



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für Verteidigung,
Bevölkerungsschutz und Sport VBS

Bundesamt für Landestopografie swisstopo

Benutzerhandbuch *ili2fme*

Erstellung: Mai 2015

Auftraggeber:
KOGIS, Bundesamt für Landestopografie swisstopo

Autor:



Chemin de Maillefer 36
1052 Le Mont-sur-Lausanne
Tel.: 021 643 77 11
Fax: 021 643 77 10
info@inser.ch

Inhaltsverzeichnis

QUELLEN, DOKUMENTE	4
1. EINFÜHRUNG	5
2. UNABDINGBARE THEORETISCHE VORAUSSETZUNGEN.....	6
2.1 INTERLIS	6
2.2 FME.....	9
3. ALLGEMEINE INFORMATIONEN - <i>ILI2FME</i>	12
3.1 Software-/Hardwarevoraussetzungen.....	12
3.2 Versionen	12
3.3 Interlis-Modell und Transferdateien	12
3.4 Beschreibung der Parameter.....	13
3.5 Codierung der Geometrie.....	14
3.6 Formatattribute.....	15
4. SCHNELLSTART	16
4.1 Installation	16
4.2 Schnelle Konvertierung	16
4.3 Schnelle Visualisierung	17
5. ANWENDUNGSFÄLLE.....	18
5.1 Schreiben einer INTERLIS 2-Datei.....	18
5.2 Lesen einer INTERLIS 2-Datei	28
5.3 Lesen und Schreiben von INTERLIS 2-Dateien mit geschachtelten Strukturen	31
5.4 Konvertierung INTERLIS ↔ Relationale Datenbanken	41
6. FAQ.....	49

QUELLEN, DOKUMENTE

- [1] KOGIS, INTERLIS 2 – Referenzhandbuch, 2006-04-13
http://www.interlis.ch/interlis2/docs23/ili2-refman_2006-04-13_d.pdf
- [2] KOGIS, Allgemeine Empfehlungen zur Methode der Definition «minimaler Geodatenmodelle», 2011-09-12
<http://www.geo.admin.ch/internet/geoportal/de/home/topics/geobasedata/models.html>
- [3] FME Transformer Reference Guide 2015, SAFE Software
<http://cdn.safe.com/resources/fme/FME-Transformer-Reference-Guide.pdf>
- [4] Eisenhut Informatik AG, INTERLIS 2 Reader/Writer for FME
<http://www.eisenhutinformatik.ch/interlis/ili2fme/interlis2-20140313.pdf>
- [5] swisstopo, Best-Practice Beispiele für die Umsetzung von konzeptionellen Geodatenmodellen, 2014
<http://www.geo.admin.ch/internet/geoportal/de/home/topics/geobasedata/models.html>
- [6] ArcGIS help 10.2, 10.2.1 and 10.2.2
<http://resources.arcgis.com/en/help/main/10.2/index.html#//004m00000003000000>
- [7] Eisenhut Informatik AG, Anwenderforum *ili2fme*
<http://www.ili2fme.ch/>
- [8] Modellablage zur Speicherung der minimalen Geodatenmodelle des Bundes
<http://models.geo.admin.ch/>
- [9] INTERLIS-Werkzeuge
http://www.interlis.ch/interlis2/download23_d.php#outils
- [10] swisstopo, Basismodule des Bundes für minimale Geodatenmodelle, 2011-08-30
<http://www.geo.admin.ch/internet/geoportal/de/home/topics/geobasedata/models.html>

1. EINFÜHRUNG

Dieses Dokument dient dem Zweck, die Nutzung des Plugin *ili2fme* in der Software FME zu erläutern und zu illustrieren. Im ersten Teil findet der Leser theoretische und fachtechnische Begrifflichkeiten, die zur Visualisierung, Modifizierung und Erstellung von INTERLIS-Dateien hilfreich oder sogar unverzichtbar sind. Im zweiten Teil werden - anhand von vier FME Workspaces - praktische Anwendungsfälle illustriert. Die beiden ersten Beispiele sind relativ einfach und dienen dazu, den Nutzer mit den Grundsätzen der Modellierung von INTERLIS-Dateien in FME vertraut zu machen. Sie enthalten eine vollständige Beschreibung der Vorgehensweise. In den beiden folgenden Beispielen handelt es sich um etwas kompliziertere konkrete Anwendungsfälle, die in diesem Dokument in ihren allgemeinen Grundsätzen dargelegt, jedoch weniger detailliert beschrieben werden, da weitere Besonderheiten der INTERLIS-Modelle in FME aufgezeigt werden sollen. Abschliessend wird auf bestimmte Probleme in Form von Fragen und Antworten eingegangen.

Für das Verständnis dieses Dokuments ist es ratsam, über das Programm FME (mindestens Version 2014) zu verfügen, um während des Lesens die jeweils beschriebenen Arbeitsschritte ausführen zu können.

2. UNABDINGBARE THEORETISCHE VORAUSSETZUNGEN

In diesem Kapitel werden einige theoretische Bausteine beschrieben, die für das Verständnis der weiteren Abschnitte des Berichts unverzichtbar sind.

2.1 INTERLIS

Alle neu eingerichteten minimalen Geodatenmodelle werden in INTERLIS 2 erstellt. Daher wird das Format INTERLIS 1 in diesem Bericht nicht behandelt, obwohl es von dem Plugin *ili2fme* auch unterstützt wird.

2.1.1 Allgemeines

INTERLIS ist das Schnittstellenformat für den Austausch der Geodaten des Bundes. Es wurde definiert, um den Datenaustausch zwischen mehreren Nutzern von GIS-Werkzeugen nach einem vorgegebenen Datenmodell zu erleichtern. Es handelt sich um ein offenes Format, das sich auf Daten textlicher Art stützt.

Ein INTERLIS-Datensatz setzt sich aus zwei Dateien zusammen:

- das Datenmodell .ILI
enthält eine Beschreibung des Datenschemas;
- die Daten .ITF (INTERLIS 1) / .XTF (INTERLIS 2)
enthalten die Daten selbst, die nach dem entsprechenden Modell strukturiert sind;
- die Datenkataloge (XML/XTF)
enthält Wertelisten sowie gegebenenfalls deren Bedeutungen in verschiedenen Sprachen.

2.1.2 Modelle, Themen, Klassen

Die Hierarchie des konzeptionellen Modells INTERLIS 2 besteht aus 3 Ebenen:

- das Modell («MODEL») enthält sämtliche Bestandteile der zu beschreibenden Realität. Es ist insbesondere durch einen eindeutigen Namen und eine Version gekennzeichnet;
- das Thema («TOPIC») enthält die Definitionen, die zur Beschreibung eines präzisen objektiven Teils der Realität dienen;
- die dritte Ebene besteht aus einer Reihe von Elementen, die den Kern des konzeptionellen Schemas definieren, wobei die nachfolgend genannten am häufigsten verwendet werden:
 - Klassen («CLASS»): ermöglichen es, die Eigenschaften aller zu ihr gehörenden Objekte zu deklarieren, insbesondere den Namen und den Typ von Attributen zu definieren;
 - Wertebereiche («DOMAIN»): werden genutzt, um den Wert eines Attributs entsprechend einer Liste oder einer bestimmten Bereichsgrösse zu erzwingen;
 - Strukturen («STRUCTURE»): bestehen aus einer Liste von Attributen (Namen und Typen), wobei die Struktur anschliessend einem oder mehreren zu einer Klasse desselben Modells gehörenden Attribut(en) zugewiesen werden kann;
 - Assoziationsklassen («ASSOCIATION»): ermöglichen die Definition der Beziehungen, die zwischen verschiedenen Klassen eines Themas existieren können; in INTERLIS lassen sich zwei Arten von Beziehungen definieren: eingebettete und nicht eingebettete (siehe [1], Kap. 3.3.9).

Quelle: INTERLIS 2 - Referenzhandbuch [1], Kap. 1.3, 1.4, 2.5

2.1.3 Schlüsselwörter und Definitionen

Die Bedeutung der folgenden reservierten Wörter muss bekannt sein, da sie in den Beispielen des Kapitels 5 verwendet werden.

- **IMPORTS**: falls ein Element sich auf eine Definition bezieht, die in einem anderen Modell vorgenommen wurde, so ist letzteres mit diesem Befehl zu importieren.

```
IMPORTS LocalisationCH_V1,GeometryCHLV03_V1;
```

- **EXTENDS**: die Vererbung ermöglicht es, einem Objekt Eigenschaften zuzuweisen, die zuvor einem anderen Objekt zugewiesen wurden. Beispielsweise kann eine Klasse Attribute erben, die einer anderen Klasse zugewiesen wurden.

```
CLASS HM_KE_Catalogue  
  EXTENDS CatalogueObjects_V1.Catalogues.Item =
```

- **LIST OF / BAG OF**: werden verwendet, wenn die Werte eines Strukturattributs mehrere Strukturelemente umfassen, beziehungsweise geordnet oder ungeordnet sein können.

```
STRUCTURE LocalisedText =  
  Language: LanguageCode_ISO639_1;  
  Text: MANDATORY TEXT;  
END LocalisedText;  
  
STRUCTURE MultilingualText =  
  LocalisedText : BAG {1..*} OF LocalisedText;  
  UNIQUE (LOCAL)  
  LocalisedText:Language;  
END MultilingualText;  
  
CLASS A =  
  Attribut 1 : LocalisationCH_V1.MultilingualText;  
END A;
```

In diesem Beispiel kann das Attribut *Attribut 1* der Klasse A zwischen 1 und n Werte enthalten, und zwar mit der in *LocalisedText* definierten Struktur.

Quelle: INTERLIS 2 - Referenzhandbuch [1], Kap. 2.1, 2.2, 2.4, 2.6

2.1.4 CHBase

Im Rahmen der Abstimmung zur Einrichtung der minimalen Geodatenmodelle (MGDM) hat der Bund Basismodule definiert, die sich auf allgemeine Aspekte der Modellierung beziehen. Diese Elemente können bei Bedarf in ein MGDM importiert werden.

```
MODEL MyModel [...] =  
  IMPORTS CHBaseModule;  
  MyClass =  
    MyAttr : CHBaseModule.CHBaseDomain;  
  END MYClass;  
END MyModel.
```

Der vorstehende Auszug zeigt, in welcher Weise die Basismodule in die Definition eines Modells einfließen: der vordefinierte Bereich *CHBaseDomain*, der sich in dem Modell *CHBaseModule* befindet, wird als Eigenschaft des Attributs *MyAttr* verwendet.

2.1.4.1 Beispiel

In dem folgenden Beispiel finden sich drei Bezüge zu CHBase-Objekten:

- das Attribut *Mutationsgrund_Text* ist ein mehrsprachiges Textobjekt;
- das Attribut *Kanton* ist mit einem Wertebereich verbunden, der die formalisierte Liste aller Kantone enthält;
- die Geometrie ist vom Typ mehrteilige Fläche.

```
CLASS Flaechenobjekt=  
  ObjNummer : MANDATORY TEXT;  
  ObjName : MANDATORY TEXT*30;  
  Mutationsdatum : MANDATORY INTERLIS.XMLDate;  
  Mutationsgrund_Text : MANDATORY LocalisationCH_V1.MultilingualText;  
  Kanton: MANDATORY CHAdminCodes_V1.CHCantonCode;  
  Geometrie : MANDATORY GeometryCHLV03_V1.MultiSurface;  
END Flaechenobjekt;
```

2.1.4.2 Hauptobjekte

Generell werden folgende CHBase-Objekte häufig verwendet:

- *MultiSurface*: mehrteilige Geometrie
- *MultilingualText*: mehrsprachiges Textobjekt
- *Codelisten*: Wertebereich, der eine Reihe von Codes und deren Bedeutungen enthält
- *ModInfo*: Historie der an den Daten vorgenommenen Änderungen

Diese minimalen Modelle sind über das Webportal von swisstopo unter folgender Adresse zugänglich:

<http://models.geo.admin.ch/CH/> [8]

Quelle:

- *Allgemeine Empfehlungen zur Methodik der Definition «minimaler Geodatenmodelle»* [2], Kap. 2 und 3
- *Basismodule des Bundes für minimale Geodatenmodelle* [10]

2.1.5 Mehrteilige versus multiple (mehrfache) Geometrie

Um eventuelle Verwechslungen zwischen einer sogenannten «*mehrteiligen*» Geometrie und multiplen (mehrfachen) Geometrien zu vermeiden, ist es wichtig, diese beiden Begriffe zu unterscheiden.

Von einer mehrteiligen Geometrie spricht man bei einem Objekt, das mehrere Geometrien identischen Typs enthalten kann. Wie wir im Beispiel 5.1 sehen werden, kann ein Kanton aus mehreren Polygonen bestehen. Dies ist insbesondere der Fall beim Kanton Waadt, der sich aus einem grossen Polygon und einer Enklave im Kanton Freiburg zusammensetzt. In der INTERLIS-Datendatei sind diese beiden Geometrien im gleichen Attribut gespeichert.

Die Klasse des Modells wird also eine einzige Linie enthalten.

```
CLASS cantonClass =  
    ...  
    position : GeometryCHLV03_V1.MultiSurface;  
    ...  
END cantonClass;
```

Von einer Klasse mit multipler (mehrfacher) Geometrie spricht man, wenn in der gleichen Klasse mehrere unterschiedliche Geometrien gespeichert werden. Beispielsweise kann eine Klasse Lawine eine Fläche zur Definition der Anrisszone, eine zweite Fläche für die Fliesszone und eine letzte Fläche für ihre Auslaufzone enthalten. Ausserdem lässt sich die Bruchkante mit Hilfe einer Geometrie vom Typ Linie definieren.

```
CLASS avalanche =  
    ...  
    zone_de_depart : Polygon;  
    zone_d_ecoulement : Polygon;  
    zone_d_arret : Polygon;  
    ligne_de_rupture : Polygon;  
    ...  
END avalanche;
```

Im Gegensatz zu einer mehrteiligen Geometrie hat somit jede multiple Geometrie eine eigene Bedeutung. Hinweis: Eine Klasse kann eine multiple Geometrie haben, die aus einer mehrteiligen Geometrie zusammengesetzt ist.

2.1.6 Behälter (Basket)

Dank des *Basket*-Konzepts lassen sich die Daten selektiv nutzen. Baskets sind Behälter, die für das jeweilige Modell angelegt werden. Sie enthalten eine kompakte (abgeschlossene) Sammlung von Objekten. Im vorliegenden Fall bedeutet abgeschlossen, dass alle innerhalb des Themas zueinander in Beziehung stehenden Objekte in dem Basket enthalten sind. Im Rahmen von Transferdateien kann es mehrere Behälter pro Thema geben. Um zu ermitteln, in welchen Behälter die Features gehen, muss unbedingt für jedes Feature ein Verweis auf den Behälter (Basket) spezifiziert werden. Alle Behälter besitzen die gleiche Struktur (entsprechend dem Thema), können jedoch verschiedene Daten enthalten.

2.1.7 Interlis Tools

Die in diesem Kapitel vorgestellten Werkzeuge sind über das Webportal www.interlis.ch [9] kostenlos erhältlich (→ INTERLIS 2 → Downloads → Werkzeuge für INTERLIS 2.3).

- Compiler: dient der Überprüfung der Konformität eines «.ili»-Modells;
- Checker: dient der Überprüfung, dass eine Transferdatei (.xtf) mit dem entsprechenden Modell (.ili) übereinstimmt;
- UML-Editor: Modellierungswerkzeug, das den direkten Export in das INTERLIS-Modell ermöglicht.

2.2 FME

FME ist eine raumbezogene *ETL* (*Extract-Transform-Load*) Software zum Austausch, zur Transformation, zum Laden und zur Kontrolle von Vektor- oder Rasterdaten mit Raumbezug. FME ist in der Lage,

mehrere hundert verschiedene (Vektor-, Raster- und nicht raumbezogene) Formate zu lesen und zu schreiben, und ist daher als Werkzeug für die Bearbeitung von Geodaten unverzichtbar.

Für das Verständnis dieses Berichts empfiehlt es sich, bereits über eine gewisse Erfahrung in der Nutzung von FME zu verfügen.

Wir erinnern daran, dass auf dem Webportal SAFE kostenlose Tutorials online verfügbar sind:

<http://www.safe.com/learning/training/on-demand/tutorials/>

2.2.1 Transformationsprogramme

Falls bestimmte Transformationsprogramme, die in den Beispielen dieses Dokuments verwendet werden, Ihnen unbekannt sind, verweisen wir hierzu auf den FME Leitfaden für Transformationsprogramme [3].

2.2.2 Attribute

Attribute sind die zu den Features gehörigen Daten. Sie können entweder dem Format eigen sein (*format attributes*, in diesem Fall sind sie nicht direkt in FME verfügbar), oder für die jeweiligen Daten spezifisch sein (*user attributes*). Im Fall des Formats INTERLIS haben beispielsweise die formateigenen Attribute (*format attributes*) das Präfix *xtf_*:

- *xtf_id*
- *xtf_class*
- *xtf_basket*

Zur Geometrie einer Klasse: wenn FME in der Lage ist, beim Lesen einen Geometrietyp in den Daten zu erkennen, dann wird sie als implizite Geometrie definiert, und ist folglich direkt in FME verfügbar. Wenn hingegen mehrere Geometrien existieren, oder die Geometrie aus einer komplexen Struktur besteht, dann wird die Geometrie in einem Attribut gespeichert, und sie muss in dem FME Workspace wieder manuell extrahiert werden.

2.2.3 Nutzung der Listen

Um alle Etappen der Vorbereitung der Daten in FME vor dem Schreiben in INTERLIS über das Plugin *ili2fme* gut nachvollziehen zu können, ist es unerlässlich, das Konzept der «Listen» in FME verstanden zu haben.

Ein Attribut vom Typ Liste, das sich an geschwungenen Klammern «{}» in seinem Namen erkennen lässt, erlaubt mehrere Werte für eine Basisentität. Beispielsweise kann ein Feature folgende Attributwerte haben:

Attribut 1	3
Attribut 2	Red
MyList{0}	50
MyList{1}	45
MyList{2}	10

Dieses Feature enthält 3 Attribute: Attribut 1, Attribut 2 und eine Liste MyList{}. Die Liste enthält 3 Elemente. Jedes Element ist durch einen Index in geschwungenen Klammern gekennzeichnet.

Ausserdem kann die Liste ihrerseits ein oder mehrere Attribute enthalten. So wird das vorherige Beispiel etwas komplizierter:

Attribut 1	3
Attribut 2	Red
MyList.direction{0}	-1
MyList.value{0}	50
MyList.direction{1}	1
MyList.value{1}	45
MyList.direction{2}	-1
MyList.value{2}	10

Das Feature enthält noch immer dieselben 3 Attribute. Jedoch ist die Liste nun strukturiert, denn sie enthält ihrerseits zwei Attribute: Richtung und Wert. Die Liste enthält noch immer 3 Elemente, aber jedes hat zwei Attributwerte.

Quellen:

- <http://fmepedia.safe.com/articles/FAQ/List-Attributes>
- http://docs.safe.com/fme/html/FME_Transformers/FME_Transformers.htm#Transformers/listbuilder.htm

3. ALLGEMEINE INFORMATIONEN - *ILI2FME*

ili2fme ist ein INTERLIS-Plugin für FME, das von der Eisenhut Informatik AG entwickelt wurde und als OpenSource-Modul verfügbar ist.

