



Schweizerische Eidgenossenschaft  
Confédération suisse  
Confederazione Svizzera  
Confederaziun svizra

Département fédéral de la défense,  
de la protection de la population et des sports DDPS

**Office fédéral de topographie swisstopo**

# Manuel de l'utilisateur *ili2fme*

Date de création : mai 2015

Mandant :  
COSIG, Office fédéral de topographie swisstopo

Auteur :



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

## Table des matières

<b>SOURCES, DOCUMENTS.....</b>	<b>4</b>
<b>1. INTRODUCTION.....</b>	<b>5</b>
<b>2. PRÉREQUIS THÉORIQUES NÉCESSAIRES.....</b>	<b>6</b>
2.1 INTERLIS .....	6
2.2 FME.....	9
<b>3. INFORMATIONS GÉNÉRALES - <i>ILI2FME</i>.....</b>	<b>12</b>
3.1 Prérequis logiciels matériels.....	12
3.2 Concernant les versions.....	12
3.3 Modèle Interlis et fichiers de transfert .....	12
3.4 Description des paramètres.....	13
3.5 Encodage de la géométrie.....	14
3.6 Attributs de format.....	15
<b>4. DÉMARRAGE RAPIDE.....</b>	<b>16</b>
4.1 Installation .....	16
4.2 Conversion rapide.....	16
4.3 Visualisation rapide.....	17
<b>5. CAS D'UTILISATION.....</b>	<b>18</b>
5.1 Écriture d'un fichier INTERLIS 2.....	18
5.2 Lecture d'un fichier INTERLIS 2 .....	28
5.3 Lecture et écriture de fichiers INTERLIS 2 avec structures imbriquées .....	31
5.4 Conversion INTERLIS $\leftrightarrow$ bases de données relationnelles .....	41
<b>6. FAQ.....</b>	<b>49</b>

## SOURCES, DOCUMENTS

[1] COSIG, INTERLIS 2 – Manuel de référence, 2006-04-13

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

[2] COSIG, Recommandations générales portant sur la méthode de définition des « modèles de géodonnées minimaux », 2011-09-12

<http://www.geo.admin.ch/internet/geoportal/fr/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, Exemples de meilleures pratiques pour la transposition de modèles de géodonnées conceptuels, 2014

<http://www.geo.admin.ch/internet/geoportal/fr/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] Répertoire de stockage des modèles de géodonnées minimaux de la confédération

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

[9] Outils INTERLIS

[http://www.interlis.ch/interlis2/download23\\_f.php#outils](http://www.interlis.ch/interlis2/download23_f.php#outils)

[10] swisstopo, Modules de base de la Confédération pour les modèles de géodonnées minimaux, 2011-08-30

<http://www.geo.admin.ch/internet/geoportal/fr/home/topics/geobasedata/models.html>

## 1. INTRODUCTION

Ce document a pour objectif d'expliquer et d'illustrer l'utilisation du plugin *ili2fme* dans le logiciel FME. La première partie fournit au lecteur des notions théoriques et techniques utiles, voire indispensables, pour permettre la visualisation, la modification et la création de fichiers INTERLIS. La seconde partie illustre, grâce à quatre workspaces FME, des cas d'utilisation pratique. Les deux premiers exemples sont relativement simples, destinés à introduire l'utilisateur aux principes de la modélisation de fichiers INTERLIS sur FME. Ils sont accompagnés d'une marche à suivre complète. Les deux exemples suivants sont des cas concrets d'utilisation plus compliquée. Ce document aborde leurs principes généraux, mais les décrit de façon moins détaillée. Le but étant de mettre en évidence d'autres particularités propres aux modèles INTERLIS dans FME. En guise de conclusion, certaines problématiques sont abordées sous forme de questions-réponses.

Il est conseillé pour la compréhension de ce document de disposer du programme FME (version 2014 au moins) afin d'effectuer les opérations décrites en parallèle de la lecture.

## 2. PRÉREQUIS THÉORIQUES NÉCESSAIRES

Ce chapitre introduit quelques descriptions d'éléments théoriques nécessaires à la compréhension de la suite du rapport.

### 2.1 INTERLIS

Tous les modèles de géodonnées minimaux nouvellement mis en place sont établis en INTERLIS 2. De ce fait, le format INTERLIS 1 n'est pas traité dans ce rapport, bien qu'il soit aussi supporté par le plugin *ili2fme*.

#### 2.1.1 Présentation générale

INTERLIS est le format d'échange de données pour les géodonnées de la Confédération. Il a été défini pour faciliter l'échange de données entre plusieurs utilisateurs d'outils SIG, selon un modèle de données prédéfini. C'est un format ouvert, qui se base sur des données de type textuel.

Un jeu de données INTERLIS se compose de deux fichiers :

- Le modèle de données .ILI  
Contient une description du schéma de données.
- Les données .ITF (INTERLIS 1) / .XTF (INTERLIS 2)  
Contiennent les données elles-mêmes, structurées selon le modèle correspondant.
- Les catalogues de données (XML/XTF)  
Contient des listes de valeurs ainsi que, le cas échéant, leurs significations dans différentes langues.

#### 2.1.2 Modèles, thèmes, classes

La hiérarchie du modèle conceptuel INTERLIS 2 est composée de 3 niveaux :

- Le modèle (« MODEL ») contient l'ensemble des éléments constitutifs de la réalité à décrire. Il est caractérisé notamment par un nom univoque et une version.
- Le thème (« TOPIC ») contient les définitions qui visent à décrire une partie objective précise de la réalité.
- Le troisième niveau est constitué d'une série d'éléments définissant le cœur du schéma conceptuel. Nous citons ici ceux qui sont les plus fréquemment utilisés :
  - les classes (« CLASS ») : permettent de déclarer les propriétés de tous les objets qui lui appartiennent, notamment la définition du nom et du type d'attributs.
  - les domaines de valeurs (« DOMAIN ») : sont utilisés pour contraindre la valeur d'un attribut selon une liste ou une certaine étendue.
  - les structures (« STRUCTURE ») sont constituées d'une liste d'attributs (noms et types). La structure peut ensuite être assignée à un ou plusieurs attributs appartenant à une classe du même modèle.
  - les classes d'association (« ASSOCIATION ») : permettent de définir les relations qui peuvent exister entre différentes classes d'un thème. INTERLIS permet de définir deux types de relations : incorporée ou non (Réf. [1], chap. 3.3.9).

Référence : Manuel de référence INTERLIS 2 [1], chap. 1.3, 1.4, 2.5

### 2.1.3 Mots-clés et définitions

La signification des mots réservés suivants sont à connaître car ils sont utilisés dans les exemples du chapitre 5.

- **IMPORTS** : si un élément se rapporte à une définition réalisée dans un autre modèle, ce dernier est à importer avec cette commande.

```
IMPORTS LocalisationCH_V1,GeometryCHLV03_V1;
```

- **EXTENDS** : l'héritage permet d'assigner à un objet les propriétés qui ont été assignées préalablement à un autre objet. Par exemple, une classe peut hériter des attributs qui ont été assignés à une autre classe.

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

- **LIST OF / BAG OF** : sont utilisés lorsque les valeurs d'un attribut structuré peut comprendre plusieurs éléments structurés, respectivement de manière ordonnée et non-ordonnée.

```
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;
```

Dans cet exemple, l'attribut *Attribut 1* de la classe A peut contenir de 1 à n valeurs, dont la structure est celle définie dans *LocalisedText*.

Référence : Manuel de référence INTERLIS 2 [1], chap. 2.1, 2.2, 2.4, 2.6

### 2.1.4 CH-Base

Dans le cadre de la coordination pour la mise en place des modèles minimaux de géodonnées (MGDM), la Confédération a défini des modules de base qui définissent des aspects généraux de modélisation. Ces éléments peuvent être importés dans un MGDM si besoin.

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

L'extrait ci-dessus décrit de quelle manière les modules de base sont importés dans la définition d'un modèle. Ainsi, le domaine prédéfini *CHBaseDomain*, qui se trouve dans le modèle *CHBaseModule*, est utilisé comme propriété de l'attribut *MyAttr*.

#### 2.1.4.1 Exemple

Dans l'exemple suivant, on trouve trois références à des objets CH-Base :

