

# INTERLIS-Erweiterung für lineare und topologische Referenzsysteme

Die kartesischen und ellipsoidischen Koordinatensysteme der verschiedenen Dimensionen sowie die Transformationen zwischen ihnen sind im Datenmodell `CoordSys` im Anhang I des INTERLIS Referenzhandbuchs modelliert.

Datenmodelle und Instanzen für lineare und topologische Koordinatensysteme sowie für die Transformationen zwischen diesen und den bereits modellierten Koordinatensystemen fehlen noch. Entsprechende Vorschläge werden in diesem Dokument vorgestellt.

Es genügt nicht, Koordinatensysteme zu modellieren, sie müssen aus Datenmodellen auch referenziert werden können. Bisher ist es möglich, im `DOMAIN` Bereich eines INTERLIS Datenmodells ein Koordinatenwertebereich zu definieren mit oder ohne Angabe, um welches Koordinatensystem es sich handelt. In beiden Fällen gilt: Wird dieser Koordinatenwertebereich in einem geometrischen Attribut einer Klasse eingesetzt, dann verwenden alle Instanzen dieser Klasse dasselbe Koordinatensystem. Neu soll es nun auch möglich sein, im Wertebereich anzugeben, dass beim Einsatz des Wertebereichs in einer Klasse nicht jede Instanz dieser Klasse dasselbe Koordinatensystem brauchen muss.

Schliesslich sind, entsprechend den Änderungen der konzeptionellen Beschreibungssprache, auch die Regeln für die automatische Herleitung der Transferformatbeschreibung anzupassen.

## 1. Konzeptionelle Sprachelemente für Graphen (Topologie allgemeiner Liniennetze)

Bisher gab es in INTERLIS nur die Möglichkeit, die Topologie der Gebietseinteilung für die Maschine erkennbar präzise zu beschreiben mit Hilfe des Datentyps `AREA`. Es soll neu auch möglich sein, die Topologie von Liniennetzen zu beschreiben, damit z.B. die sog. Fachnetze der Strassenverwaltung konzeptionell modelliert werden können. Auch, damit auf Strassennetzen Ereignisse „grob“ lokalisiert werden können, wozu man sogenannte „topologische“ Bezugssysteme benötigt. Diese wiederum brauchen als Koordinatensystem ein Liniennetz als Referenz.

Wir beschränken uns vorerst auf die Topologie von Liniennetzen. Topologie ist kein Konzept spezifisch nur für eine Fachdomäne, genauso wenig wie Geometrie, sondern eine allgemeine Eigenschaft, ein allgemeiner Aspekt. Zur Beschreibung braucht es Sprachelemente, die unabhängig sind von Fachdomänen. Das mathematische Werkzeug zur Bearbeitung der Topologie von Liniennetzen heisst Graph und besteht aus Knoten und Kanten. Die Kanten sind Knotenpaare. Die Kanten können gerichtet sein oder nicht.

Was sind die Elemente eines Netzes? Das sind Linien, Polylinien z.B., die bestimmte Bedingungen erfüllen müssen: Planar, zusammenhängend, ohne Zyklen etc. Und diese Linien müssen eben nicht unbedingt eine Geometrie

haben, nur einen Anfangsknoten und einen Endknoten, und diese müssen auch nicht unbedingt Koordinaten haben. Also, die Objekte, die eine lineare Topologie bilden, sind einerseits Punkte (oder besser Knoten) und andererseits Punktepaare (oder besser Knotenpaare) mit bestimmten Eigenschaften. Dabei können die linienartigen Objekte mit oder ohne Attribut vom Geometriety POLYLINE sein und die punktförmigen Knotenobjekte können Koordinaten haben oder auch nicht. Zur konzeptionellen Beschreibung von linienförmigen topologischen Strukturen wie Graphen und Netze führen wir ein neues INTERLIS Sprachelement TOPOLOGY ein.