In diesem Kapitel befassen wir uns schwerpunktmässig mit bestimmten Punkten, die für eine herkömmliche Nutzung des Plugin unumgänglich sind. Für alle weiteren Informationen verweisen wir auf die offizielle Dokumentation [4]:

<http://www.eisenhutinformatik.ch/interlis/ili2fme/interlis2-20140313.pdf>

Wir verweisen zudem auf die Existenz des Blogs www.ili2fme.ch [7], der Informationen zur Nutzung enthält, die sich in manchen Fällen als sehr nützlich erweisen können.

3.1 Software-/Hardwarevoraussetzungen

Zunächst einmal muss FME sich auf dem gewünschten System installieren und nutzen lassen. FME ist für Windows, Linux und Mac verfügbar. Soweit dies möglich ist, wird empfohlen, die jeweils letzten Versionen von FME zu installieren.

<http://www.safe.com/support/support-resources/fme-downloads/>

Ausserdem ist *ili2fme* ein in Java entwickeltes Plugin. Folglich ist es unerlässlich, dass eine Java-Umgebung (Java Runtime Environment), Version 1.6.0 oder jünger, auf dem System installiert ist.

<http://www.java.com/>

Ab FME 2014 wird eine Java-Umgebung direkt mit FME installiert.

3.2 Versionen

Das Plugin wird regelmässig nachgeführt, und die neuen Versionen werden auf dem Webportal www.interlis.ch bereitgestellt [9] (→ INTERLIS 2 → Downloads → Werkzeuge für INTERLIS 2.3).

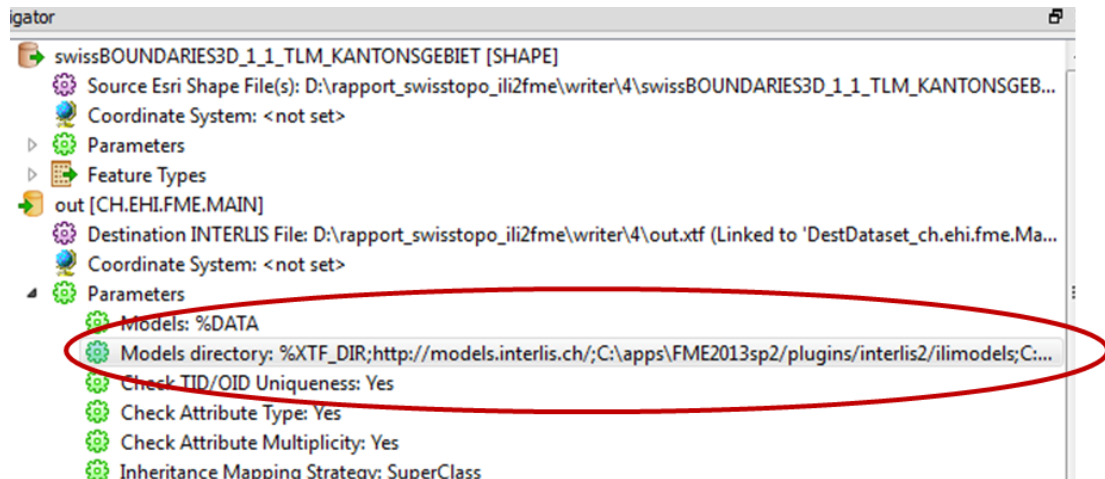
Generell ist es ratsam, mit der jeweils letzten verfügbaren Version von *ili2fme* zu arbeiten. Falls die installierte Version nicht die neueste ist, kann das Plugin nachgeführt werden (vgl. Kapitel 4.1).

3.3 Interlis-Modell und Transferdateien

Der Platz der Transferdatei (.*xtf*) muss in den Parametern ausdrücklich ausgewiesen sein, ganz gleich ob es sich um den Lese- oder Schreibmodus handelt.

Das oder die Modelle, auf die in der Transferdatei verwiesen wird, müssen in einem der Verzeichnisse verfügbar sein, das in dem Parameter *Models directory* des Plugin spezifiziert ist. Standardmässig werden die folgenden Werte zugewiesen:

```
%XTF_DIR; http://models.interlis.ch/; $(FME_HOME)plugins/interlis2/ilimodels;  
$(FME_HOME)plugins/interlis2/ili22models
```



Die INTERLIS-Modelle können somit in einem der folgenden Verzeichnisse gespeichert werden:

- %XTF_DIR: Ordner, der die Transferdateien enthält, im Lese- oder Schreibmodus;
- <http://models.interlis.ch/>: Online-Ablage der Referenzmodelle, die unter anderem auf das Verzeichnis <http://models.geo.admin.ch/> [8] zugreift; das die letzten Versionen der Modelle des Bundes enthält;
- eines der Verzeichnisse, die in den Unterlagen zur Installation von FME vorgesehen sind: \$(FME_HOME)\plugins\interlis2\ilimodels & \$(FME_HOME)\plugins\interlis2\ili22models.

Es kann sinnvoll sein, ein spezifisches Verzeichnis zur Speicherung zu definieren, wenn sehr viele Modelle verwaltet und zentralisiert werden müssen. So ist es möglich, ein lokales Verzeichnis hinzuzufügen, und zwar einfach indem ein Komma gesetzt und dann der Zugriffspfad des Verzeichnisses eingegeben wird.

3.4 Beschreibung der Parameter

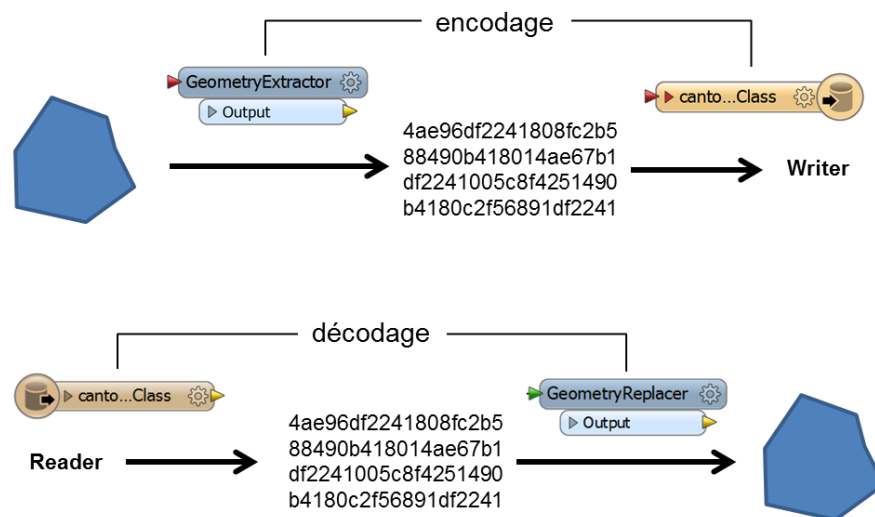
Nur die wichtigsten Parameter betreffend INTERLIS 2 werden hier aufgeführt:

- TOPICS FILTER: ermöglicht die Spezifizierung der Namen (*MODEL.TOPIC*) der Themen (TOPIC), die von FME gelesen werden;
- Check Attribute Type: ermöglicht die Überprüfung, dass der Typ der Attribute mit der Definition im Datenmodell übereinstimmt;
- Check TID/OID Uniqueness: ermöglicht die Überprüfung, dass alle Objekte einen eindeutigen Identifikator (*xtf_id*) besitzen;
- Geometry Encoding: ermöglicht *ili2fme* die Spezifizierung der in dem Workspace verwendeten Codierung, wobei der Codierungstyp mit demjenigen übereinstimmen muss, der in den Transformationsprogrammen *GeometryReplacer* oder *GeometryExtractor* benutzt wird;
- Inheritance Mapping Strategy: diese Option ist nur dann von Nutzen, wenn Klassen andere Klassen erben; durch Wahl der «SuperClass»-Strategie werden die Attribute der Hauptklasse (root class) sowie diejenigen der Nebenklassen in einer Klasse in FME zusammengeführt; durch Wahl der «SubClass»-Strategie erstellt FME eine Klasse je Unterklasse und bringt darin die Attribute der Hauptklasse unter;
- Create Feature Types For Enumerations: manche Interlis-Modelle enthalten Wertelisten; diese Listen sind im DOMAIN-Teil der .ili-Datei definiert; durch Aktivierung der Option «SingleType» schlägt FME beim Hinzufügen eines Readers eine Klasse namens XTF_ENUMS vor; dieses Element enthält sämtliche Werte aller Wertelisten des Modells; durch Wahl der Option «One feature type per enumeration» erhalten wir dann eine Klasse je Werteliste;

- Mapping of multiple Geometry Attributes: dieser seit der Version 5.10.0 existierende Parameter ermöglicht eine einfachere Visualisierung der Objekte, die multiple (mehrfache) oder mehrteilige Geometrien besitzen; standardmässig nimmt die Option *EncodeAsFmeAttribute* im Falle multipler Geometrien nur die erste Geometrie; bei mehrteiligen Geometrien müssen die in den Listen enthaltenen Informationen extrahiert werden; die Option *RepeatFeature* ermöglicht die Doppelung jedes Features, das mehrere Geometrien besitzt; somit kann FME sämtliche Geometrien direkt lesen;
- *http Proxy Host* und *http Proxy Port*: diese beiden Parameter ermöglichen die Definition eines Proxy-Servers, mit dem auf das Verzeichnis der Modelle zugegriffen werden kann.

3.5 Codierung der Geometrie

Manchmal muss in *ili2fme* die Geometrie codiert (verschlüsselt) oder decodiert werden. Dies ist insbesondere der Fall, wenn mehrere Geometrien definiert sind, oder wenn die Geometrie in einer Unterstruktur enthalten ist.



FME bietet mehrere Codierungstypen. Es ist wichtig, dass der im *GeometryExtractor* / *GeometryReplacer* gewählte Typ mit dem im *Reader* / *Writer* definierten Typ übereinstimmt. Um dies zu gewährleisten, kann die folgende Entsprechungstabelle nützlich sein.

Reader/Writer Parameter Geometry Encoding	Transformer <i>GeometryExtractor</i> oder <i>GeometryReplacer</i>
FMEXML	FME XML
FMEBIN	FME BINARY
FMEHEXBIN	HEX Encoded FME Binary
OGCHEXBIN	HEX Encoded OGC Well Known Binary

Generell empfiehlt sich die Verwendung von FMEBIN oder FMEXML. Tatsächlich kann es vorkommen, dass die beiden anderen Codierungstypen zum Verlust von Objekten führen, insbesondere dann, wenn diese Bögen enthalten.

3.6 Formatattribute

Beim Lesen oder Schreiben einer INTERLIS 2-Datei stösst man auf die folgenden, für das Format INTERLIS spezifischen Attribute:

- *xtf_class*: ermöglicht die Definition, in welcher Klasse sich die Attribute befinden (oder für welche sie bestimmt sind); in dieser Definition wird auch das entsprechende *TOPIC* berücksichtigt;
- *xtf_id*: ermöglicht die Angabe eines eindeutigen Identifikators für jedes Feature;
- *xtf_basket*: ermöglicht die Spezifizierung der Zugehörigkeit zu einem Behälter (Basket).

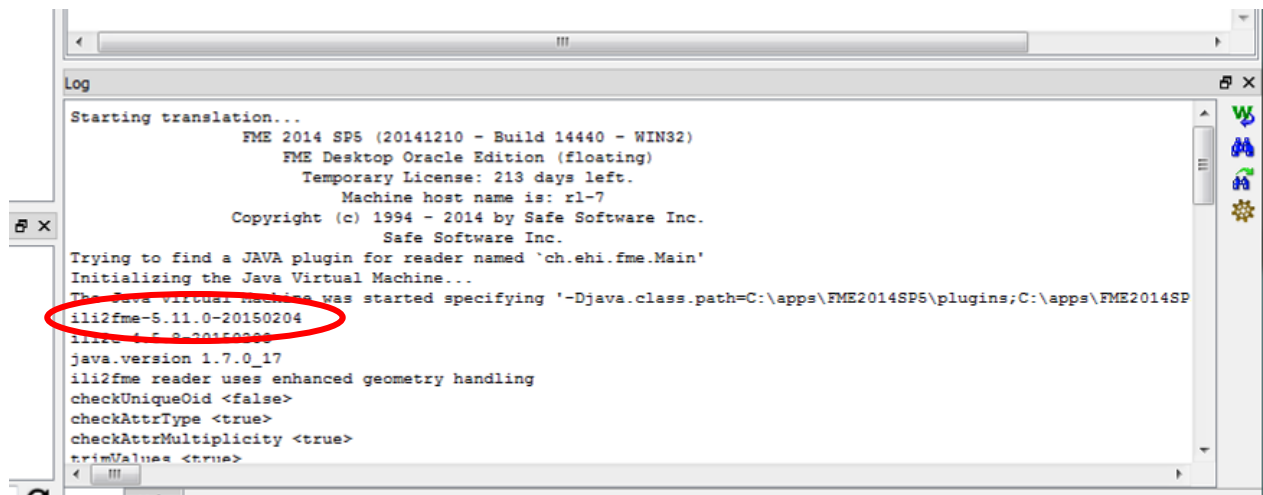
4. SCHNELLSTART

4.1 Installation

Das Plugin *ili2fme* wird direkt bei der Installation von FME installiert. Dies impliziert, dass das Format Swiss INTERLIS (*ili2fme*) sofort in der Liste der Formate verfügbar ist, und zwar für alle Module des Softwarepakets FME (Workbench, QuickTranslator, Inspector und Viewer).

Zur Überprüfung, ob das Plugin korrekt auf dem System installiert ist, genügt es, in das Menü *Readers* und anschliessend in *Add Readers* zu gehen und das Format *Swiss INTERLIS (ili2fme)* auszuwählen. Wenn der Reader korrekt hinzugefügt ist, funktioniert das Plugin.

Die Version des installierten Plugins lässt sich am einfachsten ermitteln, indem man eine Interlis-Datei in FME öffnet. Das Log-Fenster (im unteren Teil des Bildschirms) zeigt Informationen über das geladene Plugin. In der mit *ili2fme* beginnenden Zeile werden die Versionsnummer des Plugins, und dahinter das Datum der Freigabe (Release) angezeigt.



Zur Nachführung von *ili2fme* steht die neueste von swisstopo genehmigte Version von FME auf dem Webportal <http://www.interlis.ch/> zum Download bereit [9] (→ INTERLIS 2 → Downloads → Werkzeuge für INTERLIS 2.3). Die Installation erfolgt durch Entpacken der Datei und Ersetzen der Verzeichnisse und Dateien in der *FME Suite* an den entsprechenden Orten im Installationsverzeichnis von FME. Nähere Informationen finden sich in der Datei *README.txt* des Verzeichnisses *docs*.

4.2 Schnelle Konvertierung

Mit Hilfe des FME Quick Translator lässt sich eine INTERLIS-Datei rasch konvertieren. Hierfür genügt es, die Interlis-Datei als Reader, und ein anderes Format als Writer zu spezifizieren.

Diese schnelle Art der Transformation weist jedoch einige Einschränkungen auf:

- Die durchgeführte Konvertierung ist vom Typ 1:1. Jede im INTERLIS-Modell vorgefundene Klasse wird in eine Klasse im Ausgabeformat konvertiert. Häufig ist es jedoch so, dass zusätzliche Klassen im Schreibformat erforderlich sind, oder aber Vereinfachungen möglich sind. Beispielsweise lässt sich im Fall einer gewünschten Konvertierung von INTERLIS nach SHAPE ein mehrsprachiges INTERLIS-Attribut (vgl. Beispiel 5.3.1) nicht direkt im Shapefile speichern. Folglich muss manuell eine zweite Klasse zur Ausgabe (beispielsweise in .DBF) eingerichtet werden, welche die verschiedenen Werte entsprechend der Sprache sowie einen Fremdschlüssel zur Herstellung der Beziehung enthält.

- Falls bestimmte INTERLIS-Strukturen in Listenform in FME gespeichert werden (beispielsweise mehrteilige Geometrien), funktioniert die direkte Konvertierung nicht. Konvertiert ein Nutzer etwa die im ersten Beispiel angelegte *xtf*-Datei (vgl. 5.1), so erhält er nur die Attributdaten im neuen Format. Für eine vollständige und erschöpfende Übersetzung einer INTERLIS 2-Datei muss die Nutzung des Plugins *ili2fme* durch eine Reihe von Operationen in einem FME Workspace ergänzt werden. Die Nutzung der FME Workbench ist also in diesem Fall unverzichtbar.

4.3 Schnelle Visualisierung

Die Werkzeuge FME Data Inspector oder FME Universal Viewer können genutzt werden, um den Inhalt einer INTERLIS-Datei rasch zu visualisieren. Aus den bereits im Kapitel 4.2 erläuterten Gründen wird eine mehrteilige oder multiple Geometrie jedoch nur dann vollständig sichtbar, wenn *RepeatFeature* im Parameter *Mapping of multiple Geometry* gewählt wird. Hingegen finden eventuelle Verbindungen zwischen den Klassen keine Berücksichtigung.

Nutzer von ArcGIS Desktop können INTERLIS-Dateien visualisieren und bearbeiten. Hierfür muss die FME Erweiterung für ArcGIS in der FME Integration Console aktiviert werden. Nähere Informationen sind der für ArcGIS verfügbaren Online-Dokumentation [6] zu entnehmen.

5. ANWENDUNGSFÄLLE

Alle in diesem Dokument beschriebenen Workspaces sind in den ergänzenden Unterlagen zum vorliegenden Dokument verfügbar und wurden mit FME 2014 SP4 entwickelt.

5.1 Schreiben einer INTERLIS 2-Datei

Dieses Kapitel nimmt Bezug auf den Workspace *w-Doc_ili2fme_Ex1-isa.fmw*

Schlüsselwörter:

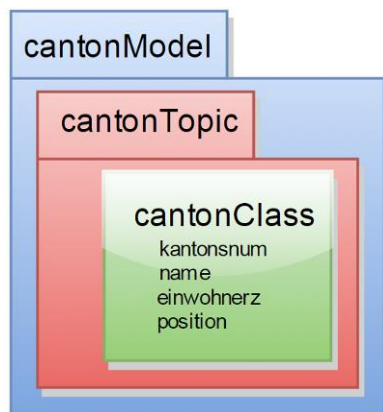
Mehrteilige Geometrie	INTERLIS-Struktur	INTERLIS Writer	Codierung
-----------------------	-------------------	-----------------	-----------

5.1.1 Darstellung des Falles

Ziel dieses Beispiels ist die Erzeugung einer INTERLIS 2-Datendatei (.xtf) aus einer SHAPE-Datei. Hierzu verfügen wir über die Datei¹, welche die Kantons Grenzen und folgende Attribute enthält:

- die Nummer des Kantons (*KANTONSNUM*),
- den Namen des Kantons (*NAME*),
- die Einwohnerzahl (*EINWOHNERZ*),
- die Geometrie des Kantons.