- L'attribut *Mutationsgrund\_Text* est un objet textuel multilingue.
- L'attribut *Kanton* est lié à un domaine de valeur contenant la liste formalisée de tous les cantons.
- La géométrie est de type surface multi-parties.

```
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 Objets principaux

De manière générale, les objets CH-Base suivants sont fréquemment utilisés :

- *MultiSurface* : géométrie multi-parties
- *MultilingualText* : objet textuel multilingue
- *Codelisten* : domaine de valeurs, incluant une série de codes et leurs significations
- *ModInfo* : historique des modifications apportées aux données

Ces modèles minimaux sont accessibles via le site de swisstopo à l'adresse :

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

Référence :

- *Recommandations générales portant sur la méthode de définition des « modèles de géodonnées minimaux »* [2], chap 2 et 3
- *Modules de base de la Confédération pour les modèles de géodonnées minimaux* [10]

### 2.1.5 Géométrie multipart vs multiple

Afin d'éviter les éventuelles confusions entre une géométrie dite « *multipart* » et des géométries multiples, il est important de distinguer ces deux termes.

Une géométrie multipart se dit d'un objet pouvant contenir plusieurs géométries de type identique. Comme nous le verrons dans l'exemple 5.1, un canton peut être composé de plusieurs polygones. C'est notamment le cas pour le canton de Vaud qui est formé d'un grand polygone et d'une enclave dans le canton de Fribourg. Le fichier de donnée INTERLIS stocke ces deux géométries dans le même attribut.

La classe du modèle comprendra donc une seule ligne.



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

Une classe à géométrie multiple consiste à stocker dans la même classe plusieurs géométries distinctes. Par exemple, une classe avalanche peut contenir une surface pour définir la zone de départ, une seconde surface pour la zone d'écoulement et une dernière surface pour sa zone d'arrêt. De plus, la ligne de rupture peut être définie à l'aide d'une géométrie de type ligne.

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

Ainsi, contrairement à une géométrie multipart, chaque géométrie multiple a une signification propre. A noter qu'il est possible d'avoir une classe de géométrie multiple composée de géométrie multipart.

### 2.1.6 Basket

L'utilisation des données de manière sélective est possible grâce à la notion de Basket. Les baskets sont des conteneurs qui sont constitués par modèle. Ils contiennent une collection compacte d'objets. Dans le cas présent, compacte signifie que tous les objets en relation au sein du thème sont contenus dans le basket. Dans le cadre de fichier de transfert, il est possible d'avoir plusieurs baskets par thème. Afin de déterminer dans quel basket vont les features, une référence au basket doit obligatoirement être spécifiée pour chaque feature. Tous les baskets possèdent la même structure (correspondant au topic) mais peuvent contenir des données différentes.

### 2.1.7 Interlis Tools

Les outils présentés dans ce chapitre sont disponibles gratuitement sur le site [www.interlis.ch](http://www.interlis.ch) [9] (→ INTERLIS 2 → A télécharger → Outils pour INTERLIS 2.3).

- Compileur : permet de vérifier la conformité d'un modèle « .ili ».
- Checker : permet de vérifier qu'un fichier de transfert (.xtf) est conforme au modèle correspondant (.ili).
- UML Editor : outil de modélisation permettant d'exporter directement en modèle INTERLIS.

## 2.2 FME

FME est un ETL (pour Extract-Transform-Load) spatial, pour l'échange, la transformation, le chargement et le contrôle des données spatiales vectorielles ou raster. Capable de lire et d'écrire plusieurs centaines de formats différents (vectoriels, raster et non-spatiaux), FME s'impose comme un outil indispensable dans la manipulation de géodonnées.

Pour la compréhension de ce rapport, il est conseillé d'avoir déjà une certaine expérience dans l'utilisation de FME.

Nous rappelons qu'il existe des tutoriaux gratuits en ligne sur le site de SAFE :

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

### 2.2.1 Transformers

Si certains transformers utilisés dans les exemples de ce document vous sont inconnus, il faut se reporter au guide de référence des transformers FME [3].

### 2.2.2 Attributs

Les attributs sont les données associées aux features. Ils peuvent être propres au format (*format attributes*, dans ce cas, ils ne sont pas directement disponibles dans FME), ou spécifiques à la données (*user attributes*). Par exemple, dans le cas du format INTERLIS, les attributs propres au format (*format attributes*) sont préfixés par *xtf\_* :

- *xtf\_id*
- *xtf\_class*
- *xtf\_basket*

Concernant la géométrie d'une classe : si FME est en mesure de détecter un type de géométrie dans les données en lecture, elle sera définie comme géométrie implicite. Elle est donc directement disponible dans FME. Par contre, si plusieurs géométries existent, ou que la géométrie est composée d'une structure complexe, la géométrie est stockée dans un attribut, et il est nécessaire de la récupérer manuellement dans le workspace FME.

### 2.2.3 Utilisation des listes

Afin de bien comprendre toutes les étapes de la préparation des données dans FME avant l'écriture en INTERLIS via le plugin *ili2fme*, il est nécessaire de bien comprendre le concept de « listes » dans FME.

Un attribut de type liste, qui est reconnaissable à la présence d'accolade « {} » dans son nom, autorise plusieurs valeurs pour une entité de base. Par exemple, une feature peut avoir les valeurs d'attributs suivantes :

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

Cette feature contient 3 attributs : Attribut 1, Attribut 2 et une liste MyList{}. La liste contient 3 éléments. Chaque élément est référencé par un index entre accolade.

De plus, la liste peut elle-même contenir un ou plusieurs attributs. En complexifiant légèrement l'exemple précédent :

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

La feature contient toujours les 3 mêmes attributs. Cependant, la liste est maintenant structurée car elle contient deux attributs : direction et value. La liste contient toujours 3 éléments, mais chacun a deux valeurs d'attributs.

*Références :*

- <http://fmepedia.safe.com/articles/FAQ/List-Attributes>
- [http://docs.safe.com/fme/html/FME\\_Transformers/FME\\_Transformers.htm#Transformers/listbuilder.htm](http://docs.safe.com/fme/html/FME_Transformers/FME_Transformers.htm#Transformers/listbuilder.htm)

### 3. INFORMATIONS GÉNÉRALES - *ILI2FME*

*ili2fme* est un plugin INTERLIS pour FME développé par Eisenhut Informatik AG et qui est disponible en open source.

Dans ce chapitre, nous mettons l'accent sur certains points qui sont nécessaires pour une utilisation classique du plugin. Pour tous compléments d'informations, il faut se référer à la documentation officielle [4] :

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

Notons également l'existence du blog [www.ili2fme.ch](http://www.ili2fme.ch) [7] qui contient des informations d'utilisation qui peuvent s'avérer très utiles dans certains cas.

#### 3.1 Prérequis logiciels matériels

Il est premièrement nécessaire de pouvoir installer et utiliser FME sur le système souhaité. FME est disponible sur Windows, Linux et Mac. Dans la mesure du possible, il est conseillé d'installer les dernières versions de FME.

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

De plus, *ili2fme* est un plugin développé en Java. Il est donc nécessaire qu'un environnement Java (Java Runtime Environment), version 1.6.0 ou plus récent, soit installé sur le système.

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

Précisons qu'à partir de FME 2014, un environnement Java est directement installé avec FME.

#### 3.2 Concernant les versions

Le plugin est régulièrement mis à jour et les nouvelles versions sont mises à disposition sur le site [www.interlis.ch](http://www.interlis.ch) [9] (→ INTERLIS 2 → A télécharger → Outils pour INTERLIS 2.3).

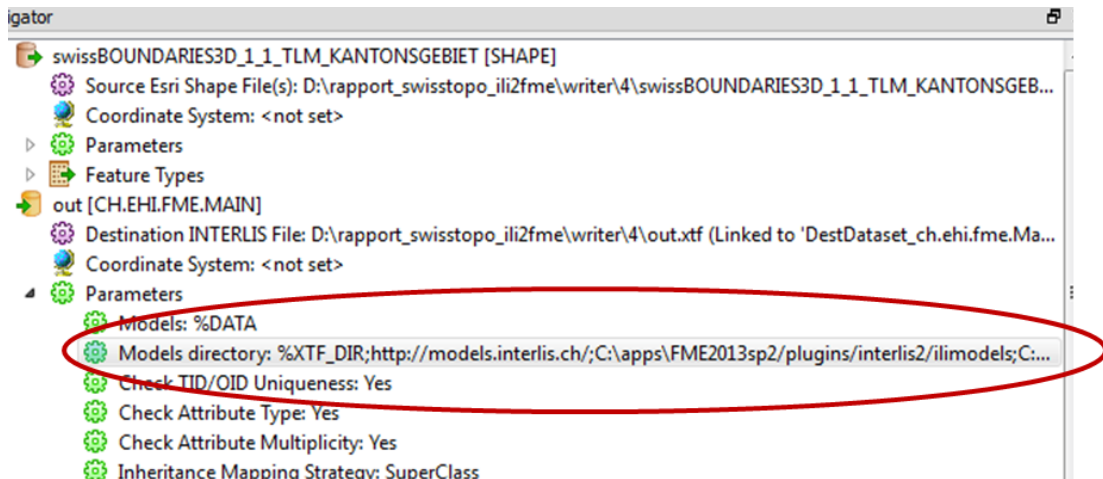
De manière générale, il est conseillé de travailler avec la dernière version disponible d'*ili2fme*. Si la version installée n'est pas la plus récente, le plugin peut être mis à jour (cf. chapitre 4.1).

#### 3.3 Modèle Interlis et fichiers de transfert

L'emplacement du fichier de transfert (*.xtf*) doit être explicité dans les paramètres, que ce soit en lecture ou en écriture.

Le ou les modèles qui sont référencés par le fichier de transfert doivent être disponibles dans l'un des répertoires qui est spécifié dans le paramètre *Models directory* du plugin. Par défaut, les valeurs suivantes sont assignées :

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



Les modèles INTERLIS peuvent dès lors être stockés dans l'un des répertoires suivants :

- %XTF\_DIR : dossier contenant les fichiers de transfert, en lecture ou en écriture.