Es folgen zunächst einige Begriffsdefinitionen, die für das Verständnis der vorgeschlagenen neuen INTERLIS Sprachelements wichtig sind (in alphabetischer Reihenfolge):

Baum zusammenhängender Graph ohne Zyklen. Syn: tree (en)

Graph besteht aus Knoten und (nicht orientierten) Kanten, wobei zu jeder Kante ein (ungeordnetes) Paar von Knoten gehört, nämlich ihre beiden Endknoten

Kante Def siehe Graph. Syn edge (en)

Knoten Def siehe Graph. Syn node (en)

Netz Syn von zusammenhängender Graph

planarer Graph Graph, dessen Knoten sich als Punkte und dessen Kanten sich als Verbindungskurven in der Ebene darstellen lassen, wobei je zwei verschiedene solche Kantenkurven höchstens Endpunkte gemeinsam haben. Syn: planar graph (en).

Schlaufe Zyklus mit einer Kante. Syn: loop (en)

Weg endliche Folge von paarweise verschiedenen Knoten  $v_i$  und (daher auch paarweise verschiedenen) Kanten  $k_i$  der Art  $(v_1, k_1, v_2, k_2, \dots, v_{n-1}, k_{n-1}, v_n)$ , wobei  $(v_i, v_{i+1})$  das zur Kante  $k_i$  gehörige Paar von Knoten ist. Syn: path (en).

zusammenhängender Graph zu je zwei verschiedenen Knoten des Graphs gibt es einen Weg, dessen Endknoten sie sind, d.h. je zwei verschiedene Knoten des Graphs sind verbunden durch einen Weg. Syn: Netz (de), connected graph, net (en)

Zyklus geschlossener Weg, d.h. Weg mit  $v_1 = v_n$ .

Die Syntax der Topologie ist die folgende:

```
TopologyDef = 'TOPOLOGY' Name Properties<NOCYCLES,NET,TREE,PLANAR> '='
  ['NODES' [CoordType-DomainRef] ['ATTRIBUTES' StructureRef]
    ['DERIVED' (Explanation
      | ('FROM' ViewableRef [GeomAttr-Name] ';' ) )]]
  ['EDGES' [LineType-DomainRef] ['ATTRIBUTES' StructureRef]
    ['DERIVED' (Explanation
      | ('FROM' ViewableRef [GeomAttr-Name] ';'
        | ('LINKED' 'BY' AssociationRef ';' ) ) )]]
'END' Name ';'.
```

Die Werte von Properties bedeuten, dass der Graph die folgenden Eigenschaften hat:

- NOCYCLES: Keine Zyklen,
- NET: Zusammenhängend,

- TREE: Keine Zyklen und zusammenhängend.
- PLANAR: es handelt sich um einen planaren Graph.

Kommt keiner der Properties vor, dann handelt es sich um die Kanten eines allgemeinen Graphs, der Zyklen enthalten kann und weder zusammenhängend noch planar zu sein braucht.

Dieselben Fachdaten bilden evtl. verschiedene Netztopologien. Die folgenden Beispiele zeigen verschiedene Varianten, um Netztopologie zu definieren am Beispiel von Strassen und Kreuzungen:

Variante 1

```

TOPIC Variante1 =
CLASS Kreuzung =
END Kreuzung;
CLASS Strasse =
END Strasse;

ASSOCIATION sk =
  s -- {1..*} Strasse;
  k -- {2} Kreuzung;
END sk;

TOPOLOGY Netz =
  NODES DERIVED FROM Kreuzung;
  EDGES DERIVED FROM Strasse;
  LINKED BY sk;
END Netz;

END Variante1;

```

Variante 2: Knoten sind eine Klasse, Kanten sind die Assoziation.

```

TOPIC Variante2 =
CLASS Kreuzung =
END Kreuzung;

ASSOCIATION Strasse=
  a -- {1} Kreuzung;
  e -- {1} Kreuzung;
END;

TOPOLOGY Netz =
  NODES DERIVED FROM Kreuzung;
  EDGES DERIVED FROM Strasse;
END Netz;

END Variante2;

```

## 2. Definition von Instanzen im Modell

Damit Koordinatensysteme vollständig als Teil des Modells definiert werden können, wird eine Syntax für die Definition von Objekten definiert.

```

Instance = '{' { AttributeValue | TechValue } '}'.
TechValue = '*' Attribute-Name ':' Constant.';'.
AttributeValue = Attribute-Name ':' (
    Value
    | ( '[' Value { ',' Value } ']' ) ';' '.
Value = Constant | ([ ClassRef ] Instance ).