Das INTERLIS-Modell, in dem wir schreiben möchten, ist folgendermassen strukturiert:



```
INTERLIS 2.3;

MODEL cantonModel
AT "http://www.inser.ch"
VERSION "10.02.2015" =
IMPORTS GeometryCHLV03_V1;

TOPIC cantonTopic =

CLASS cantonClass =
kantonsnum: TEXT*30;
name: TEXT*30;
einwohner: TEXT*30;
position : GeometryCHLV03_V1.MultiSurface;
END cantonClass;

END cantonTopic;

END cantonModel.
```

Dieses Modell besteht aus folgenden Elementen:

- das Modell *cantonModel* (MODEL) – enthält die Gesamtheit der Daten sowie potenziell Einheiten- (UNIT) oder Bereichs-Definitionen (DOMAIN);
- das Thema *cantonTopic* (TOPIC) – ein spezifischerer Teil des Modells, der ebenfalls Definitionen und Attribute enthalten kann.
- die Klasse *CantonClass* (CLASS) – enthält alle Objekte, welche dieselben Eigenschaften teilen. So werden die Attribute und ihre Typen definiert. In unserem Beispiel finden wir unsere drei zuvor beschriebenen Textattribute wieder. Wir stellen fest, dass sie vom Typ Text sein müssen und maximal 30 Zeichen enthalten dürfen (TEXT*30). Der Typ des Attributs Position wiederum ist im Modell *cantonModel* nicht direkt bestimmt. Tatsächlich greift es auf ein Basismodul CHBase (vgl.

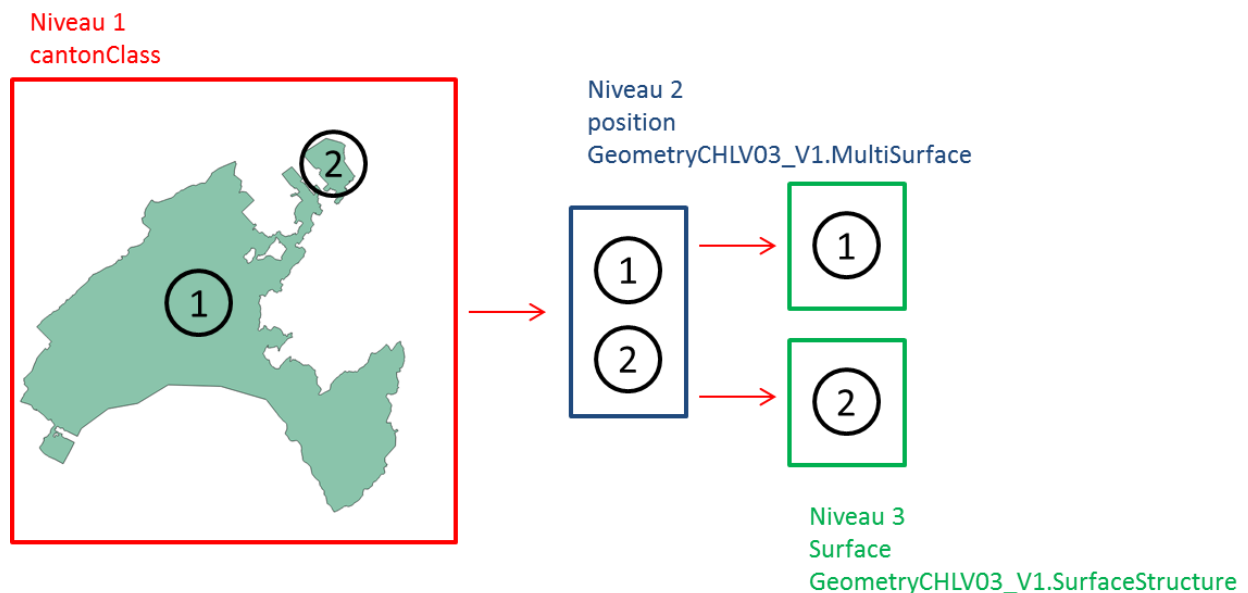
¹ Verfügbar im Internet unter http://www.toposhop.admin.ch/de/shop/products/landscape/swissBoundaries3D_1

Kapitel 2.1.4) namens GeometryCHLV03_V1.MultiSurface zurück. Letzteres wird dank der Zeile am Anfang des Modells *IMPORTS GeometryCHLV03_V1* geladen.

Schauen wir uns diese Besonderheit etwas genauer an:

Modell cantonModel.ili	Modell CHBase_Part1_GEOMETRY_20110830_20150220.ili
<pre> INTERLIS 2.3; MODEL cantonModel AT "http://www.inser.ch" VERSION "10.02.2015" = IMPORTS GeometryCHLV03_V1; UNIT DOMAIN TOPIC cantonTopic = CLASS cantonClass = kantonsnum: TEXT*30; name: TEXT*30; einwohner: TEXT*30; position : GeometryCHLV03_V1.MultiSurface; END cantonClass; END cantonTopic; END cantonModel. </pre>	<pre> STRUCTURE MultiSurface = Surfaces: BAG {1..*} OF SurfaceStructure; END MultiSurface; STRUCTURE SurfaceStructure = Surface: Surface; END SurfaceStructure; Surface = SURFACE WITH (STRAIGHTS, ARCS) VERTEX Coord2 WITHOUT OVERLAPS > 0.0001; DOMAIN Coord2 = COORD 460000.000 .. 870000.000 [m] {CHLV03[1]}, 45000.000 .. 310000.000 [m] {CHLV03[2]}, ROTATION 2 -> 1; </pre>

Das Attribut Position stimmt mit der Struktur MultiSurface überein. Letztere kann aus einer Vielzahl von SurfaceStructures bestehen. Jede SurfaceStructure besteht aus einer Fläche, die aus Koordinaten gebildet wird (Coord2). Die Grenze dieser Koordinaten ist in dem Bereich definiert. Diese Art und Weise, die Geometrie zu definieren, ist wichtig, um mehrteilige Objekte speichern zu können. Dies gilt insbesondere für den Kanton Waadt, der eine Enklave im Kanton Freiburg besitzt.



So umfasst das Element «Kanton Waadt» zwei Geometrien. Diese Besonderheit muss unbedingt im FME Workspace berücksichtigt werden, um die INTERLIS 2-Datei zu generieren. Diese muss also die Geometrie in folgender Weise speichern können:

```
<GeometryCHLV03_V1.MultiSurface>
  <Surfaces>
    <GeometryCHLV03_V1.SurfaceStructure>
      <Surface>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <COORD></COORD>
              <COORD></COORD>
              .... Koordinaten des Polygons ...
            </POLYLINE>
          </BOUNDARY>
        </SURFACE>
      </Surface>
    </GeometryCHLV03_V1.SurfaceStructure>
    <GeometryCHLV03_V1.SurfaceStructure>
      <Surface>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <COORD></COORD>
              <COORD></COORD>
              .... Koordinaten des Polygons ...
            </POLYLINE>
          </BOUNDARY>
        </SURFACE>
      </Surface>
    </GeometryCHLV03_V1.SurfaceStructure>
  </Surfaces>
</GeometryCHLV03_V1.MultiSurface>
```

5.1.2 Einrichtung des FME Workspace

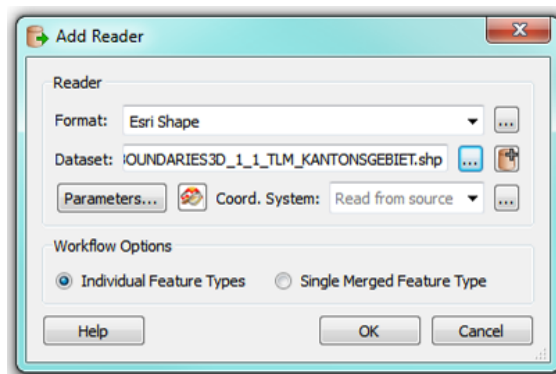
1. Bearbeitung der SHAPE-Datei

1.1. Zunächst werden wir die SHAPE-Datei lesen. Hierfür muss ein ESRI Shape Reader hinzugefügt werden:

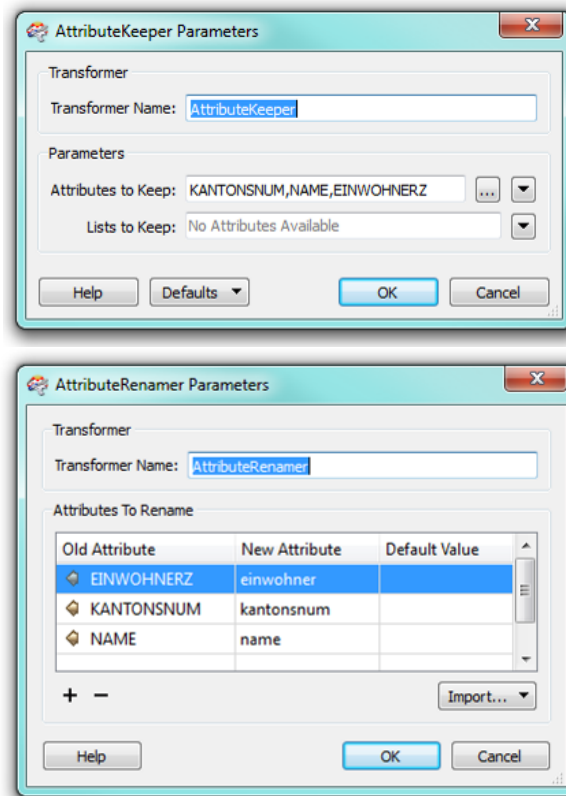
Menu Reader -> Add reader

1.2. Dann in Format -> ESRI Shape auswählen

1.3. Den Zugriffspfad der Datei in das Feld Dataset eingeben

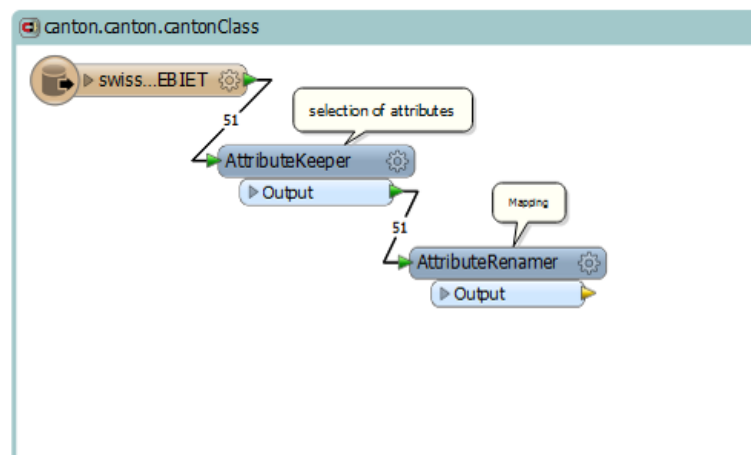


1.4. Es ist wichtig, dass die Namen der Attribute der SHAPE-Datei und die Namen der im INTERLIS-Datenmodell definierten Attribute übereinstimmen. Um dies zu gewährleisten, werden wir lediglich die Attribute beibehalten, die wir anschliessend verwenden werden, und wir werden sie danach umbenennen. Wir benutzen nacheinander einen AttributeKeeper und einen AttributeRenamer.

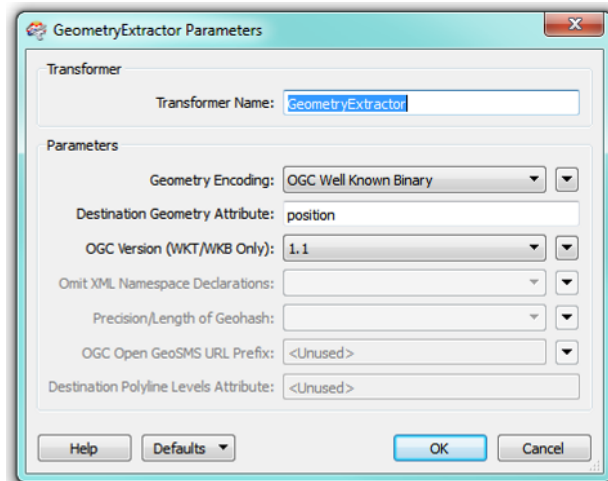


2. Bearbeitung speziell zum Schreiben in INTERLIS 2

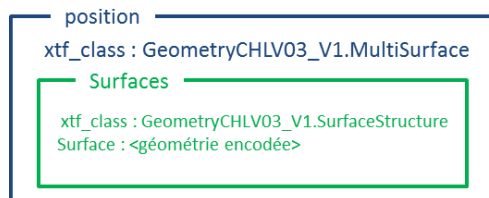
Hier muss sich unsere Verfahrensweise aktuell an der nachstehend beschriebenen orientieren.



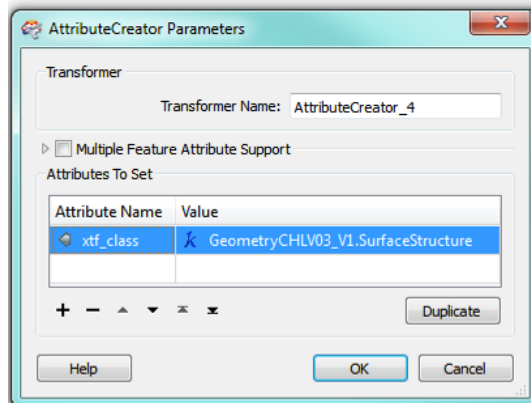
2.1. Um die Geometrie korrekt speichern zu können, muss sie codiert werden. Hierfür werden wir das Transformationsprogramm *GeometryExtractor* nutzen. Es sind mehrere Codierungsformate verfügbar. Im Rahmen dieses Tutorials wählen wir das Format *OGC Well Known Binary* im Pulldown-Menü *Geometry Encoding*. Das Zielattribut muss einen ähnlichen Namen wie im Modell tragen, im vorliegenden Fall also *position*.



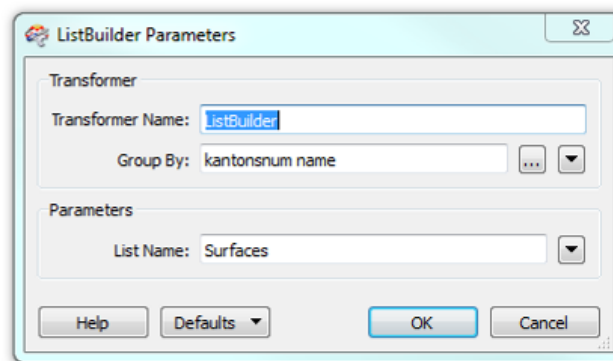
2.2. Nun müssen wir alle Geometrien, die wir gerade codiert haben, in einem einzigen Los pro Feature zusammenfassen. Hierzu müssen wir für jeden Kanton alle Geometrien in Listen zusammenfassen, und zwar unter Einhaltung des Datenmodells.



2.3. Nun werden wir die erste Liste (Ebene 3) erstellen, diejenige auf Ebene der Klasse GeometryCHLV03_V1.SurfaceStructure (sie wird also alle unsere Geometrien zusammenfassen (aggregieren)). Hierfür werden wir mit Hilfe eines *AttributeCreator* ein neues Attribut erstellen.

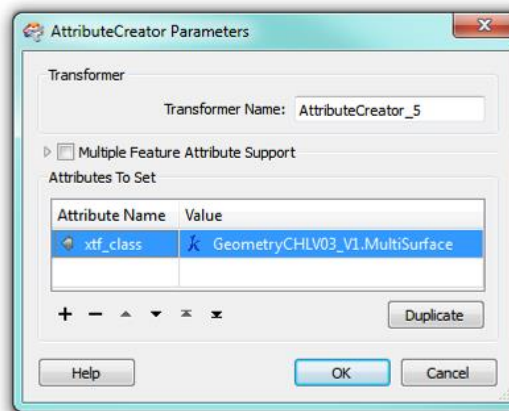


Dieses Attribut ermöglicht es, die Klasse der Liste zu spezifizieren, die wir folgendermassen erstellen werden:

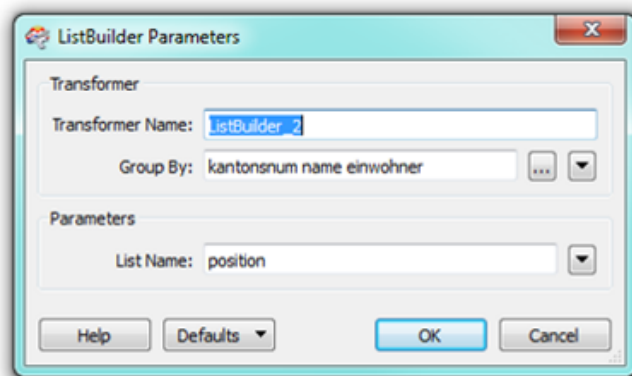


Mit diesem Befehl haben wir eine Liste der nach Kantonsnummer und -name gruppierten Attribute erstellt (wobei diese Attribute für jeden Kanton identisch sind).

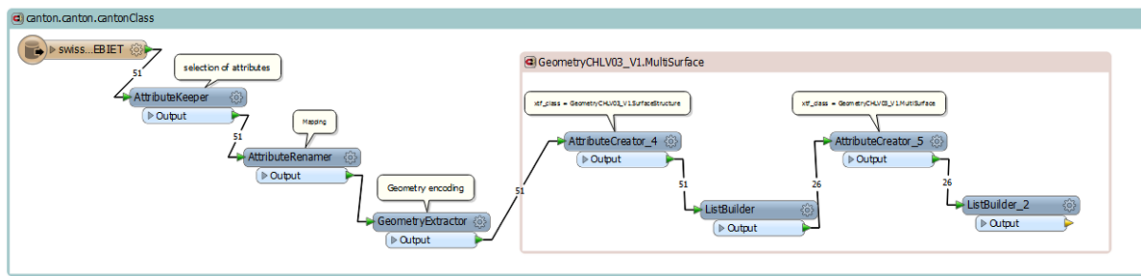
2.4. Eben diese Operation können wir nun auf der höheren Ebene (Ebene 2, diejenige der Klasse GeometryCHLV03_V1.MultiSurface) erneut durchführen.



Die Erstellung dieser zweiten Liste geschieht folgendermassen:



So stellt man fest, dass die zweite Liste die erste einschliesst, und dass wir auf jeder Ebene das Attribut *xtf_class* eingetragen haben. An diesem Punkt muss der Workspace so aussehen:

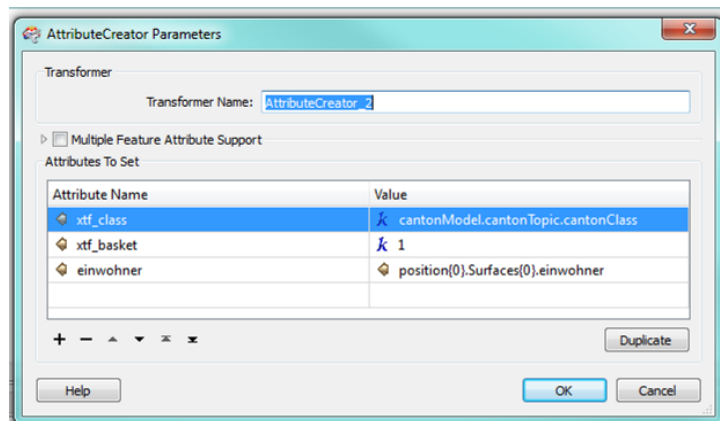


Die Technik, die darin besteht, mehrere Listen zu erstellen, um die Daten mit dem *ili*-Modell in Übereinstimmung zu bringen, ist ein grundlegendes Konzept für das Schreiben und Lesen von INTERLIS 2-Dateien über das Plugin *ili2fme*.