- <http://models.interlis.ch/> : dépôt en ligne des modèles de référence, qui fait appelle, entre autre, au répertoire <http://models.geo.admin.ch/> [8]. Ce répertoire possède les dernières versions des modèles de la Confédération.
- Dans l'un des répertoires prévus dans le dossier d'installation de FME : \$(FME\_HOME)\plugins\interlis2\ilimodels & \$(FME\_HOME)\plugins\interlis2\ili22models.

Ajoutons qu'il peut être judicieux de définir un répertoire de stockage spécifique, lorsqu'un nombre important de modèles doivent être gérés et centralisés. Ainsi, il est possible d'ajouter un répertoire local, simplement en mettant une virgule, puis le chemin d'accès du répertoire.

### 3.4 Description des paramètres

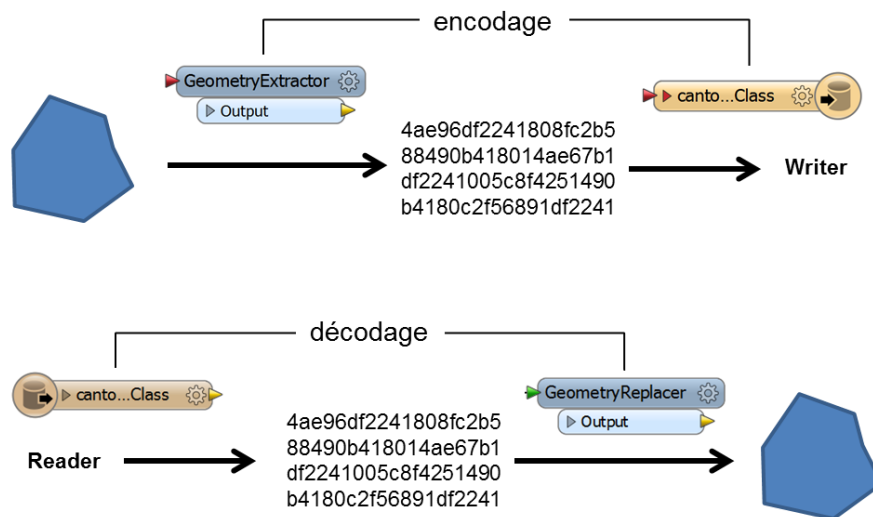
Seuls les principaux paramètres concernant INTERLIS 2 sont repris ici :

- TOPICS FILTER : Permet de spécifier les noms (*MODEL.TOPIC*) de thème (TOPIC) qui seront lus par FME.
- Check Attribute Type : Permet de vérifier que le type des attributs correspond à la définition dans le modèle de données.
- Check TID/OID Uniqueness : permet de vérifier que tous les objets possèdent un identifiant unique (*xtf\_id*).
- Geometry Encoding : ce paramètre permet à *ili2fme* de spécifier l'encodage utilisé dans le workspace. Le type d'encodage doit correspondre à celui qui est utilisé dans les transformers GeometryReplacer ou GeometryExtractor.
- Inheritance Mapping Strategy : cette option n'est utile que dans les cas où des classes héritent d'autres classes. En choisissant la stratégie « SuperClass », les attributs de la classe principale (root class), ainsi que ceux des classes annexes sont rassemblés en une classe dans FME. En choisissant la stratégie « SubClass », FME va créer une classe par sous-classe et y intègre les attributs de la classe principale.
- Create Feature Types For Enumerations : Certains modèles Interlis contiennent des listes de valeurs. Ces listes sont définies dans la partie DOMAIN du fichier ili. En choisissant d'activer l'option « SingleType », FME propose lors de l'ajout d'un reader, une classe nommée XTF\_ENUMS. Cet élément contient toutes les valeurs de toutes les listes de valeur du modèle. En choisissant « One feature type per enumeration », nous obtenons alors une classe par liste de valeur.

- Mapping of multiple Geometry Attributes : Ce paramètre apparu depuis la version 5.10.0 permet de simplifier la visualisation des objets possédant des géométries multiples ou multipart. Par défaut, l'option *EncodeAsFmeAttribute* ne prendra que la première géométrie dans le cas de géométries multiples. Pour les géométries multipart, il faudra procéder à l'extraction des informations contenues dans les listes. L'option *RepeatFeature* permet de dédoubler chaque feature possédant plusieurs géométries. Ainsi, FME peut directement lire toutes les géométries.
- *http Proxy Host* et *http Proxy Port* : Ces des paramètres permettent de définir un serveur qui peut être utilisé pour accéder au répertoire des modèles.

### 3.5 Encodage de la géométrie

Ili2fme nécessite certaine fois que la géométrie soit encodée ou décodée. C'est notamment le cas lorsque plusieurs géométries sont définies ou quand la géométrie est incluse dans une sous-structure.



FME propose plusieurs types d'encodage. Il est important que le type choisi dans le *GeometryExtractor* / *GeometryReplacer* corresponde au type défini dans le reader / writer. Pour ce faire, le tableau de correspondance suivant peut être utile.

Reader/Writer Paramètre Geometry Encoding	Transformer GeometryExtractor ou GeometryReplacer
FMEXML	FME XML
FMEBIN	FME BINARY
FMEHEXBIN	HEX Encoded FME Binary
OGCHEXBIN	HEX Encoded OGC Well Known Binary

De manière générale, il est conseillé d'utiliser FMEBIN ou FMEXML. En effet, il peut arriver que les deux autres types d'encodage implique des pertes d'objets, notamment si ceux-ci contiennent des arcs.

## 3.6 Attributs de format

Lors de la lecture ou de l'écriture d'un fichier INTERLIS 2, les attributs suivants, qui sont spécifiques au format INTERLIS, sont rencontrés :

- *xtf\_class* : permet de définir dans quelle classe sont (ou seront destinés) les attributs. Cette définition prendra en compte aussi le *TOPIC* correspondant.
- *xtf\_id* : permet de renseigner un identifiant unique pour chaque feature.
- *xtf\_basket* : permet de spécifier l'appartenance à un basket.

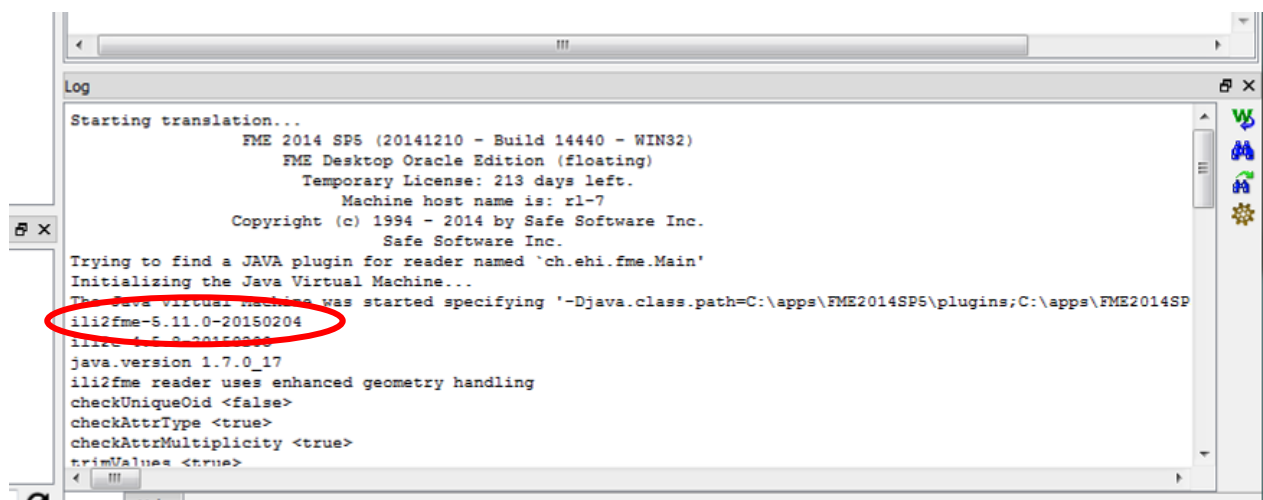
## 4. DÉMARRAGE RAPIDE

### 4.1 Installation

Le plugin *ili2fme* est directement installé lors de l'installation de FME. Ce qui implique que le format Swiss INTERLIS (*ili2fme*) est immédiatement disponible dans la liste des formats et pour tous les modules de la suite FME (Workbench, QuickTranslator, Inspector et Viewer).

Pour vérifier que le plugin est correctement installé sur le système, il suffit donc d'aller dans le menu *Readers*, puis *Add Readers* et sélectionner le format *Swiss INTERLIS (ili2fme)*. Si le reader est correctement ajouté, le plugin fonctionne.

La façon la plus simple de connaître la version du plugin installé est d'ouvrir un fichier Interlis dans FME. La fenêtre Log (située en bas de l'écran) affiche des informations concernant le chargement du plugin. Sur la ligne commençant par *ili2fme*, le numéro de version du plugin est affiché, puis sa date de release.



Pour mettre à jour *ili2fme*, la dernière version de FME approuvée par swisstopo est téléchargeable sur le site <http://www.interlis.ch/> [9] (→ INTERLIS 2 → A télécharger → Outils pour INTERLIS 2.3). L'installation s'effectue en dézipant le fichier puis en remplaçant les répertoires et fichiers dans *FME Suite* aux endroits correspondant dans le répertoire d'installation de FME. Plus d'information dans le *README.txt* du répertoire *docs*.

### 4.2 Conversion rapide

Un fichier INTERLIS peut être rapidement converti en utilisant le FME Quick Translator. Pour ce faire, il suffit de spécifier le fichier Interlis comme reader, et un autre format en writer.

Cependant, ce type de transformation rapide présente quelques limitations :

- La conversion effectuée est de type 1:1. Chaque classe rencontrée dans le modèle INTERLIS est convertie en une classe dans le format de sortie. Or, il arrive souvent que des classes supplémentaires soient nécessaires dans le format d'écriture, ou au contraire que des simplifications soient possibles.
- Par exemple, dans le cas d'une conversion souhaitée de INTERLIS vers du SHAPE. Un attribut INTERLIS multilingue (cf. exemple 5.3.1) ne peut pas être directement stocké dans le Shapefile. Il faudrait donc manuellement créer une deuxième classe en sortie (par exemple en .DBF) contenant les différentes valeurs selon la langue ainsi qu'une clé étrangère pour effectuer la relation.



- Dans le cas où certaines structures INTERLIS sont stockées sous forme de liste dans FME (par exemple, des géométries multi-parties), la conversion directe ne fonctionnera pas. Ainsi, si un utilisateur convertit le fichier *xtf* créé lors du premier exemple (cf. 5.1), il n'obtiendra que les données attributaires dans le nouveau format. Pour une traduction complète et exhaustive d'un fichier INTERLIS 2, l'utilisation du plugin *ili2fme* doit être complétée par une série d'opérations dans un workspace FME. L'utilisation du FME Workbench est donc indispensable dans ce cas.

### 4.3 Visualisation rapide

Les outils FME Data Inspector ou FME Universal Viewer peuvent être utilisés pour visualiser rapidement le contenu d'un fichier INTERLIS. Cependant, et pour la même raison qu'expliquée dans le chapitre 4.2, une géométrie multipart ou multiple sera complètement visible uniquement en spécifiant *RepeatFeature* dans le paramètre *Mapping of multiple Geometry*. Par contre, les éventuelles liaisons entre les classes ne seront pas prises en compte.

Pour les utilisateurs d'ArcGIS Desktop, il est possible de visualiser et manipuler des fichiers INTERLIS. Pour ce faire, il faut activer le FME Extension for ArcGIS dans la FME Intergration Console. Pour de plus amples informations, il faut se référer à la documentation en ligne d'ArcGIS [6].

## 5. CAS D'UTILISATION

Tous les workspaces décrits dans ce document sont disponibles en annexe et ont été développés avec FME 2014 SP4.

### 5.1 Écriture d'un fichier INTERLIS 2

Ce chapitre fait référence au workspace *w-Doc\_ili2fme\_Ex1-isa.fmw*

Mots-clés :

Géométrie multi-parties

Structure INTERLIS

Writer INTERLIS

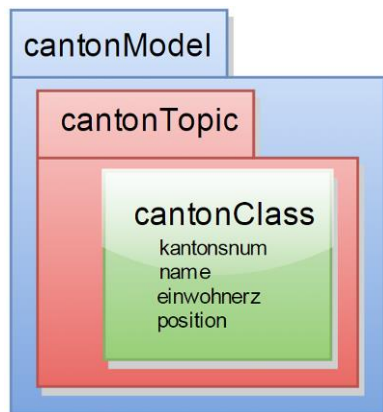
Encodage

#### 5.1.1 Présentation

L'objectif de cet exemple est de créer un fichier de données INTERLIS 2 (.xtf) à partir d'un fichier SHAPE. Pour ce faire, nous disposons du fichier<sup>1</sup> qui contient les limites cantonales ainsi que les attributs suivants :

- le numéro du canton (*KANTONSNUM*),
- le nom du canton (*NAME*),
- le nombre d'habitant (*EINWOHNERZ*)
- la géométrie du canton

Le modèle INTERLIS dans lequel nous souhaitons écrire est structuré de la manière suivante :



