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

1.4 Referenzdokumente

INTERLIS

http://www.interlis.ch/interlis2/docs23/ili2-refman_2006-04-13_d.zip

GML

http://portal.opengeospatial.org/files/?artifact_id=20509
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.

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

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 Grundsätzliches zur Codierung von Objekten XML

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

1.8.1 Objekteigenschaften (Attribute im INTERLIS-Modell)

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

1.8.2 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.

1.8.3 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 Grundsätzlich Unterschiede zu den Codierungsregeln gemäss INTERLIS Referenzhandbuch

1.9.1 Codierungsregeln

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

1.9.2 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.

1.9.3 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.

1.9.4 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.

1.9.5 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).

1.10 Schema-Codierungsregeln

1.10.1 Einleitung

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

1.10.2 Zeichencodierung

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

1.10.3 Allgemeine Regeln

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

1.10.4 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.

1.10.5 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"`
	`VERSION "2008-03-31"  =`
	`  IMPORTS Units;`
	
	`END ModelA.`

GML	`<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">`
	`  <xsd:annotation>`
	`  <xsd:appinfo source="http://www.interlis.ch/ili2c">`
	`  <ili2c:model>ModelA</ili2c:model>`
	`  <ili2c:modelVersion>2008-03-31</ili2c:modelVersion>`
	`  <ili2c:modelAt>mailto:ceis@localhost</ili2c:modelAt>`
	`  </xsd:appinfo>`
	`  </xsd:annotation>`
	`  <xsd:import namespace="http://www.opengis.net/gml/3.2"/>`
	`  <xsd:import namespace`
	`  =" http://www.interlis.ch/INTERLIS2.3/GML32/Units"/>`
	`</xsd:schema>`

1.10.6 Codierung von Wertebereichen

Wertebereichsdefinitionen werden wie folgt bgebildet:

```
  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	`DOMAIN`
	`  LKoord = COORD 480000.00 .. 850000.00 [m] {CHLV03[1]},`
	`  60000.00 .. 320000.00 [m] {CHLV03[2]},`
	`  ROTATION 2 -> 1;`
	`  Horizontbezeichung = TEXT*20;`

GML	`<xsd:complexType name="LKoord">`
	`  <xsd:complexContent>`
	`  <xsd:restriction base="gml:DirectPositionType">`
	`  </xsd:restriction>`
	`  </xsd:complexContent>`
	`</xsd:complexType>`
	`<xsd:simpleType name="Horizontbezeichnung">`
	`  <xsd:restriction base="xsd:normalizedString">`
	`  <xsd:maxLength value="10"/>`
	`  </xsd:restriction>`
	`</xsd:simpleType>`

1.10.7 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"/>
               <xsd:complexType name="%TopicName%Type">
                 <xsd:sequence>
                   <xsd:element name="member"
                       type="%TopicName%MemberType"
                       minOccurs="0" maxOccurs="unbounded"/>
                 </xsd:sequence>
               <xsd:attributeGroup
                   ref="gml:AggregationAttributeGroup"/>
             </xsd:complexType>'.
```

Beispiel

INTERLIS	`TOPIC Bodenbedeckung =`

	CLASS BoFlaechen = END BoFlaechen; CLASS Gebaeude = END Gebaeude; END Bodenbedeckung;
GML	<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"/> <xsd:complexType name="BodenbedeckungType"> <xsd:sequence> <xsd:element name="member" type="BodenbedeckungMemberType" minOccurs="0" maxOccurs="unbounded"/> </xsd:sequence> <xsd:attributeGroup ref="gml:AggregationAttributeGroup"/> </xsd:complexType>

Variante
Behälter zusätzlich als gml:AbstractFeatureType
Variante
Jedes Thema als getrenntes XML-Schema abbilden.

1.10.8 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

```
CLASS Horizont =
    Horizontbezeichnung : MANDATORY TEXT*10;
END Horizont;
```