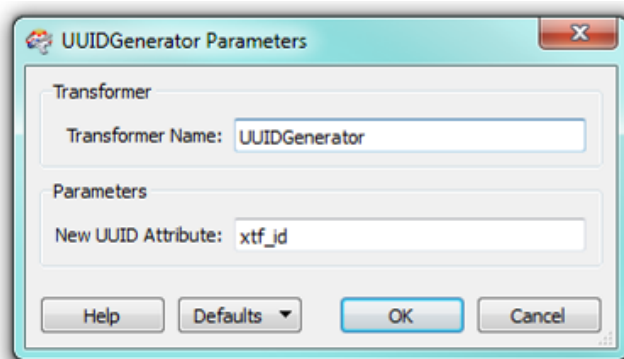
2.5. Nun können wir uns mit der letzten Ebene (Ebene 1) befassen, nämlich derjenigen der Klasse *cantonClass*. Wie bereits zuvor müssen wir ein Attribut *xtf_class* hinzufügen, dessen Wert folgendermassen definiert ist: `<model>.<topic>.<class>`.

Im vorliegenden Fall nimmt dies die Werte *cantonModel.cantonTopic.cantonClass* an. Ausserdem werden wir einen einzigen *Behälter (Basket)* einrichten, um sämtliche Informationen zu speichern. Daher müssen wir allen Entitäten das Attribut *xtf_basket* = 1 hinzufügen.

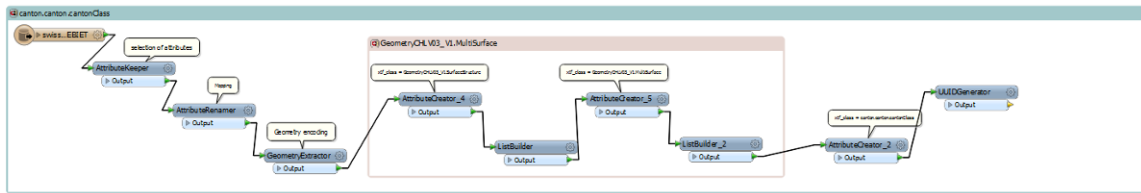
Das Attribut *einwohner* wurde in die Listen einbezogen. Tatsächlich ist dieses Attribut nicht für alle Kantonsteile identisch; nur der Hauptteil besitzt einen Wert. Daher müssen wir ihn manuell aus den Listen extrahieren.



2.6. Zum Abschluss müssen wir auch ein Attribut *xtf_id* hinzufügen, das ein eindeutiger Identifikator für jedes Objekt sein muss. Wir verwenden einen *UUIDGenerator*.

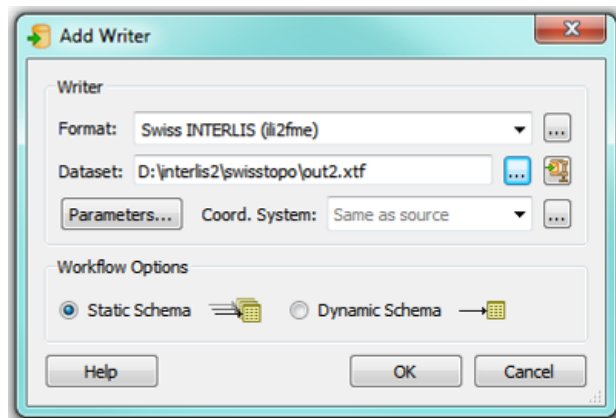


Unser Workspace sieht nun folgendermassen aus:



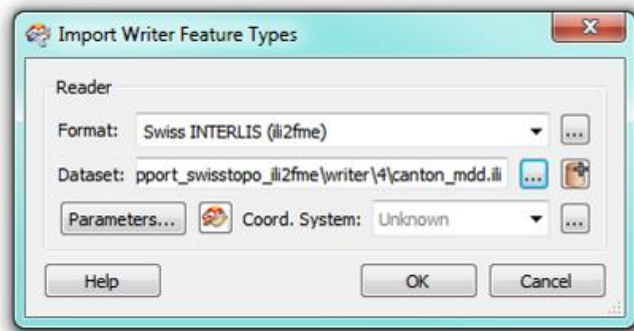
3. Schreiben im Format INTERLIS 2

3.1. Hierfür müssen wir einen Writer hinzufügen. Wir wählen das Format *Swiss INTERLIS (ili2fme)* und anschliessend den Datensatz in INTERLIS *Files .xtf*



3.2. Nachdem wir den Writer hinzugefügt haben, schlägt uns FME vor, automatisch einen neuen *feature type* zu erstellen. Da die Struktur unserer Writer in Abhängigkeit von den *ili*-Modellen zu definieren ist, müssen wir diese Frage durch Klicken auf NO beantworten.

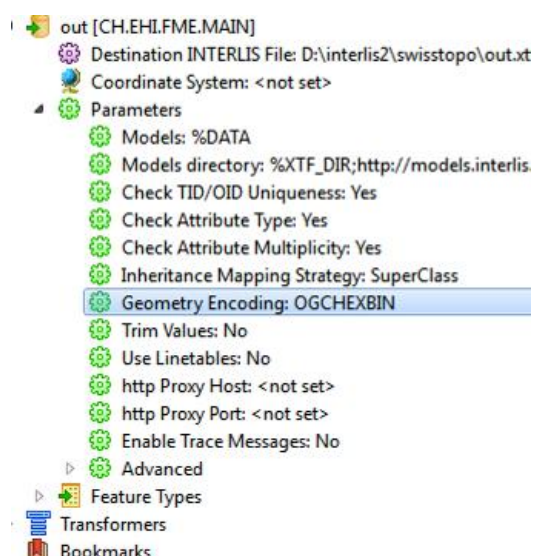
3.3. Wir fügen das Datenmodell hinzu, indem wir in Writer -> Import Feature Type gehen und anschliessend das .ili-Datenmodell auswählen.



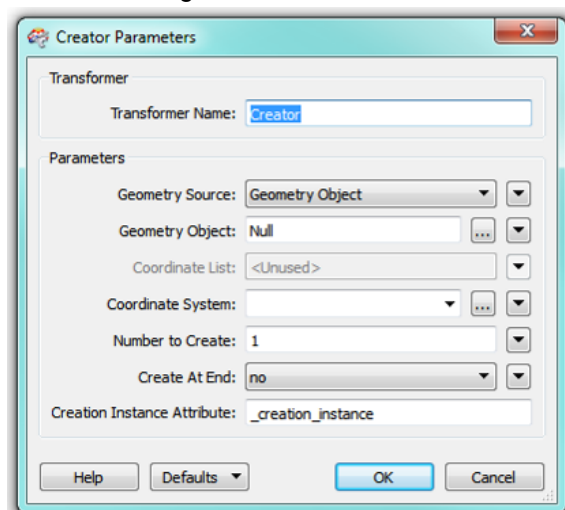
3.4. Im vorliegenden Fall dürfen nur die Typen *cantonModel.cantonTopic.cantonClass* und *XTF_BASKET* ausgewählt werden.

3.5. So kann der Writer *cantonModel.cantonTopic.cantonClass* mit der zuvor eingerichteten Vorgehensweise verknüpft werden.

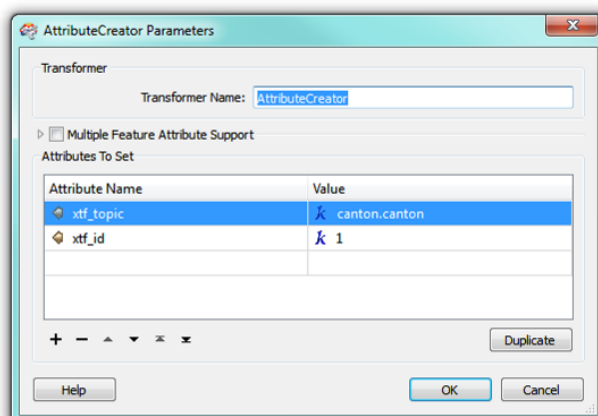
3.6. In den Parametern des Writers müssen wir überprüfen, dass die Codierung der Geometrie tatsächlich mit der zuvor definierten [Codierung] übereinstimmt. In unserem Fall müssen wir *OGCHEXBIN* wählen.



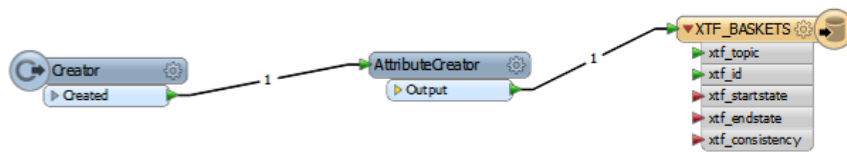
3.7. Zum Abschluss muss auch das *XTF_BASKETS* ausgefüllt werden. Für jedes Thema (*TOPIC*) muss ein Objekt erstellt werden, das die Attribute *xtf_id* und *xtf_topic* enthält. Hierfür müssen wir einen *Creator* hinzufügen, und zwar folgendermassen:



3.8. Dann erzeugen wir die Attribute *xtf_topic* = canton.canton und *xtf_id* = 1.



3.9. Das Ganze kann mit dem Writer *XTF_BASKETS* verknüpft werden.



5.2 Lesen einer INTERLIS 2-Datei

Dieses Kapitel nimmt Bezug auf den Workspace *w-Doc_ili2fme_Ex2-isa.fmw*

Schlüsselwörter:

Listen	INTERLIS-Struktur	INTERLIS Reader	Codierung
--------	-------------------	-----------------	-----------

5.2.1 Darstellung des Falles

Im Rahmen dieses zweiten Beispiels verfügen wir über eine INTERLIS-Datei, die Daten über die Kantone enthält (deskriptive und geometrische Daten). Ein Kanton kann sich aus mehreren Teilen zusammensetzen (Hauptteil und Enklave). Die Geometrie ist also mehrteilig, und ihre Struktur entspricht der Beschreibung im Kapitel 5.1.1.

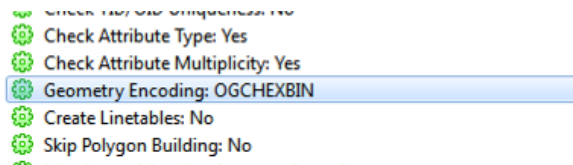
5.2.2 Einrichtung des FME Workspace

1. Lesen der INTERLIS-Datei

1.1. Im Reiter *Readers* die Option *Add reader* auswählen.

1.2. Als Format ist Swiss INTERLIS (ili2fme), und als Datensatz die Datei *cantons.xtf* auszuwählen.

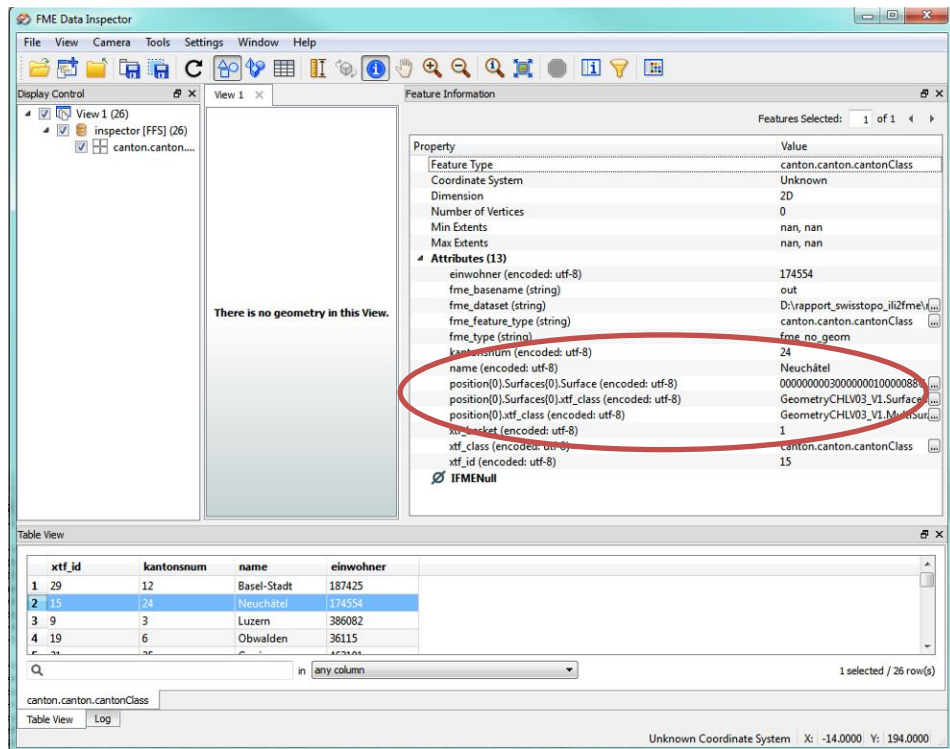
1.3. Durch Anklicken der Schaltfläche *Parameters* müssen wir den Codierungstyp der Geometrie spezifizieren. Im vorliegenden Fall handelt es sich um OGCHXBIN.



Die beiden Fenster sind mit OK zu bestätigen, um die Daten zu importieren.

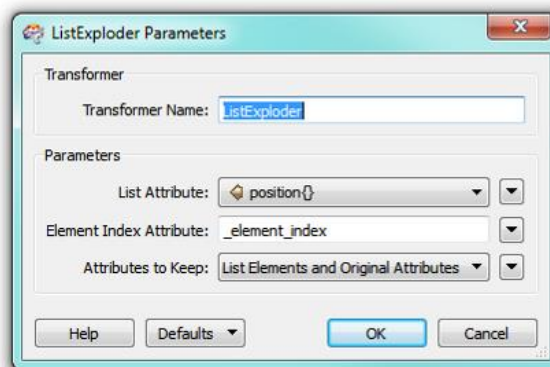
1.4. FME schlägt den Import aller Klassen vor, die in unseren Datenmodellen vorhanden sind (das allgemeine, *cantonModel.ili*, sowie alle über die Balise *IMPORT* importierten, vgl. Kapitel 2.1.3). Im vorliegenden Fall werden wir nur die Klasse *cantonClass* importieren.

1.5. Durch Verknüpfung eines *Inspector*s mit dem frisch importierten *Reader* erhalten wir keine Geometrie. Wir können jedoch feststellen, dass vier Attribute vorhanden sind. Das erste Attribut stimmt mit dem *xtf_id* überein, und die nachfolgenden sind die deskriptiven Attribute der Kantone. Die Geometrie wird in Form von Listen gespeichert. Auf diese Weise lassen sich mit *ili2fme* mehrteilige Geometrien zusammenfassen. Durch Anklicken eines Werts der angezeigten Attribute können wir sie im Fenster *Feature Information* sehen.

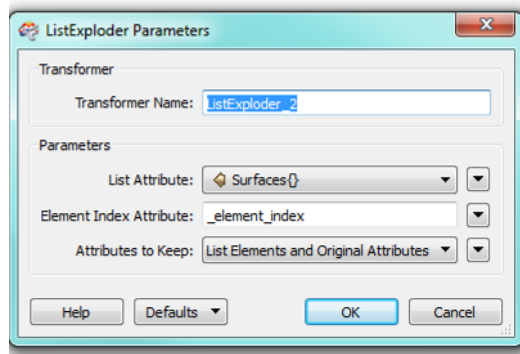


Um diese Informationen nutzen zu können, müssen diese Listen extrahiert werden.

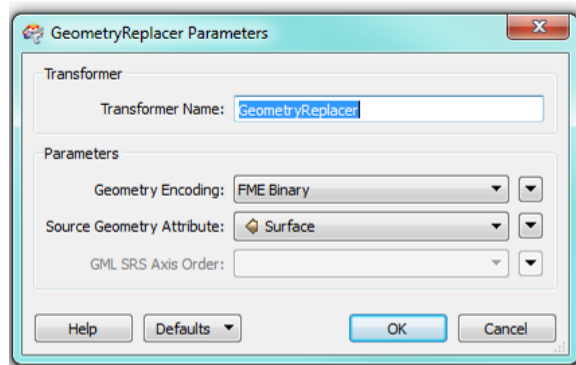
1.6. Mit Hilfe eines ListExploders werden wir dann die Liste `position{}` aufbrechen, was dem Namen des Attributs Geometrie im ili-Modell entspricht.



Indem wir erneut einen Inspector mit dem Ausgang dieses ListExploders verknüpfen, können wir sehen, dass das Attribut `xtf_class` erzeugt wurde. Die Werte dieses Attributs stimmen mit der Geometrieklasse der Kantone überein. Es ist jedoch noch immer nicht möglich, die Geometrie zu visualisieren. Hierfür müssen wir in Gestalt der Liste `Surfaces{}` eine zweite Liste exponieren.

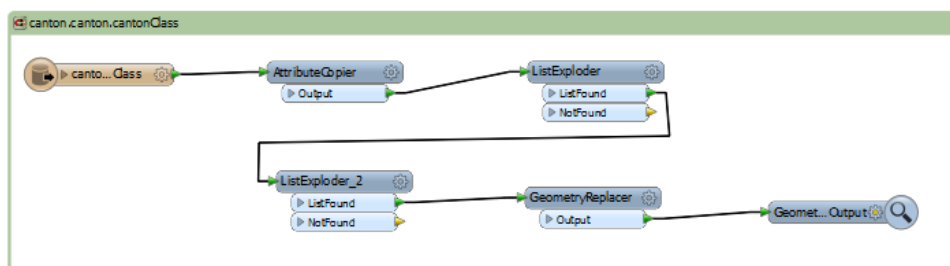


1.7. Somit wird das Attribut Surface exponiert. Man stellt fest, dass es codiert ist (weshalb es so wichtig ist, das Format im Punkt 1.3 korrekt auszuwählen). Wir werden das Transformationsprogramm GeometryReplacer verwenden, um dieses Attribut zu decodieren und die Geometrie zu erzeugen.



Durch Verknüpfung eines Inspectors können wir nun die Geometrie der Objekte2 visualisieren. Ab diesem Punkt können zahlreiche Transformationsprogramme genutzt werden, um diese Geometrie zu kennzeichnen (AreaCalculator, CircularityCalculator, usw.).

Der Workspace muss nun folgendermassen aussehen:



Schlussbemerkung: Dieses Beispiel ermöglicht das Lesen der INTERLIS-Datendatei in Abhängigkeit von ihrem Modell. Jede Liste entspricht einer Ebene (vgl. 5.1.1). Dieser Ebenenbegriff ist vor allem beim Schreiben einer INTERLIS-Datei von Belang. In diesem Beispiel hätte das Lesen der Datei durch Auswahl der Option *RepeatFeature* im Parameter *Mapping of multiple Geometry Attribute* vereinfacht werden können.

² Falls ein Fehler @Geometry function in dieser Etappe auftaucht, so entspringt dies wahrscheinlich einem Codierungskonflikt zwischen der *xtf*-Datei, der Reader-Definition (vgl. Punkt 1.3) und/oder der Definition des *GeometryReplacer* (Punkt 1.7).