```
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.
```

Ce modèle est composé des éléments suivants :

- Le modèle *cantonModel* (MODEL) – contenant l'ensemble des données, ainsi que potentiellement des définitions d'unité (UNIT) ou de domaine (DOMAIN).
- Le thème *cantonTopic* (TOPIC) – une partie plus spécifique du modèle, qui peut lui aussi contenir des définitions ainsi que des attributs.
- La classe *CantonClass* (CLASS) – Qui contient tous les objets qui partagent les mêmes propriétés. Ainsi, les attributs et leurs types sont définis. Dans notre exemple, nous retrouvons nos trois attributs textuels décrits précédemment. Nous constatons qu'ils doivent être de type textuel et comporter au maximum 30 caractères (TEXT\*30). Le type de l'attribut position n'est,

<sup>1</sup> Disponible sur le site [http://www.toposhop.admin.ch/fr/shop/products/landscape/swissBoundaries3D\\_1](http://www.toposhop.admin.ch/fr/shop/products/landscape/swissBoundaries3D_1)

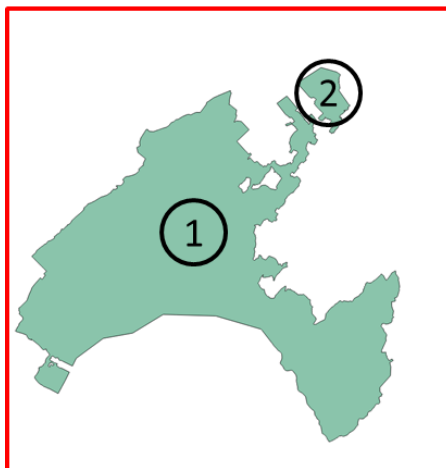
quant à lui, pas directement déterminé dans le dans le modèle *cantonModel*. En effet, il fait appel à un module de base CH-Base (c.f. chapitre 2.1.4), nommé GeometryCHLV03\_V1.MultiSurface. Cette dernière est chargée grâce à la ligne présente au début du modèle *IMPORTS GeometryCHLV03\_V1*;

Regardons plus en détail cette particularité :

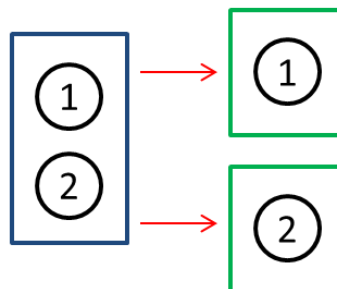
Modèle cantonModel.ili	Modèle CHBase_Part1_GEOMETRY_20110830_20150220.ili
<pre> INTERLIS 2.3;  MODEL cantonModel AT "http://www.inser.ch" VERSION "10.02.2015" = IMPORTS GeometryCHLV03_V1;  UNIT  DOMAIN  TOPIC cantonTopic =    CLASS cantonClass =     kantonsnum: TEXT*30;     name: TEXT*30;     einwohner: 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 &gt; 0.0001;  DOMAIN Coord2 = COORD 460000.000 .. 870000.000 [m] {CHLV03[1]}, 45000.000 .. 310000.000 [m] {CHLV03[2]}, ROTATION 2 -&gt; 1; </pre>

L'attribut position correspond à la structure MultiSurface. Cette dernière peut être composée d'une multitude de SurfaceStructure. Chaque SurfaceStructure est composée d'une surface formée par des coordonnées (Coord2). La zone limite de ces coordonnées est définie dans le domaine. Cette manière de définir la géométrie est importante pour pouvoir stocker des objets de type multi-parties. C'est notamment le cas pour le canton de Vaud, qui possède une enclave dans le canton de Fribourg.

Niveau 1  
cantonClass



Niveau 2  
position  
GeometryCHLV03\_V1.MultiSurface



Niveau 3  
Surface  
GeometryCHLV03\_V1.SurfaceStructure

Ainsi, l'élément « canton de Vaud » comprend deux géométries. Cette particularité devra absolument être prise en compte dans le workspace FME pour générer le fichier INTERLIS 2. Celui-ci devra donc pourvoir stocker la géométrie de la façon suivante :