```

Beispiel:

```

01: REFSYSTEM BASKET LinRefSys ~ INTERLIS.RefSys
02:   OBJECTS OF INTERLIS.GeoCurvilinear2D {
03:     Name : "RBBS2";
04:     Axis : [
05:       INTERLIS.AlongAXIS {
06:         },
07:       INTERLIS.LateralAXIS {
08:         Direction: #right;
09:       }
10:     ];
11:     CurveClassRef: "Nationalstrassen.Strassenachse.Strasse";
12:     PolylineAttributePath: "Achsgemetrie";
13:   };

```

Auf Zeile 03 wird der Wert für das Attribut Name definiert.

Auf den Zeilen 04-10 werden mehrere Werte für das Attribut Axis definiert.

Auf den Zeilen 07-09 wird ein Strukturelement der Struktur INTERLIS.LateralAXIS definiert.

Auf Zeile 08 wird der Wert für das Attribut Direction definiert.

Auf Zeile 11 wird der Wert für das Attribut CurveClassRef definiert.

Auf Zeile 12 wird der Wert für das Attribut PolylineAttributePath definiert.

### 3. Konzeptionelle Sprachelemente für Bezugssysteme

Die konzeptionelle Datenbeschreibungssprache muss Sprachelemente zur Verfügung stellen, damit man einerseits die Bezugssysteme präzise beschreiben kann und andererseits diese Bezugssysteme referenzieren kann. Dabei ist der Referenzbereich eines Bezugssystems meistens implizit durch das Koordinatensystem des Bezugssystems gegeben und der Referenzrahmen muss durch Messungen und bauliche Massnahmen in der Realwelt realisiert werden. Bei der konzeptionellen Modellierung können wir uns daher auf die Koordinatensysteme beschränken.

#### 3.1. Modellierung kartesischer und ellipsoidischer Koordinatensysteme

Ein Koordinatensystem ist von den Daten aus gesehen definiert durch seine Achsen. Diese können Geraden sein oder Kreis- bzw. Ellipsen-Bogen oder einfache Linienzüge entsprechend der Art des Referenzbereiches, den sie auszumessen erlauben. Für Geraden als Achsen steht die Struktur

**LengthAXIS** zur Verfügung, für Kreis- bzw. Ellipsen-Bogen als Achsen die Struktur **AngleAXIS**. Aus dem Datenmodell **CoordSys** im Anhang I des Referenzhandbuchs zeigen wir hier nur die INTERLIS Beschreibung der **LengthAXIS** sowie deren Einsatz in der Beschreibung der Klasse **GeoCartesian3D**, der 3-dimensionalen kartesischen Koordinatensysteme. Für Koordinatensysteme mit einem einfachen Linienzug oder Kanten eines Graphen als erste Achse werden neue Achstypen eingeführt. Definitionen aus dem internen Modell **INTERLIS**:

```

CLASS METAOBJECT (ABSTRACT) =
  Name: MANDATORY NAME;
  UNIQUE Name;
END METAOBJECT;

STRUCTURE AXIS =
  PARAMETER
  Unit: NUMERIC [ANYUNIT];
END AXIS;

CLASS REFSYSTEM (ABSTRACT) EXTENDS METAOBJECT =
END REFSYSTEM;

CLASS COORDSYSTEM (ABSTRACT) EXTENDS REFSYSTEM =
  Axis: LIST {1..3} OF AXIS;
END COORDSYSTEM;

```

Definitionen aus dem Datenmodell **CoordSys**:

```

STRUCTURE LengthAXIS EXTENDS INTERLIS.AXIS=
  ShortName: TEXT*12;
  Description: TEXT*255;
  PARAMETER
  Unit (EXTENDED): NUMERIC [INTERLIS.LENGTH];
END LengthAXIS;

CLASS GeoCartesian3D EXTENDS INTERLIS.COORDSYSTEM=
  Definition: TEXT*70;
  Axis (EXTENDED): LIST {3} OF LengthAXIS;
END GeoCartesian3D;

```

### 3.2. Modellierung kurvilinearere Koordinatensysteme

Für das konzeptionelle Datenmodell kurvilinearere Koordinatensysteme brauchen wir die beiden neuen Achsentyp **AlongAXIS** für Achsen, die dem kurvilinearen Element (Polylinie oder Kante) entlang laufen, sowie **LateralAXIS** für die ganze Menge von Geraden, die in jedem Punkt des kurvilinearen Elements auf diesem senkrecht stehen und im Referenzbereich liegen. Ihre konzeptionellen Datenmodelle sowie dasjenige der kurvilineareren Koordinatensysteme lauten:

```

STRUCTURE AlongAXIS EXTENDS INTERLIS.AXIS=
  ShortName: TEXT*12;