5.3 Lesen und Schreiben von INTERLIS 2-Dateien mit geschachtelten Strukturen

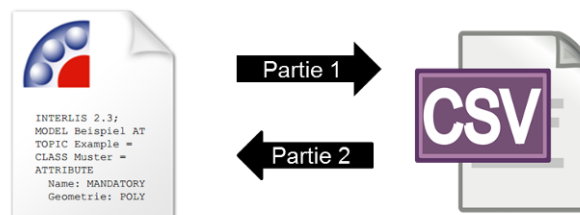
Dieses Kapitel nimmt Bezug auf die Workspaces *w-Doc_ili2fme_Ex5_3_1-isa.fmw* und *w-Doc_ili2fme_Ex5_3_2-isa.fmw*.

Schlüsselwörter:

Geschachtelte Struktur INTERLIS-Struktur INTERLIS Reader/Writer CHBase

5.3.1 Darstellung des Falles

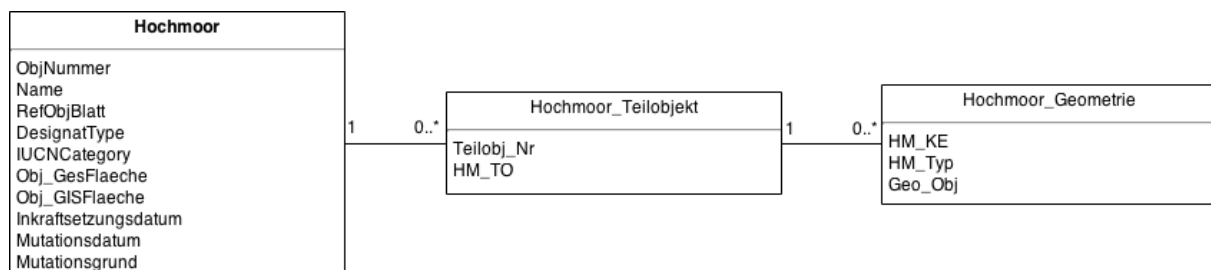
Dieses Beispiel wird in zwei Teile zerlegt. Im ersten Teil besteht das Ziel darin, die in der zugrunde liegenden INTERLIS-Datei (*hochmoore.xtf*) enthaltenen Informationen in eine CSV-Datei (*out.csv*) zu schreiben. Im zweiten Teil werden wir die umgekehrte Operation ausführen, indem wir ausgehend von der CSV-Datei eine neue INTERLIS-Datei (*out.itf*) erzeugen. Somit müssen die beiden XTF-Dateien ähnlich sein.



Wir werden dieses Beispiel auf den konkreten Fall eines INTERLIS-Modells stützen. Wir verfügen über eine Datendatei, die Informationen über Hochmoore enthält (deskriptive und geometrische Daten). Anzumerken ist auch, dass das ursprüngliche Modell *Hochmoore_V1* für die Zwecke dieser Übung geringfügig verändert wurde. Es trägt nun den Namen *Hochmoore_V1_is*.

Das Datenmodell ist etwas komplexer als das in den Beispielen 5.1 und 5.2. verwendete. Da ein Moor aus mehreren Wasserflächen bestehen kann, muss das Objekt «Moor» mehrere Geometrien innerhalb der INTERLIS-Datei speichern können. Wir sprechen in diesem Fall von mehrteiligen Geometrien.

Das Datenmodell nimmt folgende Form an:



Ein Objekt der Klasse *Hochmoor* kann mehrere *Hochmoor_Teilobjekte* haben. Ebenso können diese Teilobjekte in mehreren Geometrien beschrieben sein.

Das Datenmodell (*Hochmoore_V1_is.ili*) umfasst also diese drei Klassen sowie Assoziationsparameter, die Auskunft über die Art der Beziehung zwischen den Tabellen geben. Beispielsweise enthält die Klasse *Hochmoor* ein oder mehrere Elemente in der Klasse *Hochmoor_Teilobjekt*.

```
ASSOCIATION AssociationDef194 =
  Hochmoor_Geometrie -- {1..*} Hochmoor_Geometrie;
  Hochmoor_Teilobjekt -<#> {1} Hochmoor_Teilobjekt;
END AssociationDef194;
```

Vor dem Beginn des Einlesens der INTERLIS 2-Datei in FME ist es aufschlussreich, die Definition der Klasse *Hochmoor_Geometrie* etwas genauer zu analysieren.

```
CLASS HM_KE_Catalogue
EXTENDS CatalogueObjects_V1.Catalogues.Item =
  Code : MANDATORY TEXT*7;
  Description : MANDATORY LocalisationCH_V1.MultilingualText;
END HM_KE_Catalogue;

STRUCTURE HM_KE_CatRef
EXTENDS CatalogueObjects_V1.Catalogues.CatalogueReference =
  Reference (EXTENDED) : REFERENCE TO HM_KE_Catalogue;
END HM_KE_CatRef;

CLASS Hochmoor_Geometrie =
  HM_KE : MANDATORY Hochmoore_V1.Codelisten.HM_KE_CatRef;
  HM_Typ : MANDATORY Hochmoore_V1.Codelisten.HM_Typ_CatRef;
  Geo_Obj : MANDATORY GeometryCHLV03_V1.MultiSurface;
END Hochmoor_Geometrie;
```

Wir stellen fest, dass das erste Attribut, *HM_KE*, auf eine Codeliste Bezug nimmt. Diese (in Form einer XML-Datei gespeicherte) Codeliste bezieht sich auf die Struktur *HM_KE_CatRef*. Diese wiederum ist durch die Klasse *HM_KE_Catalogue* definiert. Wir stellen fest, dass das Attribut *Description* dem Typ *MultilingualText* angehört. Schauen wir uns einen Auszug aus der XML-Datei genauer an, dann sehen wir, dass es möglich ist, Attribute mit vordefinierten Wertelisten in Beziehung zu setzen. Im vorliegenden Fall kann jeder *TID* eine deutsche, französische oder italienische Bezeichnung für einen bestimmten Begriff liefern.

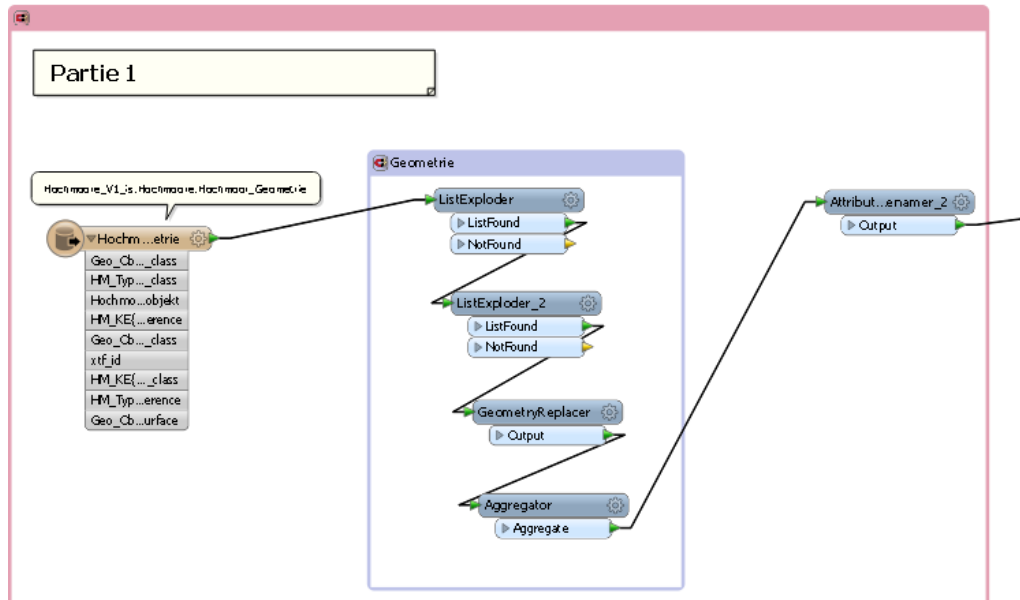
```
<Hochmoore_V1_is.Codelisten.HM_KE_Catalogue TID="2011">
  <Code>HM_KE11</Code>
  <Description>
    <LocalisationCH_V1.MultilingualText>
      <LocalisedText>
        <LocalisationCH_V1.LocalisedText>
          <Language>de</Language>
          <Text>Niedermoor, Verlandung</Text>
        </LocalisationCH_V1.LocalisedText>
        <LocalisationCH_V1.LocalisedText>
          <Language>fr</Language>
          <Text>Bas-marais, atterrissement</Text>
        </LocalisationCH_V1.LocalisedText>
        <LocalisationCH_V1.LocalisedText>
          <Language>it</Language>
          <Text>Palude bassa, interrimento</Text>
        </LocalisationCH_V1.LocalisedText>
      </LocalisedText>
    </LocalisationCH_V1.MultilingualText>
  </Description>
</Hochmoore_V1_is.Codelisten.HM_KE_Catalogue>
```

Diese Besonderheit muss natürlich in den FME Workspace integriert werden.

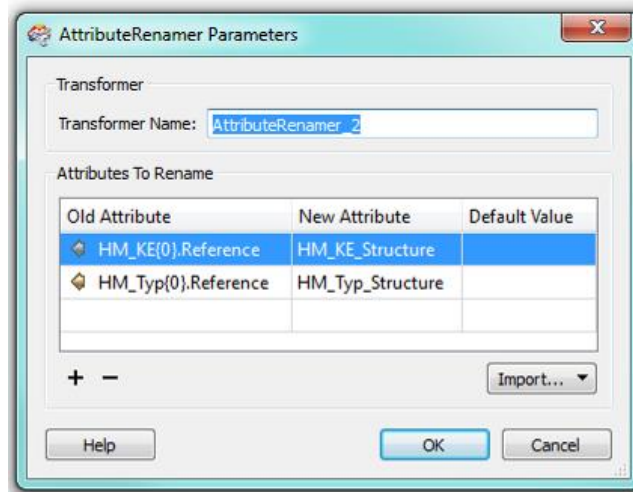
5.3.2 Speicherung in einer CSV-Datei

1. Teil 1: Lesen der Geometrie

- 1.1. Wie bereits in den vorherigen Beispielen, werden wir die 2 Listen aufsprennen, um die Geometrie zu erhalten, und dann werden wir das Ganze durch das [Attribut] *xtf_id* zusammenfassen (aggregieren). Auf diese Weise erhalten wir sämtliche Hochmoore mit ihren Geometrien (ganz gleich ob sie mehrteilig sind oder nicht).



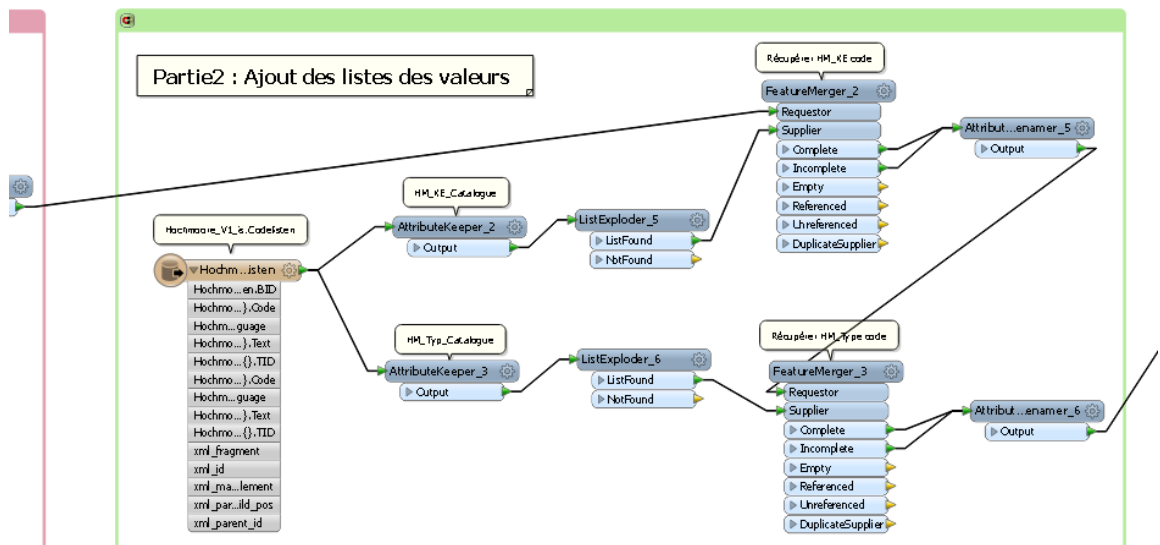
- 1.2. Mit dem letzten Transformationsprogramm AttributRenamer lassen sich die Identifikatoren der Referenzen in den Wertelisten extrahieren. So lässt sich die Nutzung des Transformationsprogramms ListExploder vermeiden.



2. Teil 2: Hinzufügung der Wertelisten

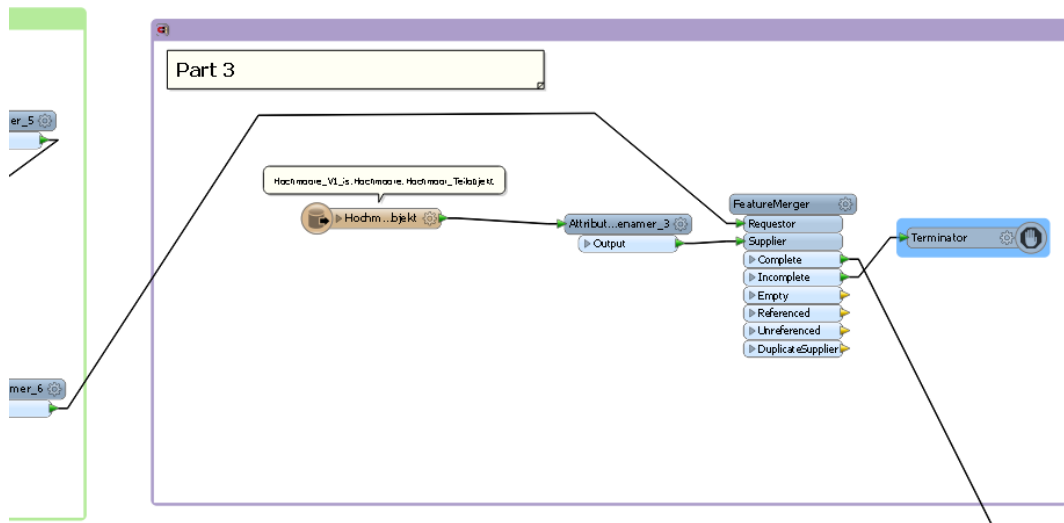
- 2.1. Wir werden nun die in der XML-Datei enthaltenen Wertelisten in den FME Workspace laden. Ziel dieser Operation ist es, die Werte der Attribute HM_KE_Structure und HM_Typ_Structure durch die in der XML-Datei enthaltenen «Übersetzungen» zu ersetzen. Hierfür extrahieren wir die Werte für jedes Attribut mit Hilfe eines AttributeKeepers und eines ListExploders.

- 2.2. Mit einem FeatureMerger fügen wir diese Werte dann schrittweise der Geometrie hinzu. Schliesslich ersetzen wir den Namen des Teils der Liste, welcher der von uns gewünschten Sprache entspricht (aus der XML-Datei stammend), durch HM_KE und HM_TYP.



3. Teil 3: Hinzufügung der Klasse *Teilobjekt*

- 3.1. Wir können die *Teilobjekte* folgendermassen der Geometrie hinzufügen:



Somit haben wir die Verknüpfung zwischen *Hochmoor_Geometrie* und *Hochmoor_Teilobjekt* geschaffen.



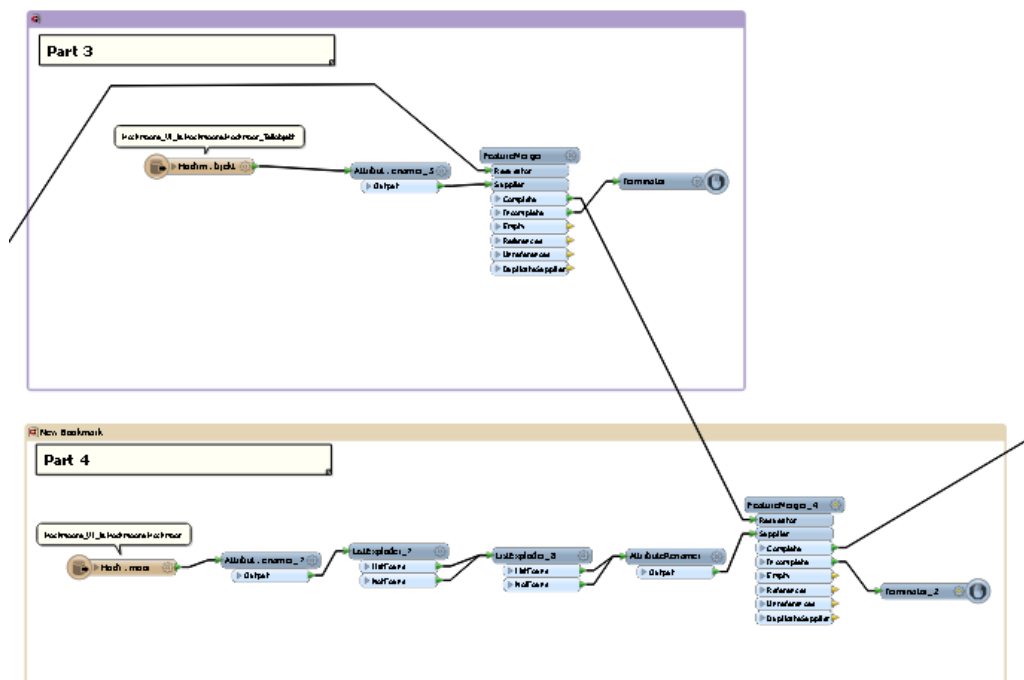
4. Teil 4: Hinzufügung der Klasse *Hochmoor*

Beim Analysieren der Struktur der Klasse *Hochmoor* bemerken wir, dass das Attribut *Mutationsgrund* auf die Klasse *LocalisationCH_V1* Bezug nimmt. Diese befindet sich im Modell *CHBase_Part2*.

```
CLASS Hochmoor =
  ObjNumber : MANDATORY TEXT;
  Name : MANDATORY TEXT*30;
  RefObjBlatt : INTERLIS.URI;
  DesignatType : DesignationType;
  IUCNCategory : MANDATORY IUCNCategory;
  Obj_GesFlaeche : MANDATORY 0.000 .. 999999.000 [Units.ha];
  Obj_GISFlaeche : MANDATORY 0.000 .. 999999.000 [Units.ha];
  Inkraftsetzungsdatum : MANDATORY INTERLIS.XMLDate;
  Mutationsgrund : INTERLIS.XMLDate;
  Mutationsgrund : LocalisationCH_V1.MultilingualMText;
END Hochmoor;
```

4.1. Somit müssen wir im FME Workspace zwei aufeinanderfolgende *ListExploder* anwenden, um die gespeicherten Werte zu erhalten. Bitte beachten Sie hier, dass dieses Attribut kein Pflichtattribut (MANDATORY) ist, und dass folglich die Wertelisten nicht unbedingt existieren müssen.

4.2. Wir können die Geometrie mit der Klasse *Hochmoor* zusammenführen, indem wir das *xtf_id* als Verknüpfungsattribut nutzen.