```
<GeometryCHLV03_V1.MultiSurface>
  <Surfaces>
    <GeometryCHLV03_V1.SurfaceStructure>
      <Surface>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <COORD></COORD>
              <COORD></COORD>
              .... Coordonnées du polygone ...
            </POLYLINE>
          </BOUNDARY>
        </SURFACE>
      </Surface>
    </GeometryCHLV03_V1.SurfaceStructure>
    <GeometryCHLV03_V1.SurfaceStructure>
      <Surface>
        <SURFACE>
          <BOUNDARY>
            <POLYLINE>
              <COORD></COORD>
              <COORD></COORD>
              .... Coordonnées du polygone ...
            </POLYLINE>
          </BOUNDARY>
        </SURFACE>
      </Surface>
    </GeometryCHLV03_V1.SurfaceStructure>
  </Surfaces>
</GeometryCHLV03_V1.MultiSurface>
```

## 5.1.2 Création du workspace FME

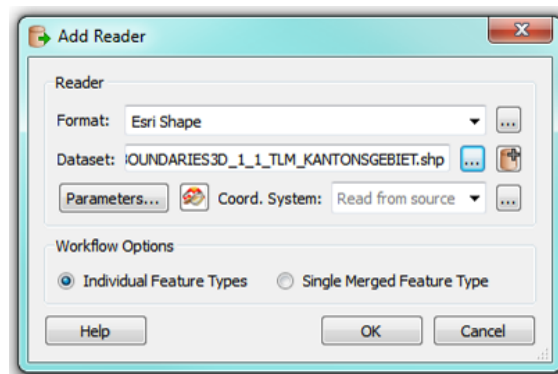
### 1. Traitement du fichier SHAPE

1.1. Dans un premier temps, nous allons lire le fichier SHAPE. Pour ce faire, il faut ajouter un reader Esri Shape:

*Menu Reader -> Add reader*

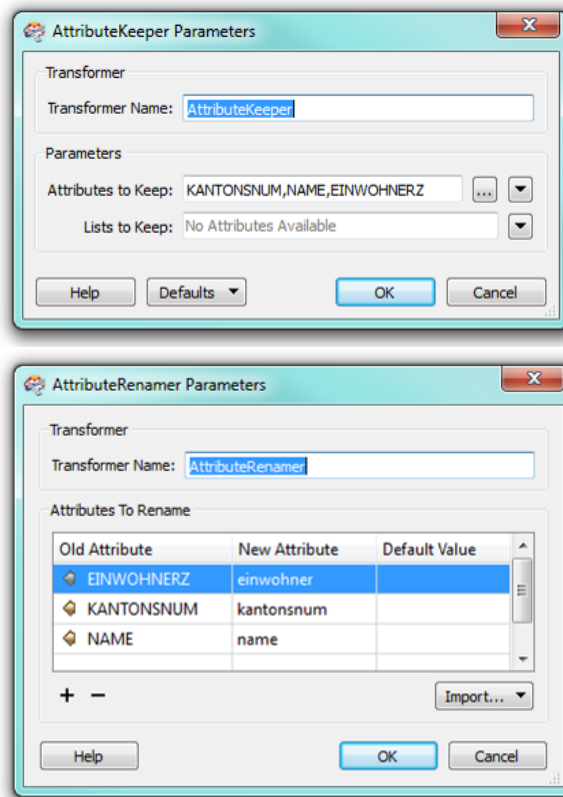
1.2. Puis dans Format, sélectionner Esri Shape

1.3. Entrer le chemin d'accès du fichier dans la partie Dataset



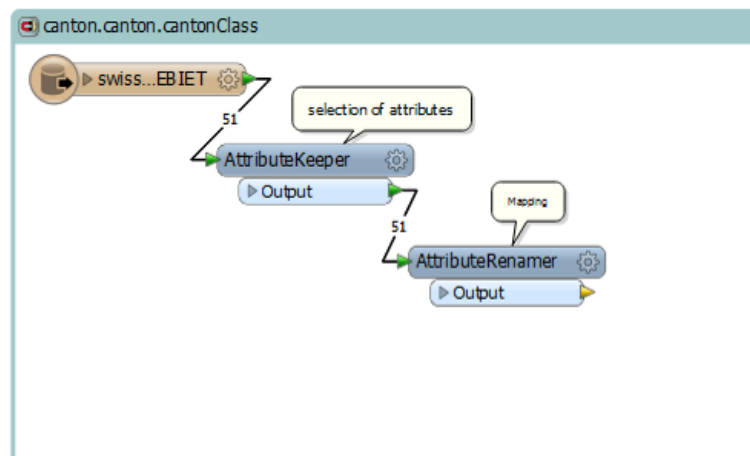
1.4. Il est important que les noms des attributs du fichier SHAPE et les noms des attributs définis dans le modèle de données INTERLIS correspondent. Pour ce faire, nous allons garder

uniquement les attributs que nous utiliserons par la suite, puis nous allons les renommer. Nous utilisons successivement un *AttributeKeeper* et un *AttributeRenamer*.

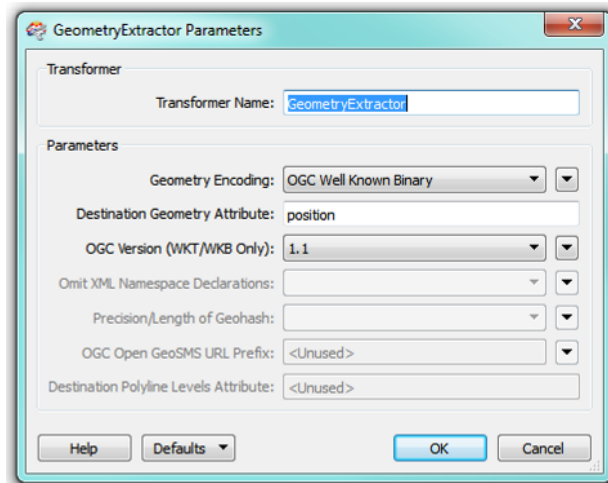


## 2. Traitement propre à l'écriture en INTERLIS 2

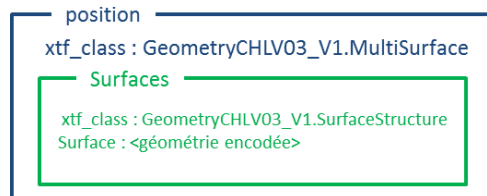
Ainsi, notre procédure doit actuellement être similaire à celle décrite ci-dessous.



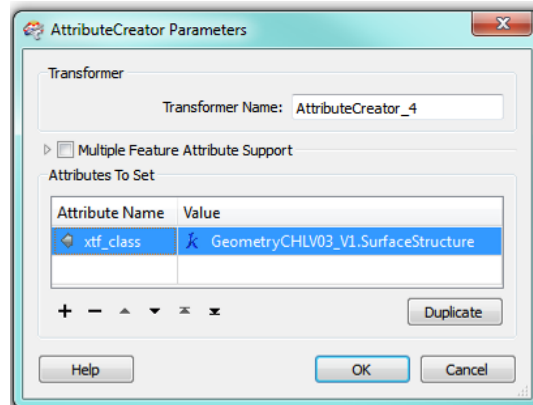
2.1. Pour qu'elle puisse être enregistrée correctement, la géométrie doit être encodée. Pour ce faire, nous allons utiliser le transformeur *GeometryExtractor*. Plusieurs formats d'encodage sont disponibles. Dans le cadre de ce tutoriel, nous choisissons le format *OGC Well Known Binary* dans le menu déroulant *Geometry Encoding*. L'attribut de destination doit être nommé de façon similaire au modèle, *position* dans le cas présent.



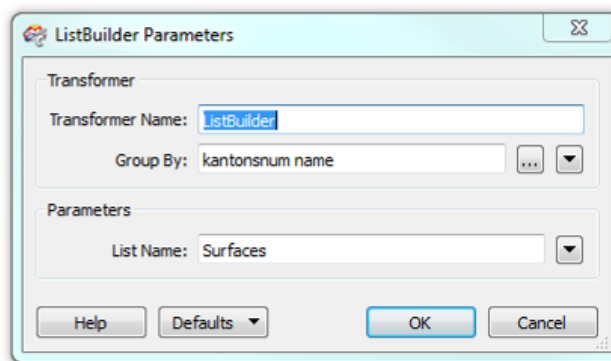
2.2. Nous devons maintenant regrouper toutes les géométries que nous venons d'encoder en un seul lot par feature. Pour ce faire, nous devons pour chaque canton, regrouper toutes les géométries dans des listes en respectant le modèle de données.



2.3. Nous allons maintenant créer la première liste (niveau 3), celle au niveau de la classe GeometryCHLV03\_V1.SurfaceStructure (elle va donc agréger toutes nos géométries). Pour ce faire, nous allons créer un nouvel attribut à l'aide d'un AttributeCreator.

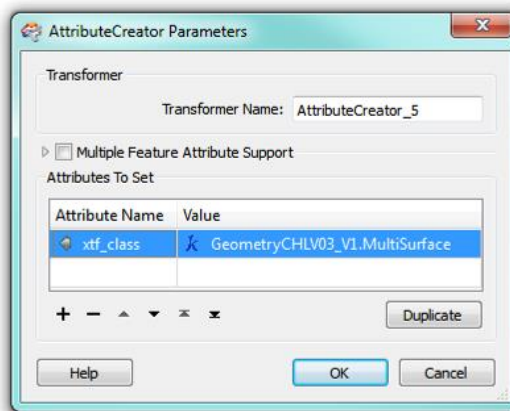


Cet attribut permet de spécifier la classe de la liste que nous allons créer de la façon suivante :

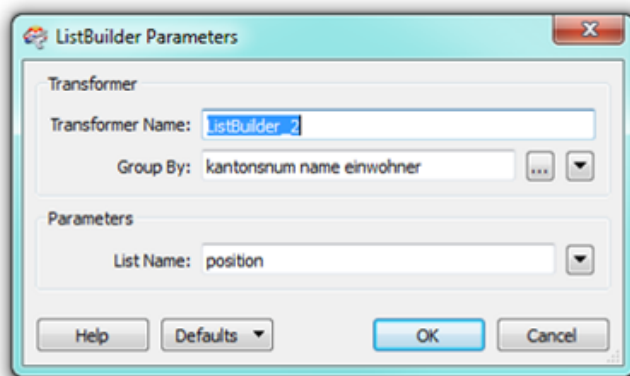


Grâce à cette commande, nous avons créé une liste des attributs groupés par numéro et nom de canton (ces attributs étant identiques pour chaque canton).

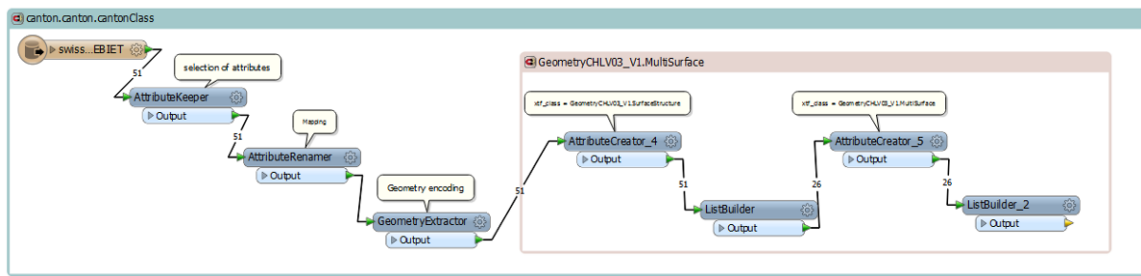
2.4. Puis nous pouvons réitérer cette même opération au niveau supérieur (niveau 2, celui de la classe GeometryCHLV03\_V1.MultiSurface).



La création de cette deuxième liste s'effectue comme suit :



On constate ainsi que la deuxième liste englobe la première et qu'à chaque niveau, nous avons renseigné le *xtf\_class*. Ainsi le workspace doit, à ce point, ressembler à cela :

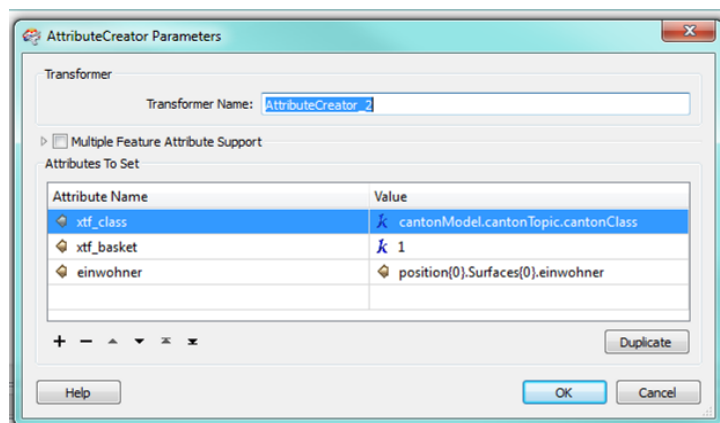


La technique, consistant à créer des listes multiples pour faire correspondre les données au modèle ili, est une notion fondamentale pour l'écriture et la lecture de fichier INTERLIS 2 via le plugin *ili2fme*.

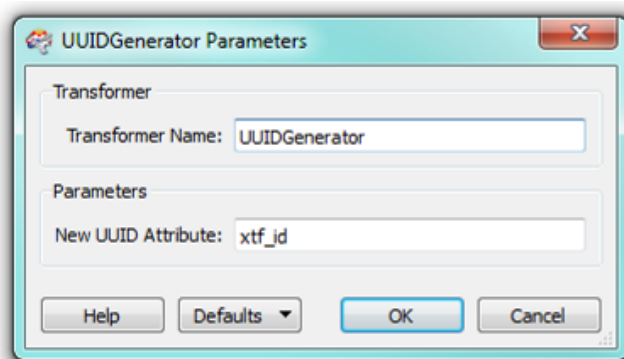
2.5. Nous pouvons maintenant nous occuper du dernier niveau (niveau 1), celui de la classe *cantonClass*. Comme précédemment, nous devons ajouter un attribut *xtf\_class* dont la valeur est définie comme suit : `<model>.<topic>.<class>`.