```

```

    Description: TEXT*255;
PARAMETER
    Unit (EXTENDED): NUMERIC [INTERLIS.LENGTH];
END AlongAXIS;

STRUCTURE LateralAXIS =
    ShortName: TEXT*12;
    Description: TEXT*255;
    Direction: MANDATORY (left, right, up, down);
PARAMETER
    Unit: NUMERIC [INTERLIS.LENGTH];
END LateralAXIS;

CLASS GeoCurviline2D EXTENDS INTERLIS.COORDSYSTEM=
    CurveClassRef: TEXT;
    PolylineAttributePath: TEXT;
    !! Axis[1] instanceof AlongAXIS
    !! Axis[2] instanceof LateralAXIS
END GeoCurviline2D;

CLASS GeoCurviline3D EXTENDS INTERLIS.COORDSYSTEM=
    CurveClassRef: TEXT;
    PolylineAttributePath: TEXT;
    !! Axis[1] instanceof AlongAXIS
    !! Axis[2] instanceof LateralAXIS
    !! Axis[3] instanceof LateralAXIS
END GeoCurviline3D;

```

Diese Klassen sind im internen Modell INTERLIS definiert, weil sie eine spezielle Kodierung der COORD Werte zur Folge haben. CurveClassRef ist der qualifizierte Name der Klasse, die die POLYLINE enthält (z.B. "Nationalstrassen.Strassenachse.Strasse"). PolylineAttributePath ist der Pfad auf das Attribut, das die Geometrie des kurvilinearen Elements darstellt (z.B. "Achsegeometrie"). Beispiel ohne Strukturattribute:

```

MODEL Nationalstrassen ... =
    TOPIC Strasseachse =
        CLASS Strasse =
            Achsegeometrie : DIRECTED POLYLINE ...;
        END Strasse;
    END Strasseachse;
END Nationalstrassen.

```

Ist das Geometrieattribut in einem Strukturelement, kann ein Attributpfad verwendet werden. Um irgendein Strukturelement zu ermöglichen, kann das Schlüsselwort ANY als Indexwert verwendet werden (z.B. "Achsegeometrie[ANY].teilgeometrie").

```

MODEL Nationalstrassen ... =
    TOPIC Strasseachse =
        STRUCTURE Strassenabschnitt =
            teilgeometrie : DIRECTED POLYLINE ...;
        END Strassenabschnitt;
        CLASS Strasse =

```

```

    Achsgeometrie : LIST {1..*} OF Strassenabschnitt;
  END Strasse;
  END Strasseachse;
END Nationalstrassen.

```

### 3.3. Modellierung topologischer Koordinatensysteme

Für das konzeptionelle Datenmodell topologischer Koordinatensysteme können wir für die Achsdefinitionen AlongAXIS und LateralAXIS übernehmen. Das konzeptionelle Datenmodell des topologischen Koordinatensystems lautet:

```

CLASS GeoTopological2D EXTENDS INTERLIS.COORDSYSTEM=
  TopologyRef: TEXT;
  !! Axis[1] instanceof AlongAXIS
  !! Axis[2] instanceof LateralAXIS
END GeoTopological2D;

CLASS GeoTopological3D EXTENDS INTERLIS.COORDSYSTEM=
  TopologyRef: TEXT;
  !! Axis[1] instanceof AlongAXIS
  !! Axis[2] instanceof LateralAXIS
  !! Axis[3] instanceof LateralAXIS
END GeoTopological3D;

```

TopologyRef ist der qualifizierte Name der Topologie-Definition, die die Kanten beschreibt (z.B. "Nationalstrassen.Strassen.Netz").

```

MODEL Nationalstrassen ... =
  TOPIC Strassen =
    CLASS Kreuzung =
      END Kreuzung;
    CLASS Strasse =
      END Strasse;

    ASSOCIATION sk =
      s -- {1..*} Strasse;
      k -- {2} Kreuzung;
    END sk;

    TOPOLOGY Netz =
      NODES DERIVED FROM Kreuzung;
      EDGES DERIVED FROM Strasse;
      LINKED BY sk;
    END Netz;
  END Strassen;
END Nationalstrassen.