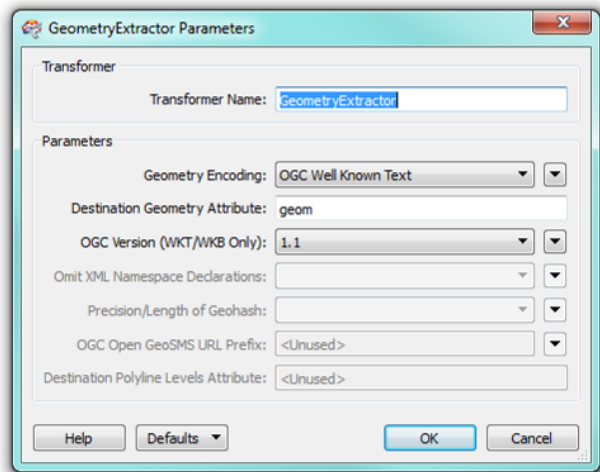


Die drei Klassen sind nun miteinander verbunden. Wir können nun mit diesen Daten arbeiten.

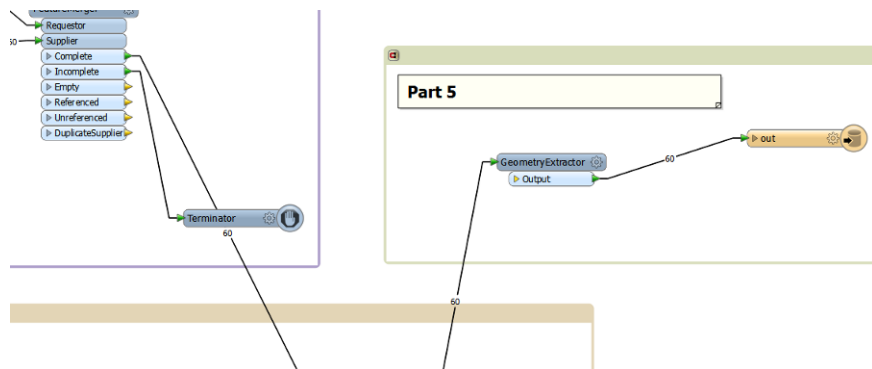
5. Teil 5: Speicherung in einer CSV-Datei

Wir werden nun diese Daten in einer CSV-Datei speichern.

5.1. Um die Geometrie speichern zu können, werden wir sie als *WKT (well-known text)* codieren.



5.2. Nun können wir einen CSV Writer hinzufügen, indem wir darin die gewünschten Attribute einfügen.



5.3.3 Speicherung in INTERLIS 2-Datei

Das Ziel dieses zweiten Teils der Übung besteht darin, die CSV-Datei wiederzuverwerten und wieder eine INTERLIS-Datei zu erstellen.

1. Teil 1: Erstellung der Liste für die Klasse *Hochmoor*

Da die Klasse *Hochmoor* die Hauptklasse des Modells ist (denn alle Objekte nehmen darauf Bezug), müssen wir alle in der CSV-Datei vorhandenen Objekte dieser Klasse zusammenfassen.

1.1. Bevor wir die Daten zusammenfassen, werden wir zunächst mit Hilfe eines *UUIDGenerators* einen eindeutigen Identifikator für jedes Objekt erstellen. Nennen wir ihn *xtf_id_object*.

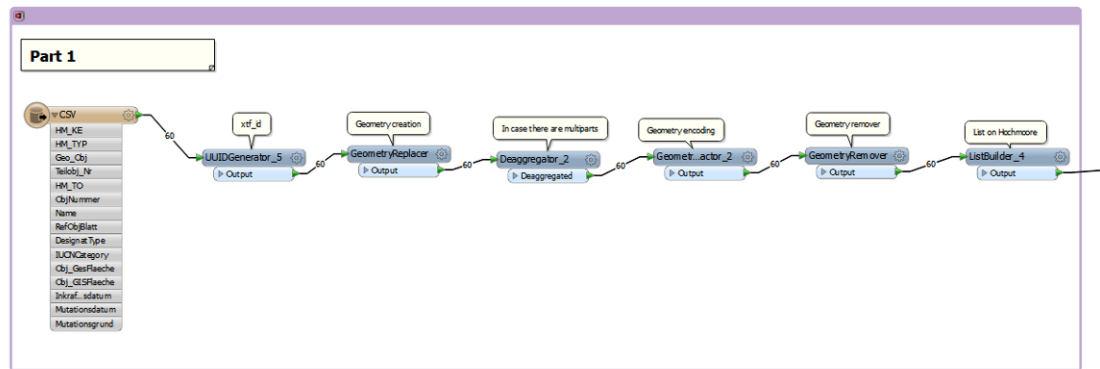
1.2. Dann können wir die Geometrie mit Hilfe eines *GeometryReplacers* wieder erzeugen, indem wir das Format FME Binary wählen.

1.3. Nun werden wir diese Geometrie zerlegen, um die mehrteiligen zu separieren.

1.4. Wir können die Geometrie erneut codieren, sodass sie sich wie ein normales Attribut behandeln lässt. Wir können wieder FME Binary wählen und speichern sie in einem neuen Attribut mit dem Namen *_geometry_object*.

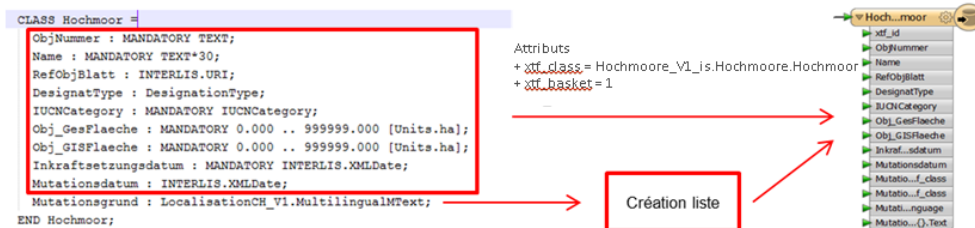
1.5. Die nicht codierte Geometrie löschen

- 1.6. Schliesslich können wir die Listen für jedes Objekt der Klasse Hochmoor erstellen. Hierfür müssen wir sie nach den in der Klasse vorhandenen Attributen gruppieren (GROUP BY). Im Rahmen dieses Beispiels gehen wir davon aus, dass es keine zwei Objekte gibt, die exakt die gleichen Attribute in dieser Klasse besitzen.

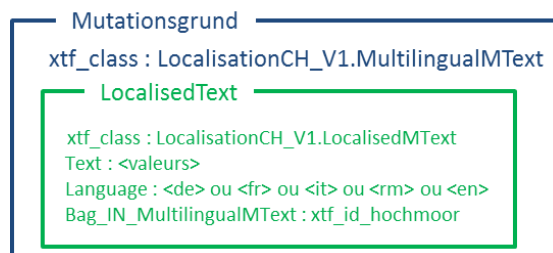


2. Teil 2: Schreiben der Klasse *Hochmoor*

Die Schwierigkeit beim Schreiben dieser Klasse ist die Beschränkung betreffend das Attribut *Mutationsgrund*.



Wir müssen die Definition der Klasse *Hochmoor*, und folglich diejenige von *CHBASE Localisation_V1* respektieren. Daher muss eine doppelte Liste zu folgendem Modell erstellt werden.



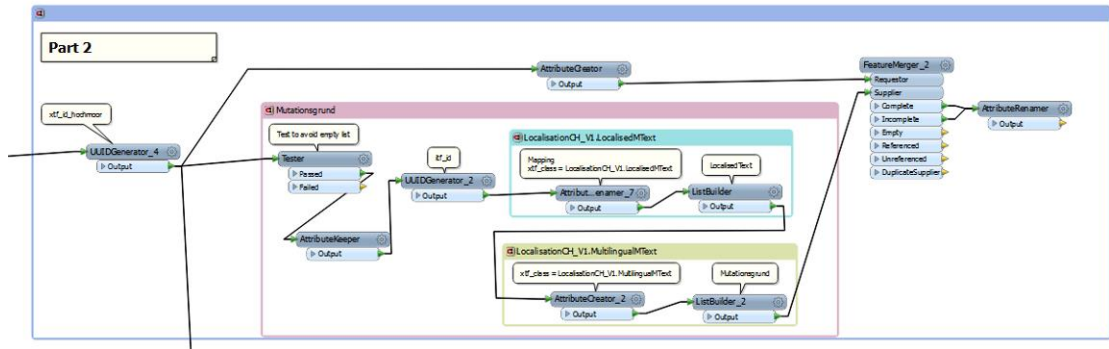
Für weitere theoretische Informationen zu dieser Typologie verweisen wir auf das Kapitel 2.1.3.

- 2.1. Bevor wir diese Listen je Objekt erstellen, müssen wir ein [Attribut] *xtf_id* erzeugen. Nennen wir es vorübergehend *xtf_id_hochmoor*. Dieses wird uns später als Verknüpfungselement zwischen den zu erstellenden Listen und den Objekten dienen.
- 2.2. Zum Schreiben der Objekte in der Klasse *Hochmoor* müssen wir *xtf_class* sowie *xtf_basket* wie zuvor erklärt definieren.
- 2.3. Die Listen können parallel erzeugt werden. Aus Gründen der Einfachheit behalten wir nur die zur Erzeugung der Listen brauchbaren Attribute bei, nämlich: Language, Mutationsgrund (das den

Text enthält) und `xtf_id_hochmoor`. Zuvor müssen wir einen Filter installieren, um Listen nur für diejenigen Attribute zu erzeugen, die Werte besitzen, um sie auszufüllen.

2.4. Dann müssen wir das [Attribut] `xtf_id` erstellen, die Attribute so umbenennen, dass sie mit dem Modell übereinstimmen, das Attribut `xtf_class` erstellen und die beiden Listen (Mutationsgrund und LocalisedText) erzeugen, indem wir sie nach dem `xtf_id_hoch_moor` gruppieren.

2.5. Mit Hilfe eines FeatureMergers können die erzeugten Listen nun mit den Objekten zusammengeführt werden - mit `xtf_id_hoch_moor` als Verknüpfungsattribut.



3. Teil 3: Geometrie - Wertetransformation entsprechend der XML-Datei

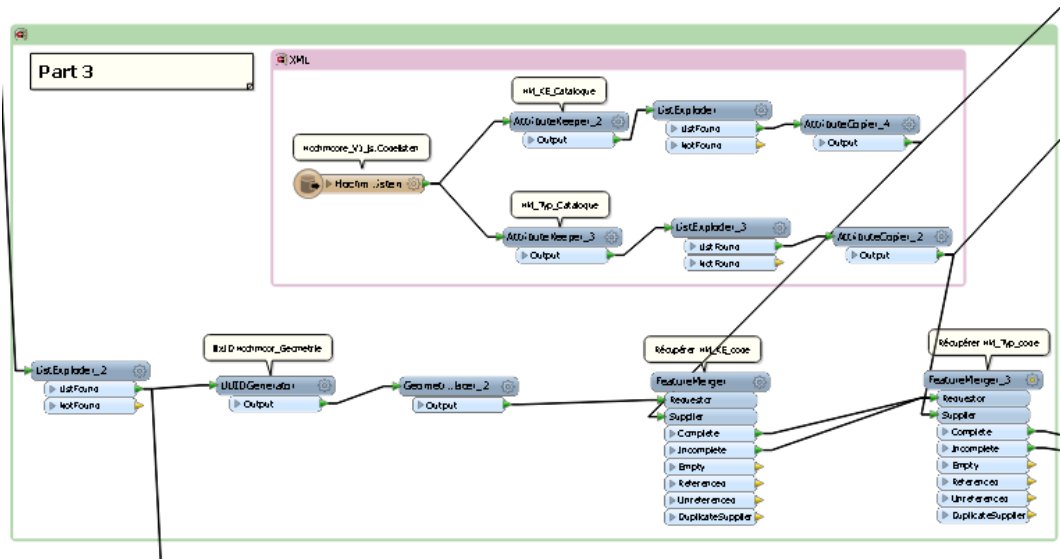
Nun können wir die Klasse *Hochmoor_Geometrie* bearbeiten. Dies erfolgt in zwei Etappen. Zuerst werden wir die Codelisten aus der XML-Datei hinzufügen, dann werden wir mit der Geometrie und ihrer Speicherung in der INTERLIS-Datei befassen.

3.1. Wir werden unsere Objekte nach der Definition des Attributs `xtf_id_hochmoor` wiederverwerten. Wir müssen die zur Bearbeitung der Klasse *Hochmoor* erstellte Liste mit Hilfe eines *ListExploders* zerlegen.

3.2. Dann können wir erneut ein [Attribut] `xtf_id` für die Geometrie erzeugen, das wir vorübergehend `xtf_id_Hochmoor_Geometrie` nennen werden.

3.3. Wir können die Geometrie wieder herstellen. Aktuell haben wir so viele Objekte wie Geometrien.

3.4. Die Codes `HM_KE_code` und `HM_Typ_code` können nun aus der XML-Datei wiedergewonnen, dann mit Hilfe von zwei aufeinander folgenden FeatureMergers in die Objekte integriert werden. Dieser dritte Teil muss nun folgendermassen aussehen:



4. Teil 4: Geometrie - Transformation der mehrteiligen

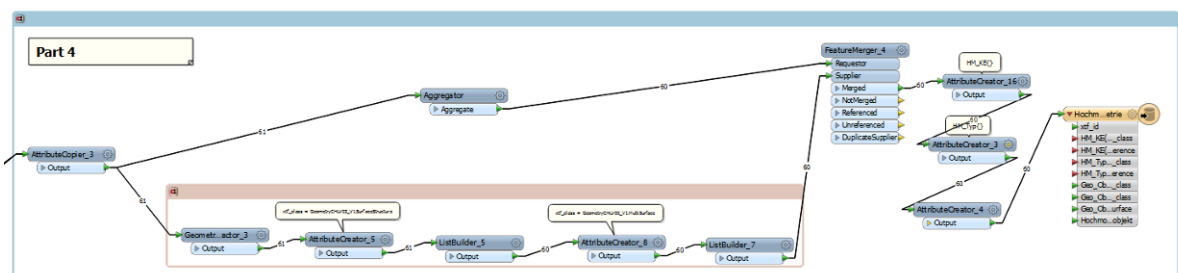
4.1. Wir können die Attribute erstellen, die für das Schreiben der Klasse Geometrie in der INTERLIS-Datei dienlich sein werden (*xtf_id*, *Hochmoor_Teilobjekt*, *Geo_Obj_Structure*, *HM_KE_Structure* et *HM_Typ_Structure*).

4.2. Wie bereits im Beispiel 5.1 geschehen, müssen wir zwei Listen erstellen, um dem Datenmodell gerecht zu werden, und so die mehrteiligen Geometrien verwalten zu können.

4.3. Das Attribut Hochmoor_Teilobjekt ist der Fremdschlüssel, der genutzt wird, um die im INTERLIS-Modell definierte Assoziation AssociationDef194 zu verwirklichen. Es wird durch das Plugin ili2fme bei der Hinzufügung des Writers automatisch hinzugefügt.

4.4. Dann werden diese Listen durch das Attribut `xtf_id_object` der Aggregation der Objekte hinzugefügt.

4.5. Zum Abschluss erstellen wir zwei neue Listen für HM_KE und HM_Typ unter Einhaltung des Datenmodells.

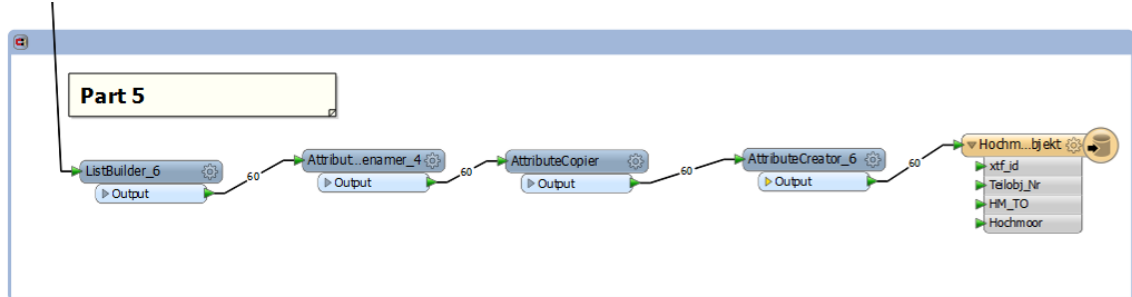


5. Teil 5: Schreiben der Klasse *Hochmoor_Teilobjekt*

5.1. Wir werden die am Anfang des Teils 3 nicht in der Liste aggregierten Objekte verwerten. Dieses Mal erstellen wir eine Liste für die Attribute der Klasse Hochmoor_Teilobjekt sowie das xtf id object.

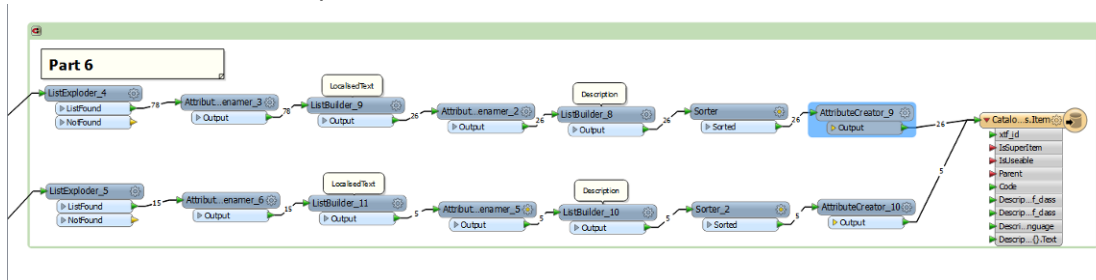
5.2. Dann erstellen wir die verschiedenen Attribute `xtf_class`, `xtf_id` und `xtf_basket`, um die Daten zu schreiben.

5.3. Das Attribut Hochmoor ist der Fremdschlüssel, der genutzt wird, um die im INTERLIS-Modell definierte Assoziation AssociationDef180 zu verwirklichen. Es wird durch das Plugin ili2fme bei der Hinzufügung des Writers automatisch hinzugefügt.



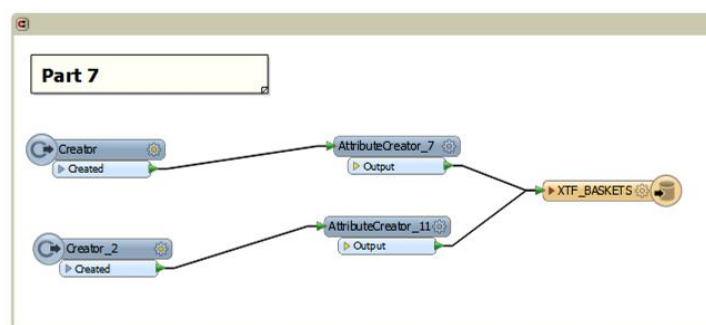
6. Teil 6: Schreiben der Klassen *HM_KE_Catalogue* et *HM_Typ_Catalogue*

6.1. Hierfür werden wir die aus der XML-Datei stammenden Daten wiederverwerten, die Listen LocalisedText daraus extrahieren, um die Attribute xtf_class hinzuzufügen. Wir erstellen also die Liste LocalisedText et Description.



7. Teil 7: XTF_BASKETS

7.1. Ähnlich wie in der Übung 5.1 müssen wir das [Attribut] xtf_basket erstellen. Diesmal erstellen wir jedoch zwei Behälter: den ersten für die Codelisten, den zweiten für den Rest der Daten.