Dans le cas présent, cela prend les valeurs : *cantonModel.cantonTopic.cantonClass*. De plus, nous allons créer un seul *basket* pour stocker toutes les informations. De ce fait, nous devons ajouter l'attribut *xtf\_basket* = 1 à toutes les entités.

L'attribut *einwohner* a été inclus dans les listes. En effet, cet attribut n'est pas identique pour chaque partie de canton, seule la partie principale possède une valeur. De ce fait, nous devons manuellement l'extraire des listes.

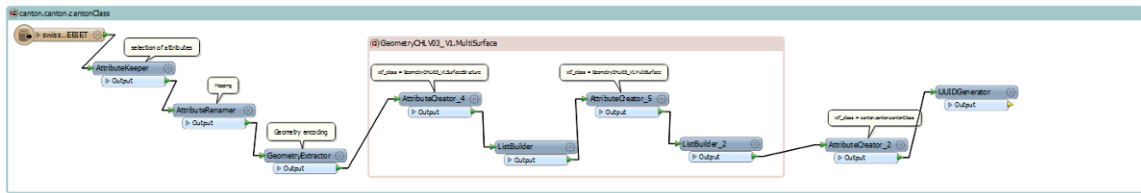


2.6. Pour terminer, nous devons aussi ajouter un *xtf\_id*, qui doit être un identifiant unique pour chaque objet. Nous utilisons un *UUIDGenerator*.



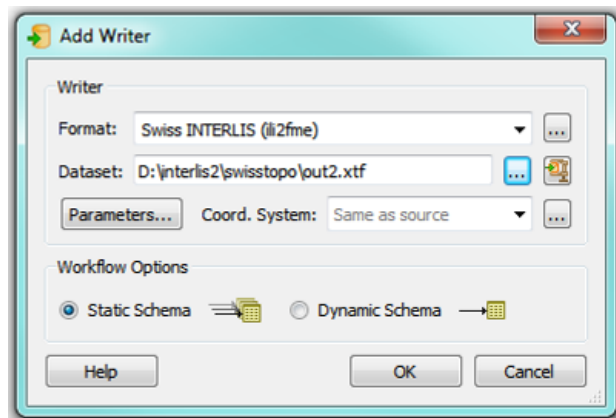


Notre workspace devient donc :



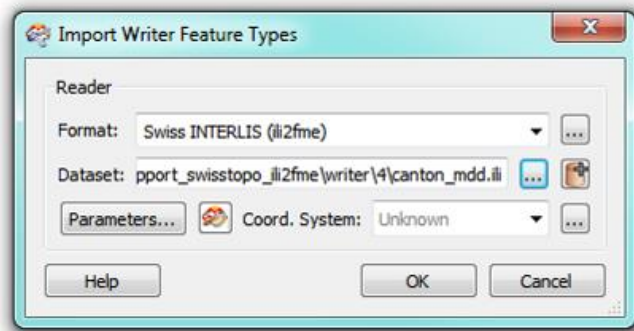
### 3. Ecriture en format Interlis 2

3.1. Pour ce faire, nous devons ajouter un writer. Nos spécifions le format *Swiss INTERLIS (ili2fme)* puis le Dataset en *INTERLIS Files .xtf*



3.2. Après avoir terminé l'ajout du Writer, FME nous propose de créer automatiquement un nouveau *feature type*. Etant donné qu'il faut définir la structure de nos Writer en fonction des modèles ili, nous devons refuser cette demande en cliquant sur NO.

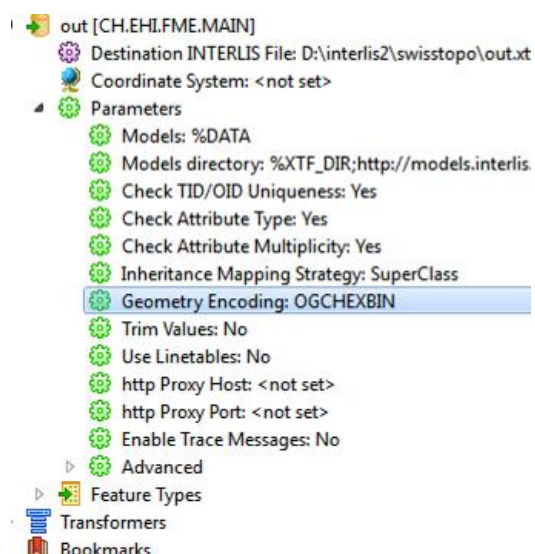
3.3. Ajouter le modèle de données en allant dans Writer -> Import Feature Type puis en sélectionnant le modèle de données .ili.



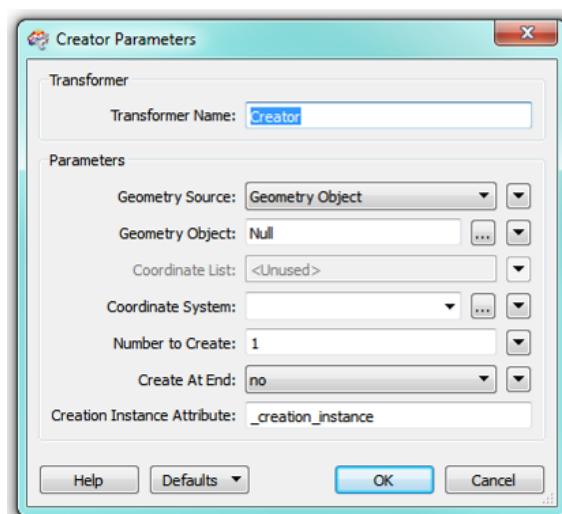
3.4. Dans le cas présent, seuls les Types *cantonModel.cantonTopic.cantonClass* et *XTF\_BASKET* doivent être sélectionnés.

3.5. Ainsi, le writer *cantonModel.cantonTopic.cantonClass* peut être connecté à la procédure précédemment créée.

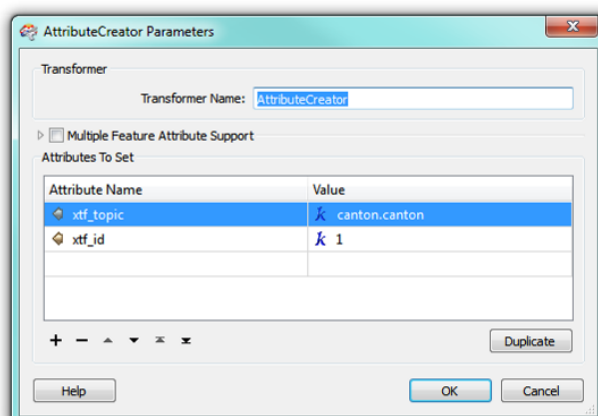
3.6. Dans les paramètres du writer, nous devons vérifier que l'encodage de la géométrie correspond bien à celui défini précédemment. Dans notre cas, nous devons sélectionner *OGCHEXBIN*.



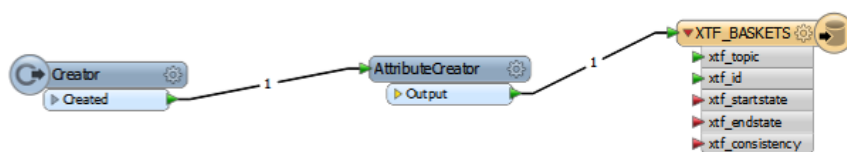
3.7. Pour terminer, le XTF\_BASKETS doit lui aussi être rempli. Pour chaque TOPIC, un objet comprenant les attributs `xtf_id` et `xtf_topic` doit être créé. Pour ce faire, nous devons ajouter un Creator comme suit :



3.8. Puis créer les attributs `xtf_topic = canton.canton` et `xtf_id = 1`



### 3.9. Le tout peut être connecté au writer *XTF\_BASKETS*



## 5.2 Lecture d'un fichier INTERLIS 2

Ce chapitre fait référence au workspace *w-Doc\_ili2fme\_Ex2-isa.fmw*

Mots-clés :

Listes	Structure INTERLIS	Reader INTERLIS	Encodage
--------	--------------------	-----------------	----------

### 5.2.1 Présentation

Dans le cadre de ce second exemple, nous disposons d'un fichier INTERLIS qui contient des données sur les cantons (données descriptives ainsi que géométriques). Un canton peut être composé de plusieurs parties (forme principale et enclave). La géométrie est donc de type multi-part et sa structure est celle qui est décrite dans le chapitre 5.1.1.

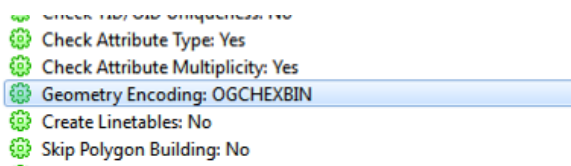
### 5.2.2 Création du workspace FME

#### 1. Lecture du fichier Interlis

1.1. Dans l'onglet Readers, sélectionner Add reader.

1.2. Le format sera du Swiss INTERLIS (ili2fme) et il faut sélectionner le fichier cantons.txtf comme data set.

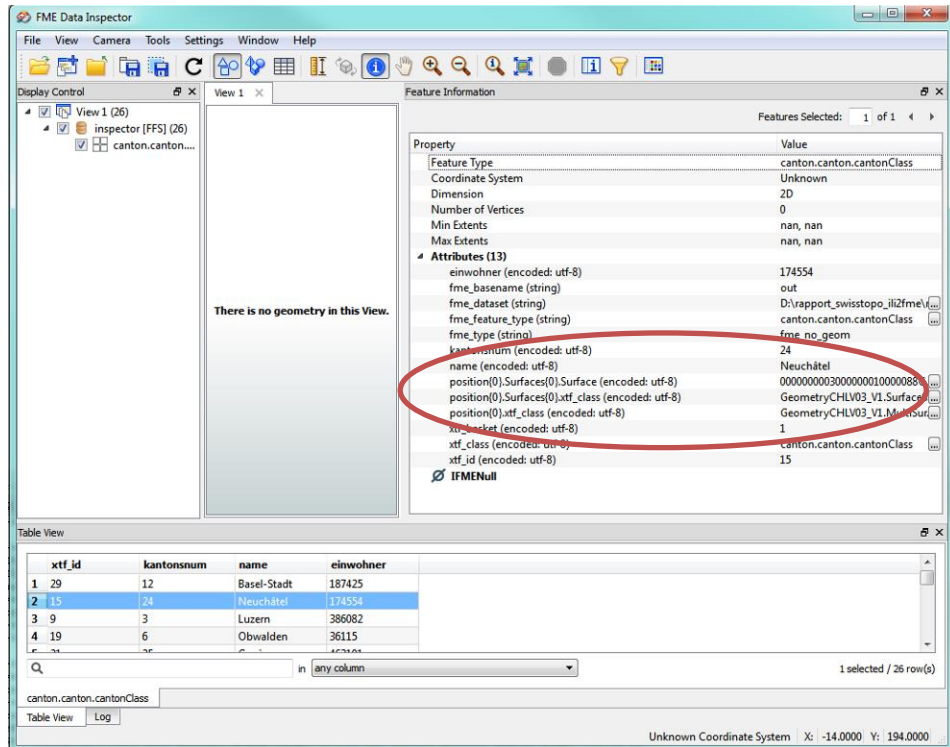
1.3. En cliquant sur le bouton Parameters, nous devons spécifier le type d'encodage de la géométrie. Dans le cas présent, il s'agit de OGCHXBIN.



Valider les deux fenêtres en cliquant sur OK pour importer les données

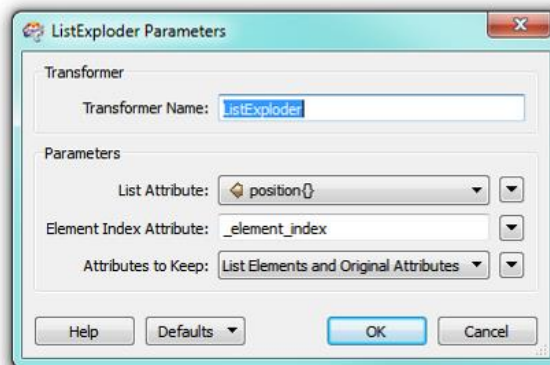
1.4. FME propose l'import de toutes les classes qui sont présentes dans nos modèles de données (le général, cantonModel.ili, ainsi que tous ceux importés via la balise IMPORT, c.f. chapitre 2.1.3). Dans le cas présent allons importer uniquement la classe : cantonClass.

1.5. En connectant un inspecteur au reader fraîchement importé, nous n'obtenons aucune géométrie. Cependant, nous pouvons constater que quatre attributs sont présents. Le premier attribut correspond à l'xtf\_id et les suivants sont les attributs descriptifs des cantons. La géométrie est stockée sous forme de liste. C'est de cette manière qu'ili2fme permet de regrouper des géométries multi-parties. En cliquant sur une valeur des attributs affichés, nous pouvons les voir dans la fenêtre Feature Information.