GML

```
<xsd:element name="Horizont" type="HorizontType"
             substitutionGroup="gml:AbstractFeature"/>
<xsd:complexType  name="HorizontType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="Horizontbezeichnung">
          <xsd:simpleType>
            <xsd:restriction base="xsd:normalizedString">
              <xsd:maxLength value="10"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Variante
STRUCTURE basierend auf gml:AbstractObject

1.10.9 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.

1.10.10 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

1.10.10.1 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

INTERLIS	`CLASS Liegenschaft =`
	`END Liegenschaft;`
	
	`CLASS Person =`
	`END Person;`
	
	`ASSOCIATION =`
	`  Eigentum -- {0..*} Liegenschaft;`
	`  Eigentuemer -- {1} Person;`
	`END;`

GML	`<xsd:element name="Liegenschaft" type="LiegenschaftType"`
	`  substitutionGroup="gml:AbstractFeature"/>`
	`<xsd:complexType  name="LiegenschaftType">`

	<xsd:complexContent> <xsd:extension base="gml:AbstractFeatureType"> <xsd:sequence> <xsd:element name="Eigentuemer" type="gml:ReferenceType"> <xsd:annotation> <xsd:appinfo> <gml:targetElement>Person</gml:targetElement> </xsd:appinfo> </xsd:annotation> </xsd:element> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> <xsd:element name="Person" type="PersonType" substitutionGroup="gml:AbstractFeature"/> <xsd:complexType name="PersonType"> <xsd:complexContent> <xsd:extension base="gml:AbstractFeatureType"> <xsd:sequence> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

1.10.10.2 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

INTERLIS	CLASS Liegenschaft = END Liegenschaft; CLASS Verein = END Verein; ASSOCIATION =

	DauerhafteVereinigung -- {0..*} Verein; Mitglied -- {1..*} Person; END;
GML	<xsd:element name="Verein" type="VereinType" substitutionGroup="gml:AbstractFeature"/> <xsd:complexType name="VereinType"> <xsd:complexContent> <xsd:extension base="gml:AbstractFeatureType"> <xsd:sequence> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> <xsd:element name="Person" type="PersonType" substitutionGroup="gml:AbstractFeature"/> <xsd:complexType name="PersonType"> <xsd:complexContent> <xsd:extension base="gml:AbstractFeatureType"> <xsd:sequence> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType> <xsd:element name="DauerhafteVereinigungMitglied" type="DauerhafteVereinigungMitgliedType" substitutionGroup="gml:AbstractFeature"/> <xsd:complexType name="DauerhafteVereinigungMitgliedType"> <xsd:complexContent> <xsd:extension base="gml:AbstractFeatureType"> <xsd:sequence> <xsd:element name="DauerhafteVereinigung" type="gml:ReferenceType"> <xsd:annotation> <xsd:appinfo> <gml:targetElement>Verein</gml:targetElement> </xsd:appinfo> </xsd:annotation> </xsd:element> <xsd:element name="Mitglied" type="gml:ReferenceType"> <xsd:annotation> <xsd:appinfo> <gml:targetElement>Person</gml:targetElement> </xsd:appinfo> </xsd:annotation> </xsd:element> </xsd:sequence> </xsd:extension> </xsd:complexContent> </xsd:complexType>

1.10.11 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.

1.10.12 Codierung von Attributen

1.10.12.1 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	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;
GML	<xsd:simpleType name="Prozent100"> <xsd:restriction base="xsd:integer"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="100"/> </xsd:restriction> </xsd:simpleType> <xsd:element name="Koernung" type="KoernungType" 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>

1.10.13 Abbildung der INTERLIS-Typen auf XML-Schema-Typen

1.10.13.1 Übersicht

INTERLIS	*Abbildung*	*Complex/Simple-Type*
TextType	xsd:string	Simple
EnumerationType	xsd:string / gml:CodeWithAuthorityType	Simple/Complex
EnumTreeValueType	xsd:string / gml:CodeWithAuthorityType	Simple/Complex
AlignmentType	xsd:string	Simple
BooleanType	xsd:boolean	Simple
NumericType	xsd:integer / xsd:decimal / xsd:double	Simple
FormattedType	xsd:string / xsd:date / xsd:time / xsd:dateTime	Simple
CoordinateType	gml:DirectPositionType	Complex
OIDType	gml:CodeWithAuthorityType	Complex
BlackboxType	xsd:anyType / xsd:base64Binary	Complex/Simple
ClassType	xsd:string	Simple
AttributePathType	xsd:string	Simple
LineType	gml:CurveType / gml:PolygonType	Complex

1.10.13.2 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	DOMAIN
	  TestText = TEXT;
	  TestTextLen = TEXT*10;
	  TestMText = MTEXT;
	  TestMTextLen = MTEXT*10;
	  TestURI = URI;
	  TestName = NAME;

GML	<xsd:simpleType name="TestText"> <xsd:restriction base="xsd:normalizedString"> </xsd:restriction> </xsd:simpleType> <xsd:simpleType name="TestTextLen"> <xsd:restriction base="xsd:normalizedString"> <xsd:maxLength value="10"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType name="TestMText"> <xsd:restriction base="xsd:normalizedString"> </xsd:restriction>

	</xsd:simpleType> <xsd:simpleType name="TestMTextLen"> <xsd:restriction base="xsd:normalizedString"> <xsd:maxLength value="10"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType name="TestURI"> <xsd:restriction base="xsd:anyURI"/> </xsd:simpleType> <xsd:simpleType name="TestName"> <xsd:restriction base="xsd:normalizedString"> <xsd:maxLength value="255"/> <xsd:pattern value="OPEN"/> </xsd:restriction> </xsd:simpleType>

1.10.13.3 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 Aufzahlung abgebildet, wenn der Wertebereich oder das Attribut als FINAL markiert ist. Ansonsten wird die Aufzählung als gml:CodeWithAuthority abgebildet.

Beispiel

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

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

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

Beispiel

INTERLIS
```
DOMAIN
    TestInt = 1 .. 10;
    TestDec = 1.0 .. 10.0;
    TestDouble = 0.123 .. 0.234;
```

GML
```
<xsd:simpleType name=" TestInt">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="10"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="TestDec">
  <xsd:restriction base="xsd:decimal">
    <xsd:minInclusive value="1.0"/>
    <xsd:maxInclusive value="10.0"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="TestDouble">
  <xsd:restriction base="xsd:double">
    <xsd:minInclusive value="0.123"/>
    <xsd:maxInclusive value="0.234"/>
  </xsd:restriction>
</xsd:simpleType>
```

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

1.10.13.5 Codierung von formatierten Wertebereichen
Formatierte Werdebereiche, die INTERLIS.XMLDate, INTERLIS.XMLTime oder INTERLIS.XMLDateTime referenzieren, werden in die entsprechenden XML-Schema Datumstypen (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
```
    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");
GML	<xsd:simpleType name="BuchungsDatum"> <xsd:restriction base="xsd:date"/> </xsd:simpleType> <xsd:simpleType name="StartZeit"> <xsd:restriction base="xsd:time"/> </xsd:simpleType> <xsd:simpleType name="MessZeitpunkt"> <xsd:restriction base="xsd:dateTime"/> </xsd:simpleType> <xsd:simpleType name="EigenesDatum"> <xsd:restriction base="xsd:string"> <xsd:pattern value="OPEN"/> </xsd:restriction> </xsd:simpleType>