5.4 Konvertierung INTERLIS ↔ Relationale Datenbanken

Schlüsselwörter:

ESRI file Geodatabase

PostGIS

Modellierung

CHBase

5.4.1 Darstellung des Falles

Ziel dieses Beispiels ist es, einige nützliche Hinweise zu geben, um die Konvertierung zwischen einem INTERLIS-Datensatz und einer relationalen Datenbank durchzuführen bzw. zu erleichtern. Hierzu wurden zwei Technologien ausgewählt, die in Systemen zur Verwaltung von Datenbanken Anwendung finden: ESRI file Geodatabase und PostGIS.

Das in diesem Beispiel verwendete INTERLIS-Modell ist direkt dem minimalen Modell des Katasters der belasteten Standorte auf Zivilflugplätzen (KbS BAZL) entnommen. Das vollständige Modell *KbS_LV03_V1_2* ist im Internet unter folgender Adresse verfügbar:

<http://models.geo.admin.ch/BAFU>

Für die Zwecke der Übung wurde das Basismodell so verändert, dass nur eine der Besonderheiten, die es enthält, erhalten bleibt. Das so gewonnene Modell trägt den Namen *model_54.ili*.

Ausserdem wird das zweite Modell *model_54bis.ili* generiert, das die Definition einer abstrakten Klasse enthält, sodass das Konzept der Vererbung im ersten Modell eingeführt werden kann.

Die nachstehende Tabelle beschreibt die verschiedenen Elemente der beiden Modelle.

model_54.ili	Kommentare
<pre> INTERLIS 2.3; MODEL model_54 (de) AT "http://www.inser.ch" VERSION "2015-02-17" = IMPORTS LocalisationCH_V1,GeometryCHLV03_V1, ,model_54bis; TOPIC Belastete_Standorte = DOMAIN Polygon = SURFACE WITH (STRAIGHTS) VERTEX GeometryCHLV03_V1.Coord2 WITHOUT OVERLAPS > 0.0001; UntersMassn = (UntMassn1, UntMassn2, UntMassn3, UntMassn4, UntMassn5, UntMassn6); STRUCTURE UntersMassn_ = value : MANDATORY UntersMassn; END UntersMassn_; CLASS ZustaendigkeitKataster EXTENDS model_54bis.ZustaendigkeitKatasterTopic.ZustaendigkeitKataster = Bemerkung : TEXT; END ZustaendigkeitKataster; CLASS Belasteter_Standort = Katasternummer : MANDATORY TEXT; URL_Standort : INTERLIS.URI; Geo_Lage_Polygon : Polygon; Geo_Lage_Punkt : GeometryCHLV03_V1.Coord2; InBetrieb : BOOLEAN; Nachsorge : BOOLEAN; Untersuchungsmassnahmen : BAG {1..*} OF UntersMassn_; Ersteintrag : MANDATORY INTERLIS.XMLDate; LetzteAnpassung : MANDATORY INTERLIS.XMLDate;</pre>	<p>In dem Thema <i>Belastete Standorte</i> sind zwei Bereiche definiert:</p> <ul style="list-style-type: none"> <i>Polygon</i> bezieht sich auf CHBase zur Definition der Geometrie <i>UntersMassn</i> enthält eine Werteliste <p>Die Struktur <i>UntersMassn_</i> ist mit einem Attribut <i>value</i> definiert, dessen Wert in dem Bereich <i>UntersMassn</i> enthalten sein muss.</p> <p>Diese erste Klasse erweitert die in dem Modell <i>model_54bis</i> definierte Klasse <i>ZustaendigkeitKataster</i> durch Hinzufügung des Attributs <i>Bemerkung</i>.</p> <p>Diese zweite Klasse kann zwei Geometrietypen enthalten: Polygon und Punkt.</p> <p>Das Attribut <i>Untersuchungsmassnahmen</i> ist vom Typ BAG. Dies bedeutet, dass es aus einem oder mehreren Werten bestehen kann, die nach <i>UntersMassn_</i> strukturiert sind.</p>

<pre> URL_KbS_Auszug : INTERLIS.URI; END Belasteter_Standort; ASSOCIATION ZustaendigkeitKatasterBelasteter_Standort = ZustaendigkeitKataster -- {1} ZustaendigkeitKataster; Belasteter_Standort -<- {0..*} Belasteter_Standort; END ZustaendigkeitKataster_bBelasteter_Standort; END Belastete_Standorte; TOPIC Codelisten = CLASS Untersuchungsmassnahmen_Definition = Code : MANDATORY KbS_LV03_V1_2_is.Belastete_Standorte.UntersMassn; Definition : MANDATORY LocalisationCH_V1.MultilingualText; END Untersuchungsmassnahmen_Definition; END Codelisten; END model_54. </pre>	<p>Diese Assoziation der Kardinalität 1:n definiert eine Verknüpfung zwischen den Klassen <i>ZustaendigkeitKataster</i> und <i>Belasteter_Standort</i>.</p> <p>Dieses zweite Thema enthält die Bedeutungen der Werte des Bereichs <i>UntersMassn</i> in den verschiedenen Sprachen.</p>
model_54bis.ili	Kommentare
<pre> INTERLIS 2.3; MODEL model_54bis AT "http://www.inser.ch" VERSION "10.02.2015" = IMPORTS LocalisationCH_V1,GeometryCHLV03_V1; TOPIC KatasterTopic = CLASS Kataster (ABSTRACT) = URL_Behoerde : MANDATORY INTERLIS.URI; URL_Kataster : MANDATORY INTERLIS.URI; END Kataster; END KatasterTopic; END model_54bis. </pre>	<p>Diese abstrakte Klasse definiert 2 Attribute, die in einer oder mehreren anderen Klassen eines anderen Modells erweitert werden können.</p>

5.4.2 Implementierung des konzeptionellen Modells

Die erste Etappe dieser Übung besteht darin, die relationale Datenbank ausgehend von dem konzeptionellen Modell INTERLIS 2 zu modellieren. In einem Bericht von swisstopo [5] zu dieser Thematik werden Besonderheiten der INTERLIS-Modellierung und die Möglichkeiten ihrer Implementierung in einer relationalen Datenbank dargelegt.

5.4.2.1 ESRI file GDB

Das Modell für dieses Beispiel wurde mit ArcGIS 10.2.2 erzeugt. Die Datenbank *data_54.gdb* ist in den ergänzenden Unterlagen zum vorliegenden Dokument verfügbar.

Vier Tabellen und zwei geometrische *Feature classes* werden benötigt, um das konzeptionelle Modell INTERLIS in ein ESRI File GDB Schema zu konvertieren.



Die folgende Übersicht enthält eine Beschreibung der einzelnen Tabellen:

KbS_LV03_V1_2_is.gdb	Kommentare																						
<p>Belastete_Standort:</p> <table> <tr> <th>Field Name</th><th>Data Type</th></tr> <tr> <td>OBJECTID</td><td>Object ID</td></tr> <tr> <td>ilixID</td><td>Text</td></tr> <tr> <td>Katasternummer</td><td>Text</td></tr> <tr> <td>URL_Standort</td><td>Text</td></tr> <tr> <td>InBetrieb</td><td>Text</td></tr> <tr> <td>Nachsorge</td><td>Text</td></tr> <tr> <td>Ersteintrag</td><td>Text</td></tr> <tr> <td>LetzteAnpassung</td><td>Text</td></tr> <tr> <td>URL_KbS_Auszug</td><td>Text</td></tr> <tr> <td>ZustaendigkeitKataster_FK</td><td>Text</td></tr> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Katasternummer	Text	URL_Standort	Text	InBetrieb	Text	Nachsorge	Text	Ersteintrag	Text	LetzteAnpassung	Text	URL_KbS_Auszug	Text	ZustaendigkeitKataster_FK	Text	<p>Haupttabelle, die keine Geometrie enthält. Sie enthält insbesondere einen Primärschlüssel <i>ilixID</i> sowie einen Fremdschlüssel <i>ZustaendigkeitKataster_FK</i>. Letzterer verknüpft die in dem INTERLIS-Modell definierte Assoziation <i>ZustaendigkeitKatasterBelasteter_Standort</i>.</p> <p>Die Attribute <i>InBetrieb</i> und <i>Nachsorge</i> sind mit einem Wertebereich <i>Booleschen</i> Typs verknüpft.</p> <p>Man stellt fest, dass das Attribut <i>Untersuchungsmassnahmen</i> nicht in dieser Tabelle erscheint (siehe unten).</p>
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
Katasternummer	Text																						
URL_Standort	Text																						
InBetrieb	Text																						
Nachsorge	Text																						
Ersteintrag	Text																						
LetzteAnpassung	Text																						
URL_KbS_Auszug	Text																						
ZustaendigkeitKataster_FK	Text																						
<p>Belasteter_Standort_Geo_Lage_Polygon und Belasteter_Standort_Geo_Lage_Punkt:</p> <table> <tr> <th>Field Name</th><th>Data Type</th></tr> <tr> <td>OBJECTID</td><td>Object ID</td></tr> <tr> <td>ilixID</td><td>Text</td></tr> <tr> <td>Belasteter_Standort</td><td>Text</td></tr> <tr> <td>SHAPE</td><td>Geometry</td></tr> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Belasteter_Standort	Text	SHAPE	Geometry	<p>Was die Geometrien betrifft, so lässt sich mit den ESRI File GDB nur ein Geometrietyp je <i>Feature class</i> speichern. In unserem Fall ist es also notwendig, zwei <i>Feature classes</i> zu erstellen: eine vom Typ Polygon und die andere vom Typ Punkt.</p> <p>Diese zwei Tabellen haben eine ähnliche Struktur. Sie enthalten einen Primärschlüssel <i>ilixID</i> und einen Fremdschlüssel <i>Belasteter_Standort</i>, der sich auf den Primärschlüssel der Klasse <i>Belasteter_Standort</i> bezieht.</p>												
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
Belasteter_Standort	Text																						
SHAPE	Geometry																						
<p>Rel_UntersMassn:</p> <table> <tr> <th>Field Name</th><th>Data Type</th></tr> <tr> <td>OBJECTID</td><td>Object ID</td></tr> <tr> <td>ilixID</td><td>Text</td></tr> <tr> <td>Bag_IN_Belasteter_Standort</td><td>Text</td></tr> <tr> <td>value</td><td>Short Integer</td></tr> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Bag_IN_Belasteter_Standort	Text	value	Short Integer	<p>Diese Beziehungstabelle wird benötigt, um das Attribut <i>Untersuchungsmassnahmen</i> zu berücksichtigen.</p> <p>Tatsächlich kann eine Entität der Klasse <i>Belasteter_Standort</i> mit mehreren <i>Values</i> der Struktur <i>UnterMassn_</i> verknüpft sein.</p> <p>So setzt diese Tabelle das Attribut <i>Bag_IN_Belasteter_Standort</i>, das auf den Primärschlüssel der Klasse <i>Belasteter_Standort</i> Bezug nimmt, in Beziehung zu einem oder mehreren <i>Values</i>, je nach Bereich <i>UntersMassn</i>.</p>												
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
Bag_IN_Belasteter_Standort	Text																						
value	Short Integer																						
<p>Untersuchungsmassnahmen_Definition:</p> <table> <tr> <th>Field Name</th><th>Data Type</th></tr> <tr> <td>OBJECTID</td><td>Object ID</td></tr> <tr> <td>ilixID</td><td>Text</td></tr> <tr> <td>Code</td><td>Short Integer</td></tr> <tr> <td>Text</td><td>Text</td></tr> <tr> <td>Sprache</td><td>Text</td></tr> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Code	Short Integer	Text	Text	Sprache	Text	<p>Für jeden Code des Bereichs <i>UntersMassn</i> definiert man in dieser Tabelle eine Bedeutung in jeder der Sprachen Deutsch, Französisch und Italienisch.</p>										
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
Code	Short Integer																						
Text	Text																						
Sprache	Text																						
<p>ZustaendigkeitKataster:</p> <table> <tr> <th>Field Name</th><th>Data Type</th></tr> <tr> <td>OBJECTID</td><td>Object ID</td></tr> <tr> <td>ilixID</td><td>Text</td></tr> <tr> <td>URL_Behoerde</td><td>Text</td></tr> <tr> <td>URL_Kataster</td><td>Text</td></tr> <tr> <td>Bemerkung</td><td>Text</td></tr> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	URL_Behoerde	Text	URL_Kataster	Text	Bemerkung	Text	<p>Diese Klasse ist mit der Haupttabelle über eine Assoziation der Kardinalität 1:n verknüpft. Mit anderen Worten: ein Element aus <i>ZustaendigkeitKataster</i> kann mit n Elementen aus <i>Belasteter_Standort</i> assoziiert werden. Folglich wird der Klasse <i>Belasteter_Standort</i> ein Fremdschlüssel hinzugefügt.</p>										
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
URL_Behoerde	Text																						
URL_Kataster	Text																						
Bemerkung	Text																						

5.4.2.2 PostGIS

PostgreSQL 9.3.2 und PostGIS 2.1.1 wurden verwendet, um das Modell für dieses Beispiel zu generieren. Der SQL Code zur Erzeugung des Modells ist in den ergänzenden Unterlagen zum vorliegenden Dokument verfügbar.

Die Definition des Datenmodells für die Datenbank PostgreSQL/PostGIS erfolgt ähnlich wie im Kapitel 5.4.2.1 für eine ESRI fGDB. Es werden vier Tabellen ohne Geometrien sowie eine Geometrietabelle vom Typ Polygon (*belasteter_standort_geo_lage_polygon*) und eine vom Typ Punkt (*belasteter_standort_geo_lage_punkt*) erstellt.

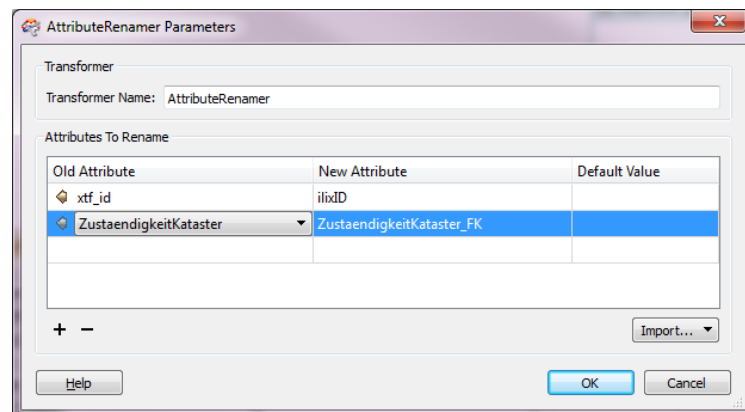


Anzumerken ist jedoch, dass PostgreSQL die Verwendung von Kleinbuchstaben in den Klassen- und Attributnamen vorschreibt.

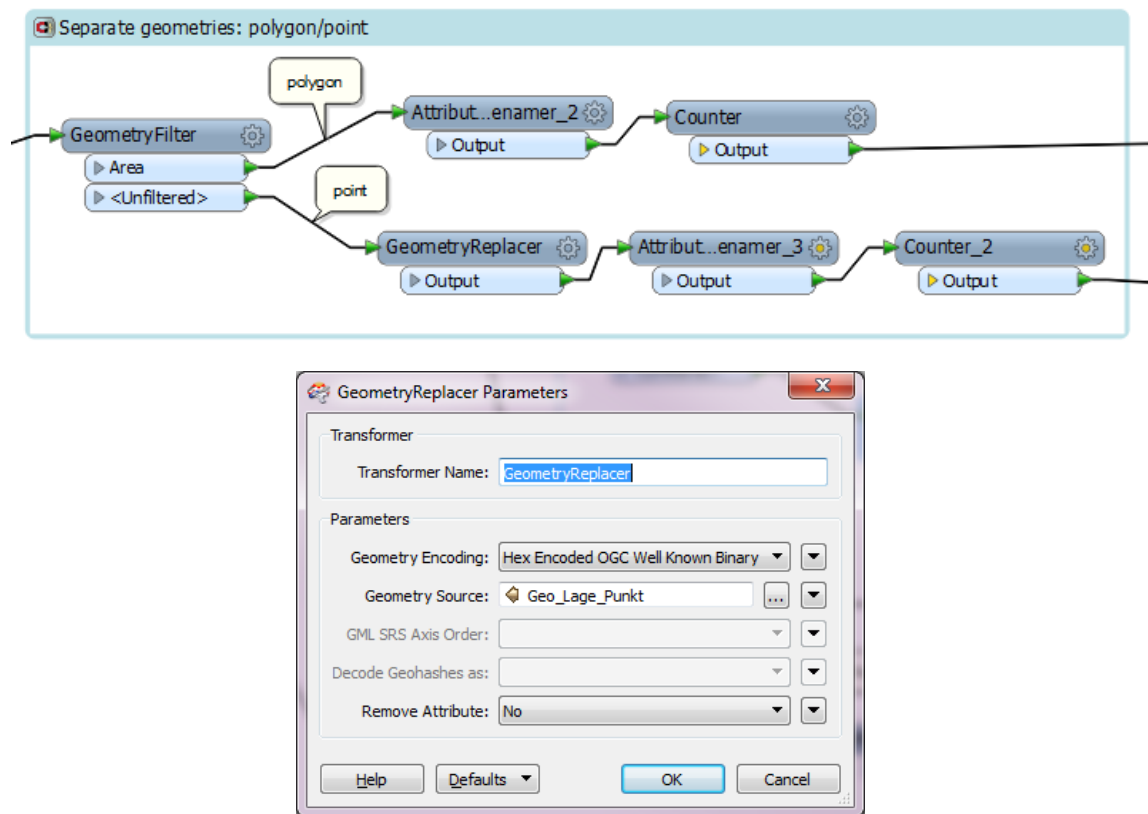
5.4.3 Transformation INTERLIS → fGDB

Dieses Kapitel nimmt Bezug auf den Workspace *w-Doc_ili2fme_Ex4_ili2fGDB-isb.fmw*.

1. In der Featureklasse *Belasteter_Standort* des INTERLIS Readers sieht man, dass das Attribut *ZustaendigkeitKataster* automatisch durch das Plugin *ili2fme* hinzugefügt wurde. Dieses Attribut ist der Fremdschlüssel, der die Assoziation *ZustaendigkeitKatasterBelasteter_Standort* verwirklicht. Dieses Attribut muss umbenannt werden in *ZustaendigkeitKatasterBelasteter_Standort_FK*, um mit dem in der Ausgabe-Datenbank definierten Attribut übereinzustimmen.

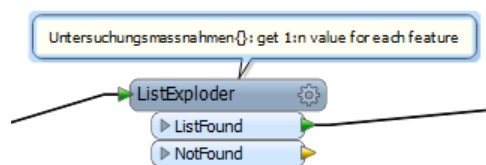


2. Ausgehend von der Featureklasse *Belasteter_Standort* des INTERLIS Readers haben wir zuvor bereits gesehen, dass die Geometrien separiert werden müssen, um in zwei ESRI Featureklassen geladen zu werden, je nachdem, ob die Geometrie vom Typ Punkt oder Polygon ist.



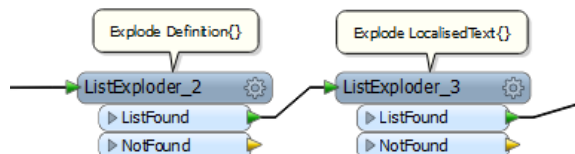
Mit dem *GeometryFilter* lässt sich diese Trennung vornehmen. Der *GeometryReplacer* ermöglicht anschliessend die Bewahrung der Geometrie vom Typ Punkt, die im Attribut *Geo_Lage_Punkt* gespeichert ist.