```

### 3.4. Referenzierung von Koordinatensystemen

Bisher ist es möglich, im **DOMAIN** Bereich eines INTERLIS Datenmodells ein Koordinatenwertebereich zu definieren mit der Angabe, welches Koordinatensystem gilt:

```

DOMAIN
HKoord = COORD
  480000.000 .. 850000.000 [m] {CHLV03[1]},
  70000.000 .. 310000.000 [m] {CHLV03[2]},
  -200.000 .. 5000.000 [m] {SwissOrthometricAlt[1]},
ROTATION 2 -> 1;

```

Die Angabe, um welches Koordinatensystem es sich handelt, kann im Datenmodell weggelassen werden, ist aber in Prosa zum Modell zu dokumentieren.

```

DOMAIN
HKoord = COORD
  480000.000 .. 850000.000 [m],
  70000.000 .. 310000.000 [m],
  -200.000 .. 5000.000 [m],
ROTATION 2 -> 1;

```

Neu soll bei der Definition eines Koordinatenwertebereichs im **DOMAIN** Abschnitt auch angegeben werden können, dass pro Instanz ein unterschiedliches Koordinatensystem zulässig ist (mit dem Schlüsselwort **ANY**):

```

DOMAIN
HKoord = COORD
  480000.000 .. 850000.000 [m] {ANY},
  70000.000 .. 310000.000 [m] {ANY},
  -200.000 .. 5000.000 [m] {SwissOrthometricAlt[1]},
ROTATION 2 -> 1;

```

Der Verweis auf ein kurvilineares Koordinatensystem funktioniert identisch. Aber das Koordinatensystem-Objekt muss eine Instanz der Klasse **GeoCurviline2D** oder **GeoCurviline3D** oder einer Erweiterung davon sein.

```

REFSYSTEM BASKET LinRefSys ~ INTERLIS.RefSys
OBJECTS OF INTERLIS.GeoCurviline2D {
  Name : "RBBSD2";
  Axis : [
    INTERLIS.AlongAXIS {
    },
    INTERLIS.LateralAXIS {
      Direction: #right;
    }
  ];
  CurveClassRef: "Nationalstrassen.Strassenachse.Strasse";
  PolylineAttributePath: "Achsgeometrie";
};

```

```

DOMAIN
Rbbs2dKoord = COORD
  0.000 .. 10000.000 [m] {RBBS2D[1]},
  0.000 .. 500.000 [m] {RBBS2D[2]},
ROTATION 1 -> 2;

```

Der Verweis auf ein topologisches Koordinatensystem funktioniert identisch. Aber das Koordinatensystem-Objekt muss eine Instanz der Klasse `GeoTopological2D` oder `GeoTopological3D` oder einer Erweiterung davon sein.

```
REFSYSTEM BASKET TopoRefSys ~ INTERLIS.RefSys
  OBJECTS OF INTERLIS.GeoTopological2D {
    Name : "GDF2D";
    Axis : [
      INTERLIS.AlongAXIS {
      },
      INTERLIS.LateralAXIS {
        Direction: #right;
      }
    ];
    TopologyRef: "Nationalstrassen.Strassen.Netz";
  };

DOMAIN
  Gdf2dKoord = COORD
    0.000 .. 10000.000 [m] {GDF2D[1]},
    0.000 .. 500.000 [m] {GDF2D[2]},
  ROTATION 1 -> 2;
```

Die Syntaxregeln werden wie folgt geändert:

```
RefSys = ( '{' (RefSys-MetaObjectRef [ '[' Axis-PosNumber ']' ])
           / 'ANY' '}'
           | '<' Coord-DomainRef [ '[' Axis-PosNumber ']' ] '>' ).

MetaDataBasketDef = ( 'SIGN' | 'REFSYSTEM' ) 'BASKET' Basket-Name
  Properties<FINAL>
  [ 'EXTENDS' MetaDataBasketRef ]
  '~' TopicRef
  { 'OBJECTS' 'OF' ClassRef
    ( ':' MetaObject-Name { ',' MetaObject-Name } )
    / (MetaObject-Instance { ',' MetaObject-Instance } )
  } ';'.
```

## 4. Kodierungsregeln

Die folgenden Kapitel sind erst Ideen.

