF="10" />
</Beispiel.Bodenbedeckung.Gebaeude>
<Beispiel.Bodenbedeckung.BoFlaechen TID="30">
  <Art>humusiert</Art>
  <Form><SURFACE><BOUNDARY>
    <POLYLINE>

```

```

<COORD><C1>140.69</C1><C2>156.63</C2></COORD>
<COORD><C1>170.18</C1><C2>176.0</C2></COORD>
<ARC><C1>186.38</C1><C2>206.82</C2><A1>183.26</A1><A2>188.19</A2></ARC>
<COORD><C1>148.3</C1><C2>228.97</C2></COORD>
<COORD><C1>113.0</C1><C2>219.97</C2></COORD>
<COORD><C1>118.19</C1><C2>179.82</C2></COORD>
<COORD><C1>140.69</C1><C2>156.63</C2></COORD>
</POLYLINE>
</BOUNDARY>
<BOUNDARY>
  <POLYLINE>
    <COORD><C1>146.92</C1><C2>174.98</C2></COORD>
    <COORD><C1>163.64</C1><C2>185.96</C2></COORD>
    <COORD><C1>158.15</C1><C2>194.31</C2></COORD>
    <COORD><C1>149.79</C1><C2>188.82</C2></COORD>
    <COORD><C1>147.04</C1><C2>193.0</C2></COORD>
    <COORD><C1>138.68</C1><C2>187.51</C2></COORD>
    <COORD><C1>146.92</C1><C2>174.98</C2></COORD>
  </POLYLINE>
</BOUNDARY></SURFACE></Form>
</Beispiel.Bodenbedeckung.BoFlaechen>
<Beispiel.Bodenbedeckung.BoFlaechen TID="20">
  <Art>befestigt</Art>
  <Form><SURFACE><BOUNDARY>
    <POLYLINE>
      <COORD><C1>140.69</C1><C2>156.63</C2></COORD>
      <COORD><C1>145.08</C1><C2>149.94</C2></COORD>
      <COORD><C1>174.1</C1><C2>169.0</C2></COORD>
      <ARC><C1>194.26</C1><C2>208.19</C2><A1>190.75</A1><A2>185.21</A2></ARC>
      <COORD><C1>186.38</C1><C2>206.82</C2></COORD>
      <ARC><C1>170.18</C1><C2>176.0</C2><A1>183.26</A1><A2>188.19</A2></ARC>
      <COORD><C1>140.69</C1><C2>156.63</C2></COORD>
    </POLYLINE>
  </BOUNDARY></SURFACE></Form>
</Beispiel.Bodenbedeckung.BoFlaechen>
<Beispiel.Bodenbedeckung.BoFlaechen TID="10">
  <Art>Gebaeude</Art>
  <Form><SURFACE><BOUNDARY>
    <POLYLINE>
      <COORD><C1>146.92</C1><C2>174.98</C2></COORD>
      <COORD><C1>163.64</C1><C2>185.96</C2></COORD>
      <COORD><C1>158.15</C1><C2>194.31</C2></COORD>
      <COORD><C1>149.79</C1><C2>188.82</C2></COORD>
      <COORD><C1>147.04</C1><C2>193.0</C2></COORD>
      <COORD><C1>138.68</C1><C2>187.51</C2></COORD>
      <COORD><C1>146.92</C1><C2>174.98</C2></COORD>
    </POLYLINE>
  </BOUNDARY></SURFACE></Form>
</Beispiel.Bodenbedeckung.BoFlaechen>

```