- Wir haben gesehen, dass jede Entität der Klasse *Belasteter_Standort* mehrere Werte für das Attribut *Untersuchungsmassnahmen* haben kann. Um sämtliche Werte zu bewahren, muss man die im Reader durch das Plugin *ili2fme* erstellte Liste *Untersuchungsmassnahmen* aufsprengen:



So erhält man die Beziehungen, die es ermöglichen, die Tabelle *Rel_UntersMassn* auszufüllen.

- Schliesslich wahrt der Prozess die verschiedenen Bedeutungen der Codes je nach Sprache, die ebenfalls durch das Plugin *ili2fme* in Listen der Featureklasse *Untersuchungsmassnahmen_Definition* gespeichert wurden.



Der Name der Listen (*Definition* und *LocalisedText*), die durch *ili2fme* erstellt werden, entstammen direkt der Struktur, die in den INTERLIS-Modellen definiert ist. So findet man in *KbS_LV03_V1_2_is.ili*:

```
CLASS Untersuchungsmassnahmen_Definition =
  Code : MANDATORY KbS_LV03_V1_2_is.Belastete_Standorte.UntersMassn;
  Definition : MANDATORY LocalisationCH_V1.MultilingualText;
END Untersuchungsmassnahmen_Definition;
```

Und wenn man dann in die Struktur MultilingualText im Modell LocalisationCH_V1 schaut:

```
STRUCTURE LocalisedText =
  Language: LanguageCode_ISO639_1;
  Text: MANDATORY TEXT;
END LocalisedText;

STRUCTURE MultilingualText =
  LocalisedText : BAG {1..*} OF LocalisedText;
  UNIQUE (LOCAL) LocalisedText:Language;
END MultilingualText;
```

Das von swisstopo publizierte Benutzerhandbuch [10] enthält eine detaillierte Erläuterung der Art und Weise, die minimalen Modelle in eine INTERLIS-Modellierung zu integrieren.

5.4.4 Transformation INTERLIS → PostGIS

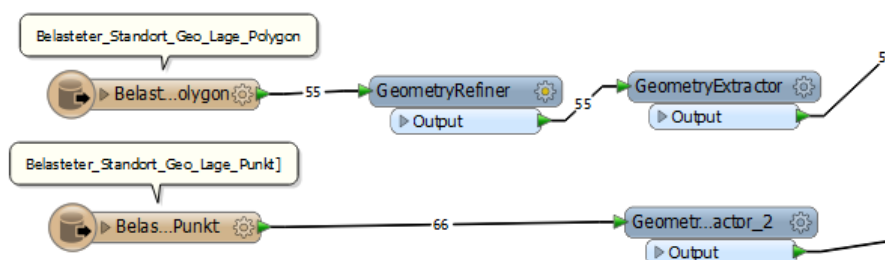
Dieses Kapitel nimmt Bezug auf den Workspace *w-Doc_ili2fme_Ex4_ili2PostGIS-isa.fmw*.

Die Vorgehensweise beim Transfer in eine PostGIS Datenbank ist der oben beschriebenen sehr ähnlich. Nur die Writer ändern sich. Daher wird der Workspace in dem vorliegenden Bericht nicht beschrieben. Dies bleibt dem Leser überlassen.

5.4.5 Transformation fGDB → INTERLIS

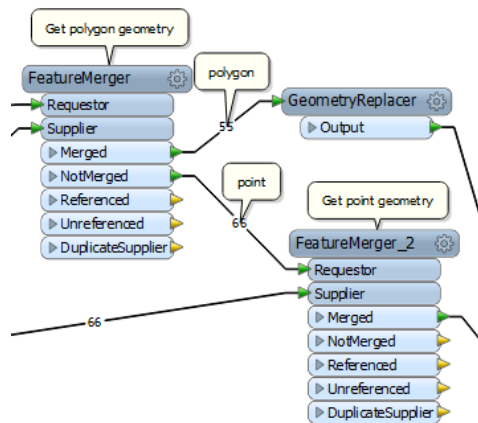
Dieses Kapitel nimmt Bezug auf den Workspace *w-Doc_ili2fme_Ex4_fGDB2ili-isb.fmw*.

1. Ausgehend von den zwei Geometrie-Featureklassen der fGDB (*Belasteter_Standort_Geo_Lage_Polygon* und *Belasteter_Standort_Geo_Lage_Punkt*) beginnt der Prozess mit der Speicherung der Geometrien in Attributen mit Hilfe von zwei *GeometryExtractors*.

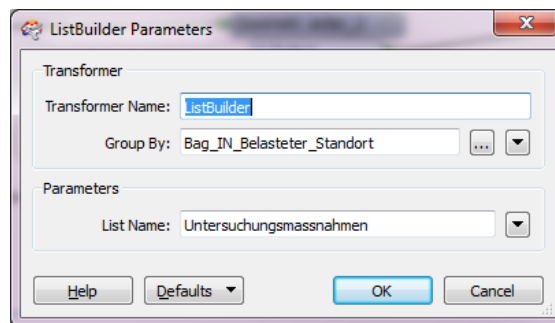


In diesem Beispiel verwenden wir den Codierungstyp *FME_XML*. Man sieht auch, dass das Transformationsprogramm *GeometryRefiner* hinzugefügt wurde. Es ermöglicht eine Vereinfachung der Definition der Geometrie, wenn diese für spätere Bearbeitungsschritte problematisch sein kann.

2. Diese Attribute, welche die Geometrie enthalten (*Geo_Lage_Polygon* und *Geo_Lage_Punkt*), müssen anschliessend dem Objekt zugeordnet werden, auf das sie sich in der Hauptklasse *Belasteter_Standort* beziehen. Die beiden Transformationsprogramme *FeatureMerger* und *FeatureMerger_2* führen diese Operation aus.



Ausserdem muss die Liste *Untersuchungsmassnahmen* ausgehend von der Beziehungstabelle *Rel_UntersMassn* rekonstruiert werden. Hierbei ist darauf zu achten, dass zuvor das Attribut *xtf_class* korrekt eingetragen wurde. Mit einem *ListBuilder* lässt sich diese Liste rekonstruieren. Man führt ein *GroupBy* nach dem Fremdschlüssel durch, der auf den Primärschlüssel der Hauptklasse *Belasteter_Standort* Bezug nimmt. So erhält man eine Liste für jede Entität dieser Klasse. Mit einem *FeatureMerger* kann die Liste dann mit der jeweiligen Entität verknüpft werden.



- Schliesslich müssen auch die Listen rekonstruiert werden, damit sie mit der für die Klasse *Untersuchungsmassnahmen_Definition* definierten Struktur übereinstimmen. Indem man genau umgekehrt vorgeht wie im Punkt 3 des Kapitels 5.4.3, beginnt man mit der Rekonstruktion der Liste *LocalisedText*, die für jeden *Code* die Bedeutungen in den drei Sprachen enthält. Dann geht man in der Struktur weiter, indem man die Liste *Definition* erstellt, die in der Klasse *Untersuchungsmassnahmen_Definition* enthalten ist. In dieser Form kann das Plugin *ili2fme* korrekt in die XML-Datei schreiben. Nachstehend sieht man einen Auszug aus der *.xtf*-Datei, die durch den Prozess geschrieben wurde.

```
<KbS_LV03_V1_2_is.Codelisten.Untersuchungsmassnahmen_Definition TID="cl20023um5">
  <Code>UntMassn5</Code>
  <Definition>
    <LocalisationCH_V1.MultilingualText>
      <LocalisedText>
        <LocalisationCH_V1.LocalisedText>
          <Language>de</Language>
          <Text>Überwachung</Text>
        </LocalisationCH_V1.LocalisedText>
        <LocalisationCH_V1.LocalisedText>
          <Language>fr</Language>
          <Text>Surveillance</Text>
        </LocalisationCH_V1.LocalisedText>
        <LocalisationCH_V1.LocalisedText>
          <Language>it</Language>
          <Text>Sorveglianza</Text>
        </LocalisationCH_V1.LocalisedText>
      </LocalisedText>
    </LocalisationCH_V1.MultilingualText>
  </Definition>
</KbS_LV03_V1_2_is.Codelisten.Untersuchungsmassnahmen_Definition>
```

Man erkennt, dass der Aufbau in FME «aus der Mitte» erfolgt. Man beginnt damit, die Listen *LocalisedText*{ in rot, und anschliessend *Definition*{ in grün zu erstellen. Bei jedem Schritt ist es wichtig das Attribut *xtf_class* einzutragen, das die Verknüpfung mit dem konzeptionellen Modell ermöglicht.

5.4.6 Transformation INTERLIS →PostGIS

Dieses Kapitel nimmt Bezug auf den Workspace *w-Doc_ili2fme_Ex4_PostGIS2ili-isa.fmw*.

Die Vorgehensweise beim Schreiben in INTERLIS aus einer PostGIS Datenbank ist der oben beschriebenen sehr ähnlich. Nur die Writer ändern sich. Daher wird der Workspace in dem vorliegenden Bericht nicht beschrieben. Dies bleibt dem Leser überlassen.

6. FAQ

1. Was sind BASKETS? Und wie sollen sie genutzt werden?
Hierbei handelt es sich um Behälter, die genutzt werden, um die INTERLIS 2 XML-Datei zu strukturieren. Wenn man in INTERLIS 2 schreibt, muss man also einen BASKET je TOPIC erstellen. Dann muss jeder Featuretyp mit Hilfe des Attributs xtf_basket mit einem BASKET verknüpft werden.
2. Ist es möglich, ein INTERLIS 1-Modell in INTERLIS 2 zu konvertieren? Und Daten?
Ja, es gibt das von der infoGrips GmbH entwickelte Werkzeug ITF2XML, mit dem man eine XML-Datei sowie das Modell in INTERLIS 2 ausgehend von einer ITF-Datei und einem Modell in INTERLIS 1 erzeugen kann.
<http://www.interlis.ch/general/itf2xml.php?language=d>
3. Existieren in INTERLIS 2 Einschränkungen in Bezug auf Attributnamen oder Tabellennamen?
INTERLIS 2 unterstützt nur Namen, die keine Symbole enthalten. Ausserdem dürfen bestimmte reservierte Namen (z.B. BOOLEAN, END, IN, NAME, MODEL, NULL, OID, PARAMETER, THIS, SURFACE) nicht verwendet werden. Eine erschöpfende Liste der in INTERLIS 2 reservierten Namen findet sich im Kapitel 2.2.7 des Referenzhandbuchs für INTERLIS 2 [1].
4. Welche Geometrien werden in INTERLIS 2 unterstützt?
In INTERLIS 2 existieren folgende Geometrietypen:
 - *Punkte, die zwei- oder dreidimensional definiert sein können:*

```
COORD
480000.000 .. 840000.000 [m],
7000.000 .. 300000.000 [m],
200.000 .. 5000.000 [m];
```
 - *Linien werden als POLYLINE definiert; mit dem Schlüsselwort ARCS können bogenförmige Linien zugelassen werden:*

```
POLYLINE WITH (STRAIGHTS, ARCS) VERTEX Coorddef;
```
 - *Polygone, können vom Typ SURFACE oder AREA sein; im Gegensatz zu AREA erlaubt der Typ SURFACE Objektzusammenlegungen innerhalb einer Klasse.*

```
SURFACE WITH (STRAIGHTS, ARCS) VERTEX Coorddef
WITHOUT OVERLAPS > 0.0001;

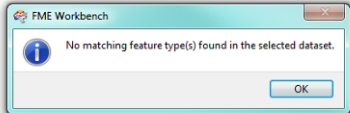
AREA WITH (STRAIGHTS, ARCS) VERTEX Coorddef
WITHOUT OVERLAPS > 0.0001;
```

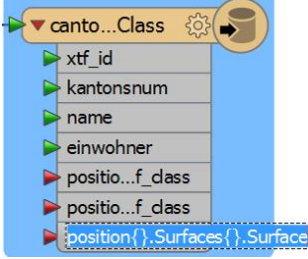
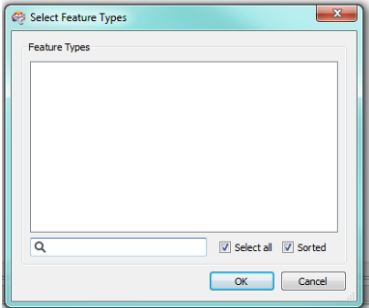
*Mit dem Schlüsselwort ARCS können auch Bögen zugelassen werden.
Die Erweiterung WITHOUT_OVERLAPS > «Toleranz» ermöglicht es, eine maximale Toleranz für die Zusammenlegung innerhalb einer Klasse zu definieren. Ab INTERLIS 2.3 ist die Definition eines von Null verschiedenen Toleranzwerts für diese Geometrietypen zwingend erforderlich.*
5. Wie werden mehrteilige Geometrien verwaltet?
Mehrteilige Geometrien werden in INTERLIS nicht nativ unterstützt. Um dieser Beschränkung zu entgehen, wird in den CHBase-Modellierungsbausteinen ein mehrteiliger Geometrietyp definiert, der eine Aggregation grundlegender Geometrietypen ist.

6. Wie kann man sich vergewissern, dass während einer Bearbeitung kein Polygon verloren gegangen ist?
Ganz allgemein ist es zunächst ratsam, die Warnmeldungen im Protokoll (Log) der FME-Bearbeitung zu überprüfen.
Ausserdem ist es möglich, die Anzahl der auftretenden Fälle, die in einer bestimmten Klasse der xtf-Datei existieren (durch Öffnen der Datei mit einem Texteditor) mit der Anzahl der in FME erstellten Polygone zu vergleichen.
7. Kann die Geometrie der Objekte bei einer FME-Bearbeitung geändert werden?
Betreffend das Plugin ili2fme: nein, die INTERLIS 2 Reader und Writer verändern die Geometrie der Objekte nicht.
Wenn durch eine Bearbeitung INTERLIS-Daten in ein anderes Format konvertiert werden, ist es möglich, dass Beschränkungen hinsichtlich des Schreibformats (Genauigkeit, Auflösung, Toleranz, ...) die ursprünglichen Geometrien beeinträchtigen.
8. Ermöglicht INTERLIS die Definition topologischer Regeln?
Im INTERLIS 2-Modell lassen sich Regeln definieren, um Zusammenlegungen über einer bestimmten Toleranz für Objekte vom Typ SURFACE auszuschliessen:

```
SURFACE WITHOUT OVERLAPS > 0.002;
```

Das Plugin ili2fme überprüft nicht, ob die Daten beim Lesen oder Schreiben mit dieser Definition übereinstimmen. Ein INTERLIS Checker erfüllt jedoch diese Funktion.
9. Wenn ein INTERLIS-Modell auf andere Modelle (Units, CoordSys, Time, Symbology, CHBase) zurückgreift, wie lässt sich gewährleisten, dass diese Modelle zugänglich sind?
Hier sind zwei Methoden zu unterscheiden:
 - *Online: wenn eine Internet-Verbindung verfügbar ist und keine Sicherheitsauflage dies verhindert, sind alle minimalen Modelle unter folgender Adresse zugänglich:*
models.interlis.ch.
 - *Lokal: die minimalen Modelle werden in besonderen Verzeichnissen gespeichert.*
Mit dem Parameter «Models directory» des Plugins ili2fme lässt sich der Platz der minimalen Modelle definieren (vgl. Kapitel 3.3).
10. Exemplarische Probleme, die mit ili2fme häufig angetroffen werden:

Problem	Ursache	Lösung
<i>Model(s) not found</i> oder: 	<ul style="list-style-type: none"> - Nichtübereinstimmung zwischen den in der XTF-Datendatei eingetragenen Modellen (Kopf des Dokuments) und dem Namen der Referenzmodelle (Namens-, Versionsänderung) - Modelle in den Referenzverzeichnissen nicht vorhanden 	<ul style="list-style-type: none"> - Die Namen der Modelle überprüfen, und zwar durch Öffnen der XTF- und ILI-Dateien in einem Texteditor - Die Zugriffspfade in <i>Parameters</i> von FME ändern (vgl. Kapitel 3.3)
Datentyp nicht eingehalten	<ul style="list-style-type: none"> - Der in FME entwickelte Prozess zur Erzeugung der XTF-Datendatei hat diese Einschränkungen nicht berücksichtigt. Achtung, das Plugin ili2fme 	<ul style="list-style-type: none"> - Transformationen im FME Workspace vornehmen, um dem Modell gerecht zu werden. - die Datendatei mit dem infoGrips Checker überprüfen

	schreibt die Daten, ohne ihre Übereinstimmung mit dem Modell zu überprüfen.	
Pflichtdaten nicht gespeichert	<ul style="list-style-type: none"> - Manche Pflichtattribute (MANDATORY) wurden im Workspace nicht eingetragen. Achtung, das Plugin <i>ili2fme</i> schreibt die Daten, ohne Überprüfung. 	<ul style="list-style-type: none"> - Diese Attribute in dem FME Workspace erstellen, um dem Modell gerecht zu werden. - Die Datendatei mit dem infoGrips Checker überprüfen.
Nichteinhaltung der Klassenstruktur in den FME-Listen 	<ul style="list-style-type: none"> - der Workspace muss die schrittweise Erstellung von Listen beinhalten, um den Klassen zu entsprechen. Ist dies nicht der Fall, werden die Daten nicht eingetragen. Achtung, es geschieht mitunter, dass manche Attribute des Writers rot bleiben, obwohl die Struktur der Daten korrekt hergestellt wurde. In diesem Fall werden die Daten korrekt gespeichert. 	<ul style="list-style-type: none"> - die Struktur des Modells einhalten, indem man eine Liste je Klasse erstellt, und <i>xtf_id</i>, <i>xtf_class</i> sowie die übrigen konstituierenden Attribute der Klasse darin einfügt. Dabei mit der Klasse beginnen, die das höchste Niveau hat, um sie in die folgenden Klassen einzuschliessen. Im vorliegenden Beispiel muss Surface als erstes erstellt werden, und dann in Position eingeschlossen werden.
<i>Translation was SUCCESSFUL</i> aber die XTF-Datei ist leer	<ul style="list-style-type: none"> - Es sind viele Ursachen denkbar (Auflistung ohne Anspruch auf Vollständigkeit): - Keine Erstellung der Attribute <i>xtf_basket</i> je Klasse - Keine Erstellung der Klasse <i>XTF_BASKETS</i> - Falsche Definition des Attributs <i>xtf_basket</i> 	<ul style="list-style-type: none"> - Diese Attribute oder Klasse erstellen.
Geometrieproblem <i>@Geometry function could not parse ...</i>	<ul style="list-style-type: none"> - Die Codierung der Geometrie mit Hilfe eines <i>GeometryExtractor</i> entspricht nicht der in den Parametern des Writers spezifizierten Codierung. 	<ul style="list-style-type: none"> - Die beiden Codierungstypen in Übereinstimmung bringen.
	Beim Import eines Datenmodells ist das Fenster zur Auswahl der Klassen leer. Das Problem entsteht durch einen Fehler im <i>ili</i> -Datenmodell.	<ul style="list-style-type: none"> - Das Modell korrigieren.