### 4.1. Erweiterung der Regeln für Koordinatenwertebereiche

Bis jetzt gab es keine Angabe des Koordinatensystems im Transfer, da alle Instanzen eines Attributs einer Klasse dasselbe Koordinatensystem haben.

```
<COORD><C1>Wert</C1><C2>Wert</C2>...</COORD>
```

Neu soll es nun aber möglich sein, dass verschiedene Instanzen desselben Attributs (derselben Klasse) verschiedene Koordinatensysteme brauchen. Man

muss daher das Koordinatensystem im Transfer angeben können. Die Kodierung erfolgt als XML-Attribut (ähnlich zur Kodierung der übrigen Referenzen) und der Verweis selbst als URI, sodass weitere Arten von Verweisen auf Koordinatensysteme ohne weitere Änderung der Kodierung definiert werden können.

```
<COORD CRS="{URI}">
  <C1>Wert</C1><C2>Wert</C2>...
</COORD>
```

Die Abbildung einer OID (Objekt mit der Liniengeometrie) oder einer BID+Name (CRS gem. CoordSys) auf eine URI muss noch definiert werden. Oder nur OID, wir können immer noch etwas raffinierteres erfinden.

Für TopoRefSys muss definiert werden wie die Topologienetzwerk und die Kante referenziert und die Kantenrichtung definiert werden kann. Ref mit OID oder Name. Kantenrichtung mir noch unklar. Für gerichtete Kanten gibt es im Prinzip zwei Möglichkeiten:

- Entweder zwei Kanten mit vertauschtem Anfangs- und Endknoten für den Fall der gerichteten mit beiden Richtungen vorkommenden Kante und eine Kante mit Definition der Richtung durch Wahl von Anfangs- und Endknoten im Falle einfacher Richtung.
- Oder: Immer nur eine Kante und Aufzählattribut mit den Werten (hin,gegen,beide) oder ähnlich.

## 4.2. Zusätzliche Regeln für Graphen

### 4.2.1. Minimum Daten

```
<MeinModel.MeinTopic.MeinNetzwerk TID="0">
  <NODE TID="1">
  </NODE>
  <NODE TID="2">
  </NODE>
  <EDGE TID="3">
    <NODE1 REF="1"/>
    <NODE2 REF="2"/>
  </EDGE>
</MeinModel.MeinTopic.MeinNetzwerk>
```

### 4.2.2. Daten haben optional Verweis auf Basisobjekte

```
<MeinModel.MeinTopic.Kreuzung TID="100">
</MeinModel.MeinTopic.Kreuzung>
<MeinModel.MeinTopic.Strasse TID="200">
</MeinModel.MeinTopic.Strasse>
<MeinModel.MeinTopic.MeinNetzwerk TID="0">
  <NODE TID="1">
```

```

    <BASE REF="100"/>
  </NODE>
  <NODE TID="2">
  </NODE>
  <EDGE TID="3">
    <NODE1 REF="1"/>
    <NODE2 REF="2"/>
    <BASE REF="200"/>
  </EDGE>
</MeinModel.MeinTopic.MeinNetzwerk>

```

#### 4.2.3. Daten haben optional Geometrie

```

<MeinModel.MeinTopic.MeinNetzwerk TID="0">
  <NODE TID="1">
    <COORD>...</COORD>
  </NODE>
  <NODE TID="2">
  </NODE>
  <EDGE TID="3">
    <NODE1 REF="1"/>
    <NODE2 REF="2"/>
    <POLYLINE>...</POLYLINE>
  </EDGE>
</MeinModel.MeinTopic.MeinNetzwerk>

```

#### 4.2.4. Daten haben optional weitere Facheigenschaften

```

<MeinModel.MeinTopic.MeinNetzwerk TID="0">
  <NODE TID="1">
    <USER>
      <MeinModel.MeinTopic.MeineStruktur>
        ...
      </MeinModel.MeinTopic.MeineStruktur>
    </USER>
  </NODE>
  <NODE TID="2">
  </NODE>
  <EDGE TID="3">
    <NODE1 REF="1"/>
    <NODE2 REF="2"/>
    <USER>
      <MeinModel.MeinTopic.MeineStruktur>
        ...
      </MeinModel.MeinTopic.MeineStruktur>
    </USER>
  </EDGE>
</MeinModel.MeinTopic.MeinNetzwerk>

```