```
</Beispiel.Bodenbedeckung>
</DATASECTION>
</TRANSFER>
```

## GML-Daten

```
<?xml version="1.0" encoding="UTF-8"?>
<ili:TRANSFER
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:ili="http://www.interlis.ch/INTERLIS2.3/GML32/INTERLIS"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  gml:id="tid">
  <ili:baskets>
  <ceis:Bodenbedeckung
    xmlns:ceis="http://www.interlis.ch/INTERLIS2.3/GML32/Beispiel"
    gml:id="bid">
  <ceis:features>
  <ceis:Gebaeude gml:id="x40">
  <ceis:AssNr>958</ceis:AssNr>
  <ceis:Flaeche xlink:href="10"/>
  </ceis:Gebaeude>
  </ceis:features>
  <ceis:features>
  <ceis:BoFlaechen gml:id="x30">
  <ceis:Art>humusiert</ceis:Art>
  <ceis:Form>
  <gml:Polygon gml:id="g1" srsName="EPSG:21781" srsDimension="2">
  <gml:exterior>
  <gml:Ring>
  <gml:curveMember>
  <gml:Curve gml:id="g1.1">
  <gml:segments>
  <gml:LineStringSegment interpolation="linear">
  <gml:posList>140.69 156.63 170.18 176</gml:posList>
  </gml:LineStringSegment>
  <gml:Arc numArc="1" interpolation="circularArc3Points">
  <gml:posList>170.18 176.0 183.26 188.19 186.38 206.82</gml:posList>
  </gml:Arc>
  <gml:LineStringSegment interpolation="linear">
  <gml:posList>186.38 206.82 148.3 228.97 113 219.97 118.19 179.82 140.69
  156.63</gml:posList>
  </gml:LineStringSegment>
  </gml:segments>
  </gml:Curve>
  </gml:curveMember>
  </gml:Ring>
  </gml:exterior>
  <gml:interior>
  <gml:LinearRing>
  <gml:pos>146.92 174.98 </gml:pos>
```

```

<gml:pos>163.64 185.96 </gml:pos>
<gml:pos>158.15 194.31 </gml:pos>
<gml:pos>149.79 188.82 </gml:pos>
<gml:pos>147.04 193 </gml:pos>
<gml:pos>138.68 187.51 </gml:pos>
<gml:pos>146.92 174.98</gml:pos>
</gml:LinearRing>
</gml:interior>
</gml:Polygon>
</ceis:Form>
</ceis:BoFlaechen>
</ceis:features>
<ceis:features>
<ceis:BoFlaechen gml:id="x20">
<ceis:Art>befestigt</ceis:Art>
<ceis:Form>
<gml:Polygon gml:id="g20.1" srsName="EPSG:21781" srsDimension="2">
<gml:exterior>
<gml:Ring>
<gml:curveMember>
<gml:Curve gml:id="g20.1.1">
<gml:segments>
<gml:LineStringSegment interpolation="linear">
<gml:posList>140.69 156.63 145.08 149.94 174.1 169.0</gml:posList>
</gml:LineStringSegment>
<gml:Arc numArc="1" interpolation="circularArc3Points">
<gml:posList>174.1 169.0 190.75 185.21 194.26 208.19</gml:posList>
</gml:Arc>
<gml:LineStringSegment interpolation="linear">
<gml:posList>194.26 208.19 186.38 206.82</gml:posList>
</gml:LineStringSegment>
<gml:Arc numArc="1" interpolation="circularArc3Points">
<gml:posList>186.38 206.82 183.26 188.19 170.18 176.0</gml:posList>
</gml:Arc>
<gml:LineStringSegment interpolation="linear">
<gml:posList>170.18 176.0 140.69 156.63</gml:posList>
</gml:LineStringSegment>
</gml:segments>
</gml:Curve>
</gml:curveMember>
</gml:Ring>
</gml:exterior>
</gml:Polygon>
</ceis:Form>
</ceis:BoFlaechen>
</ceis:features>
<ceis:features>
<ceis:BoFlaechen gml:id="x10">
<ceis:Art>Gebaeude</ceis:Art>

```

```
<ceis:Form>
<gml:Polygon gml:id="g3" srsName="EPSG:21781" srsDimension="2">
<gml:exterior>
<gml:LinearRing>
<gml:posList>146.92 174.98 163.64 185.96 158.15 194.31 149.79 188.82 147.04 193 138.68
187.51 146.92 174.98</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Polygon>
</ceis:Form>
</ceis:BoFlaechen>
</ceis:features>
</ceis:Bodenbedeckung>
</ili:baskets>
</ili:TRANSFER>
```