1.10.13.6 Codierung von Gefässen

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

Beispiel

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

1.10.13.7 Codierung von Klassentypen

Der Klassentyp wird als xsd:normalizedString codiert.

Beispiel

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

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

1.10.13.8 Codierung von Attributpfadtypen

Der Attributepfadtyp wird als xsd:normalizedString codiert.

Beispiel	
INTERLIS	`DOMAIN` `  InterlisAttributeRef = ATTRIBUTE;`

GML	`<xsd:simpleType name="InterlisAttributeRef">`
	`  <xsd:restriction base="xsd:normalizedString">`
	`  </xsd:restriction>`
	`</xsd:simpleType>`

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

1.10.13.9 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.

1.10.13.10 Codierung von Linienzügen

Linienzüge werden als gml:CurveType 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:CurveType">`
	`  </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 falls im Linienwertebereich nur Geraden zulässig sind

1.10.13.11 Codierung von Einzelflächen und Gebietseinteilungen
Einzelflächen und Gebietseinteilungen werden als gml:PolygonType codiert.

Beispiel	

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

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

1.10.13.12 Codierung von Typen für Objektidentifkation
Typen für Objektidentifikationen werden als xsd:normalizedString abgebildet.

Beispiel	

INTERLIS	`DOMAIN`
	`  BFS_EGID = OID 1 .. 999999999;`
	`  UUID = OID TEXT*36;`

GML	`<xsd:simpleType name="BFS_EGID">`
	`  <xsd:restriction base="xsd:normalizedString">`
	`  </xsd:restriction>`
	`</xsd:simpleType>`
	`<xsd:simpleType name="UUID">`
	`  <xsd:restriction base="xsd:normalizedString">`
	`  </xsd:restriction>`
	`</xsd:simpleType>`

Variante
Als gml:CodeWithAuthorityType
Variante
Als xsd:anyURI

1.10.13.13 Codierung von zusätzlichen Kurvenstück-Formen
OPEN

Anhang A

Vordefiniertes Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.interlis.ch/INTERLIS2.3/GML32/INTERLIS"
    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: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 GebaeudeFlaeche =
      Gebaeude -- {0..*} Gebaeude;
      Flaeche -- {1} BoFlaechen;
    END GebaeudeFlaeche;

  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;
                  Flaeche: -> BoFlaechen // Art = Gebaeude //;
                  IDENT
                  AssNr; !! Annahme AssNr sei eindeutig.
                  Flaeche; !! Dem Gebaeude ist genau eine
                           !! Flaeche 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:PointPropertyType">
    </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:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Form" type="gml:PolygonType">
        </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"/>
<xsd:complexType name="BodenbedeckungType">
  <xsd:sequence>
    <xsd:element name="member" type="BodenbedeckungMemberType"
                        minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gml:AggregationAttributeGroup"/>
</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
OPEN