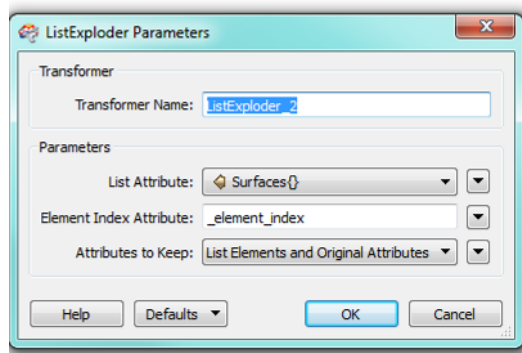


Afin de pouvoir utiliser ces informations, il faudra donc extraire ces listes.

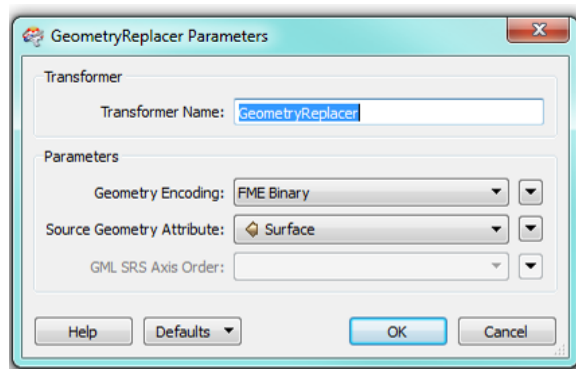
- 1.6. Puis à l'aide d'un ListExploder, nous allons exposer la liste position{}, ce qui correspond au nom de l'attribut géométrie dans le modèle ili.



En reconnectant un inspector à la sortie de ce ListExploder, nous pouvons voir que l'attribut xtf\_class a été créé. Les valeurs de cet attribut correspondent à la classe de géométrie des cantons. Cependant, il n'est toujours pas possible de visualiser la géométrie. Pour ce faire, nous devons effectuer une seconde exposition de liste, en sélectionnant la liste Surfaces{}.

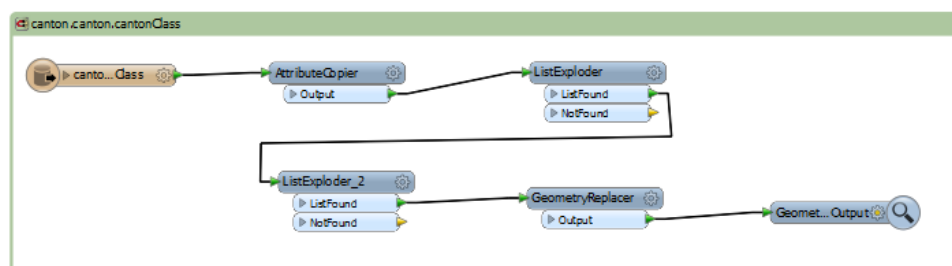


1.7. Ainsi, l'attribut surface est exposé. On remarque qu'il est encodé (d'où l'importance de choisir correctement le format au point 1.3). Nous allons utiliser le transformeur GeometryReplacer pour décoder cet attribut et créer la géométrie.



En connectant un inspecteur, nous pouvons maintenant visualiser la géométrie des objets<sup>2</sup>. A partir de ce point, il est possible d'utiliser de nombreux transformeurs pour caractériser cette géométrie (AreaCalculator, CircularityCalculator, etc...).

Le workspace doit ressembler à cela :



Remarque finale : Cet exemple permet de lire le fichier de données INTERLIS en fonction de son modèle. Chaque liste correspond à un niveau (cf. 5.1.1). Cette notion de niveau est importante principalement lors de l'écriture d'un fichier INTERLIS. Dans cet exemple, la lecture du fichier aurait pu être simplifiée en spécifiant RepeatFeature dans le paramètre Mapping of multiple Geometry Attribute.

<sup>2</sup> Si une erreur @Geometry function apparaît à cette étape, cela provient probablement d'un conflit d'encodage entre le fichier *xtf*, la définition de reader (cf. point 1.3) et/ou la définition du GeometryReplacer (point 1.7)

## 5.3 Lecture et écriture de fichiers INTERLIS 2 avec structures imbriquées

Ce chapitre fait référence aux workspaces *w-Doc\_ili2fme\_Ex5\_3\_1-isa.fmw* et *w-Doc\_ili2fme\_Ex5\_3\_2-isa.fmw*.

Mots-clés :

Structure imbriquées

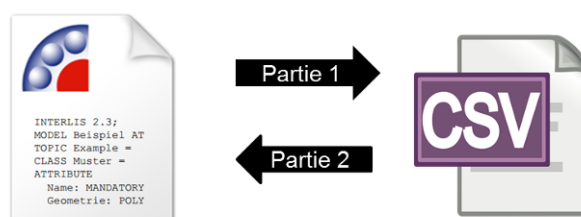
Structure INTERLIS

Reader/Writer INTERLIS

CHBase

### 5.3.1 Présentation

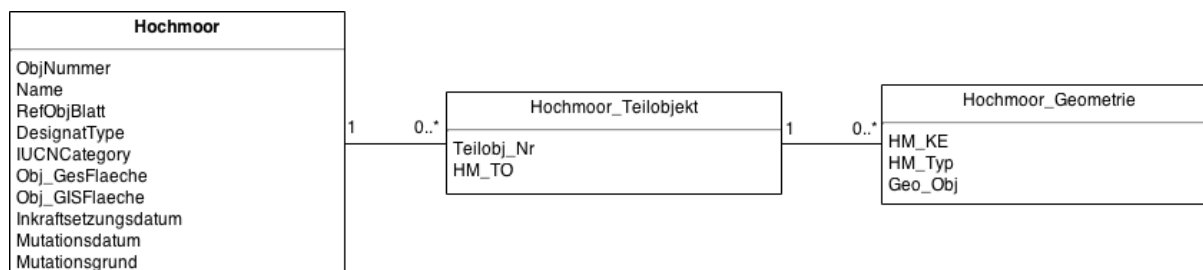
Cet exemple sera divisé en deux parties. Dans la première, l'objectif sera de retranscrire les informations contenues dans le fichier INTERLIS de base (*hochmoore.xtf*) dans un fichier CSV (*out.csv*). Pour la seconde partie, nous allons effectuer l'opération inverse, en régénérant un nouveau fichier INTERLIS (*out.itf*) sur la base du CSV. Ainsi, les deux fichiers XTF devront être semblables.



Nous allons baser cet exemple sur un cas concret de modèle INTERLIS. Nous disposons d'un fichier de données qui contient des informations sur les hauts-marais (données descriptives ainsi que géométriques). Notons que le modèle original *Hochmoore\_V1* a été légèrement modifié pour les besoins de l'exercice. Il est donc appelé *Hochmoore\_V1\_is*.

Le modèle de données est relativement plus complexe que celui utilisé dans les exemples 5.1 et 5.2. En effet, un marais peut être composé de plusieurs étendues d'eau. L'objet « marais » devra donc pouvoir stocker plusieurs géométries à l'intérieur du fichier INTERLIS. Nous parlons dans ce cas de géométries multi-part.

Le modèle de données prend la forme suivante :



Un objet de la classe *Hochmoor* peut avoir plusieurs *Hochmoor\_Teilobjekt*. De même, ces *Teilobjekt* peuvent être décrites en plusieurs géométries.

Le modèle de données (*Hochmoore\_V1\_is.ili*) comprend donc ces trois classes ainsi que des paramètres d'association qui renseignent sur le type de relation entre les tables. Par exemple, la classe *Hochmoor* contient de 1 à plusieurs éléments dans la classe *Hochmoor\_Teilobjekt*.

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

Avant de débiter la lecture du fichier INTERLIS 2 dans FME, il est intéressant d'analyser d'un peu plus près la définition de la classe *Hochmoor\_Geometrie*.

```
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;
```

Nous constatons que le premier attribut, *HM\_KE*, fait référence à une liste de code. Cette liste de code (stockée sous forme de fichier XML) fait référence à la structure *HM\_KE\_CatRef*. Celle-ci est elle-même définie grâce à la classe *HM\_KE\_Catalogue*. Nous constatons que l'attribut *Description* est de type *MultilingualText*. En regardant de plus près un extrait du fichier XML, nous voyons qu'il est ainsi possible de mettre en relation des attributs avec des listes de valeurs prédéfinies. Dans le cas présent, chaque *TID* permet de fournir une dénomination allemande, française ou italienne d'un terme précis.

```
<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>
```

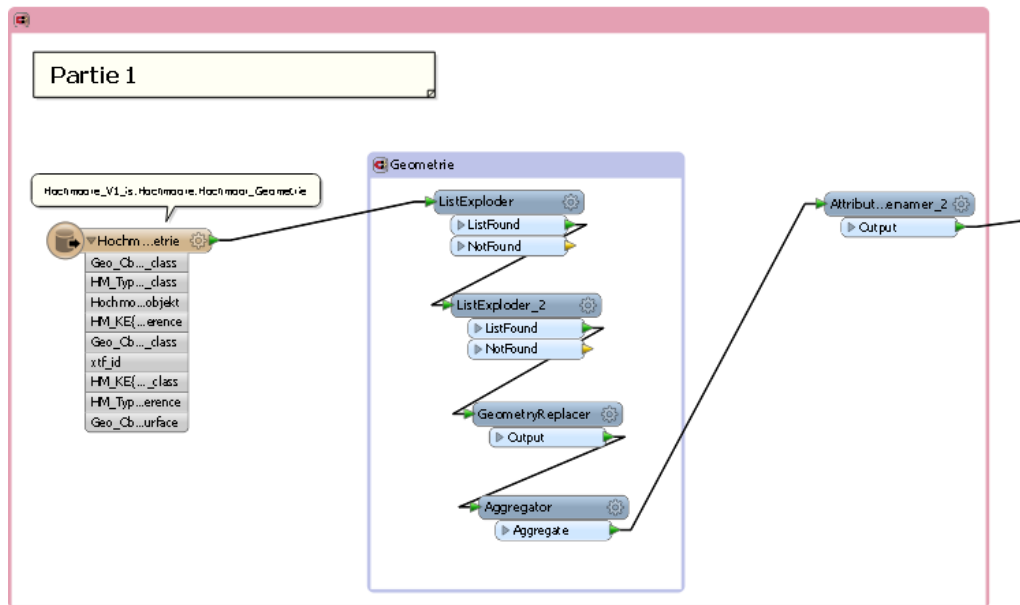
Cette particularité devra naturellement être intégrée dans le workspace FME.



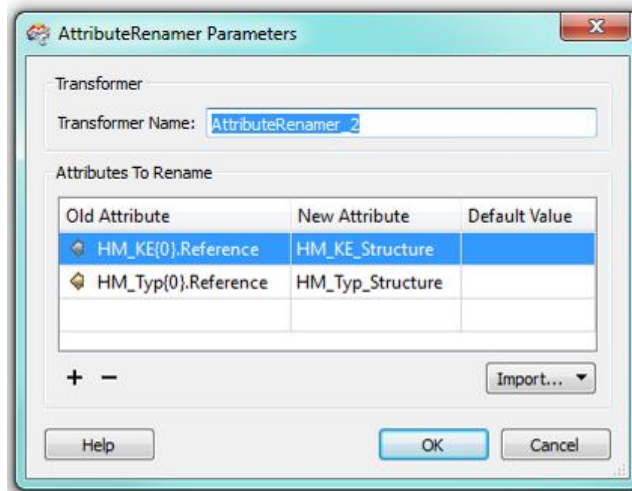
### 5.3.2 Enregistrement dans un fichier CSV

#### 1. Partie 1 : Lecture de la géométrie

1.1. Comme effectué dans les exemples précédents, nous allons exploser les 2 listes pour obtenir la géométrie, puis nous allons agréger le tout par le *xtf\_id*. Nous obtenons donc tous les hauts-marais avec leurs géométries (multi-parties ou non).



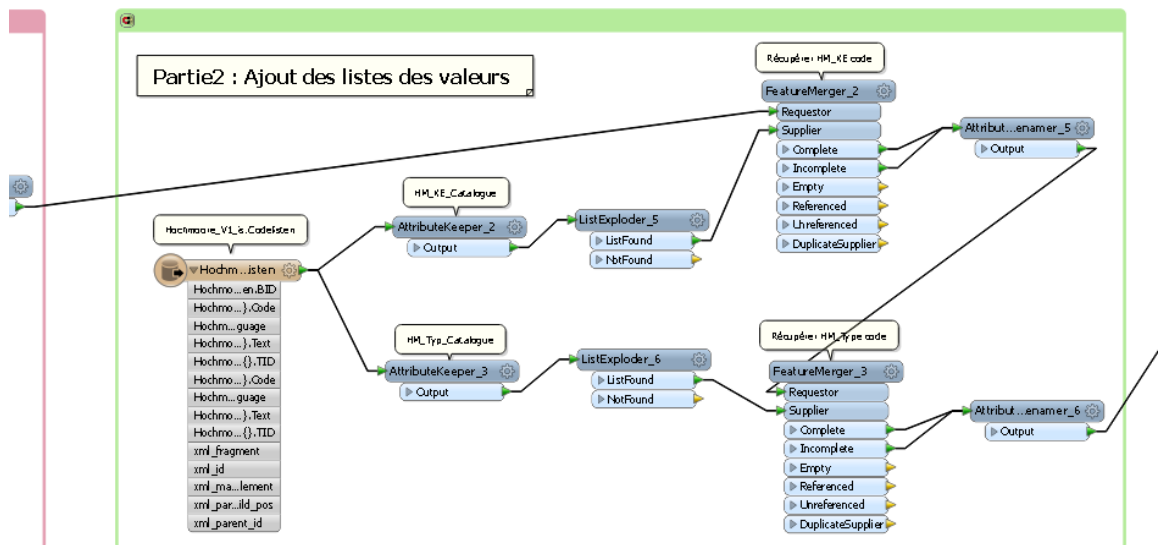
1.2. Le dernier transformeur AttributRenamer permet d'extraire les identifiants des références dans les listes de valeurs. Cela évite d'utiliser le transformeur ListExploder.



#### 2. Partie 2 : Ajout des listes de valeurs

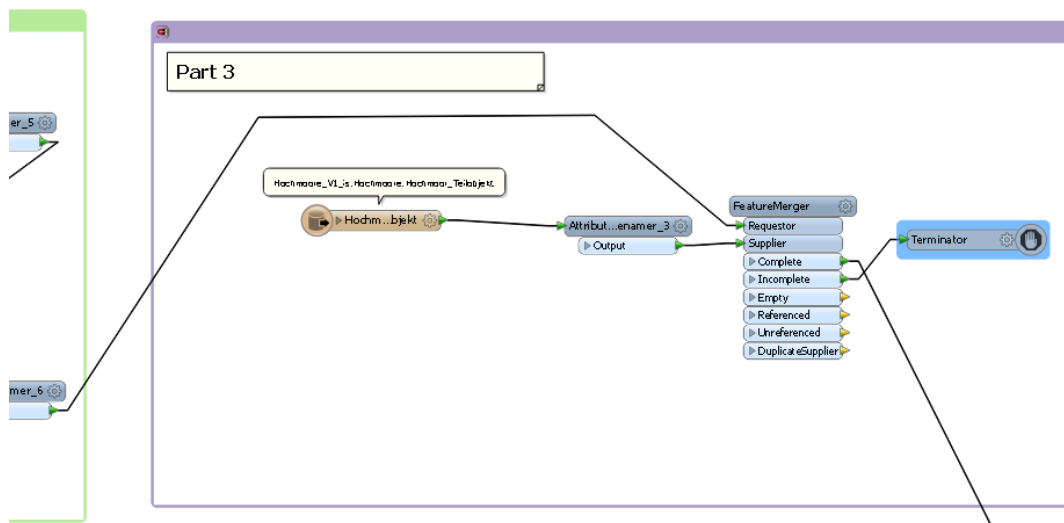
2.1. Nous allons maintenant charger dans le workspace FME les listes de valeurs contenues dans le fichier XML. Le but de cette opération est de remplacer les valeurs des attributs *HM\_KE\_Structure* et *HM\_Typ\_Structure* par les « traductions » contenues dans le XML. Pour ce faire, nous extrayons les valeurs pour chaque attribut à l'aide d'un *AttributeKeeper* et d'un *ListExploder*.

- 2.2. Puis nous ajoutons successivement ces valeurs à la géométrie grâce à un FeatureMerger. Finalement, nous renommons la partie de la liste correspondante à la langue que nous souhaitons (provenant du fichier XML) par HM\_KE et HM\_TYP.



### 3. Partie 3 : Ajout de la classe *Teilobjekt*

- 3.1. Nous pouvons ajouter les *Teilobjekt* à la géométrie de la façon suivante :



Nous avons ainsi créé la liaison entre *Hochmoor\_Geometrie* et *Hochmoor\_Teilobjekt*.



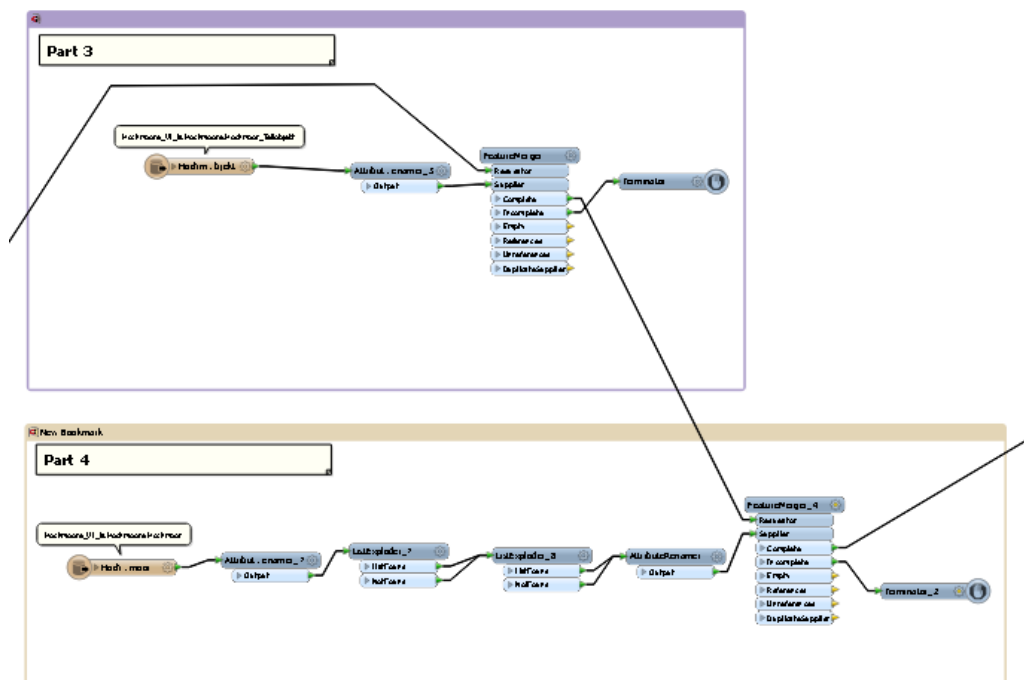
#### 4. Partie 4 : Ajout de la classe *Hochmoor*

En analysant la structure de la classe *Hochmoor*, nous nous apercevons que l'attribut *Mutationsgrund* fait référence à la classe *LocalisationCH\_V1*. Celle-ci se trouve dans le modèle *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. Ainsi, nous devons dans le workspace FME de appliquer deux *ListExploder* consécutifs pour obtenir les valeurs stockées. Veuillez noter ici que cet attribut n'est pas obligatoire (MANDATORY), et que donc les listes valeurs ne sont pas forcément existantes.

4.2. Nous pouvons assembler la géométrie à la classe Hochmoor en utilisant le xtf\_id comme attribut de jointure.

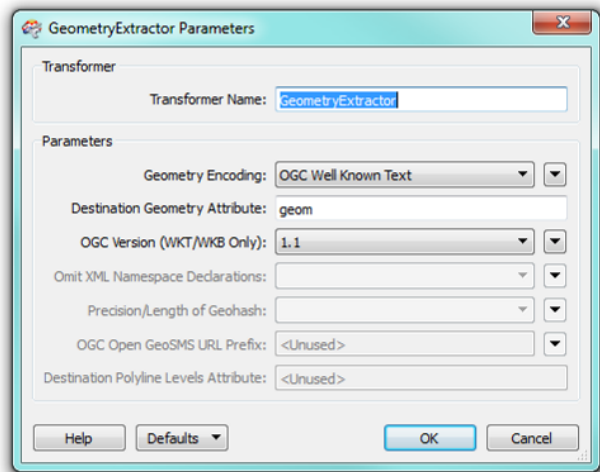


Les trois classes sont maintenant assemblées. Nous pouvons à présent travailler avec ces données.

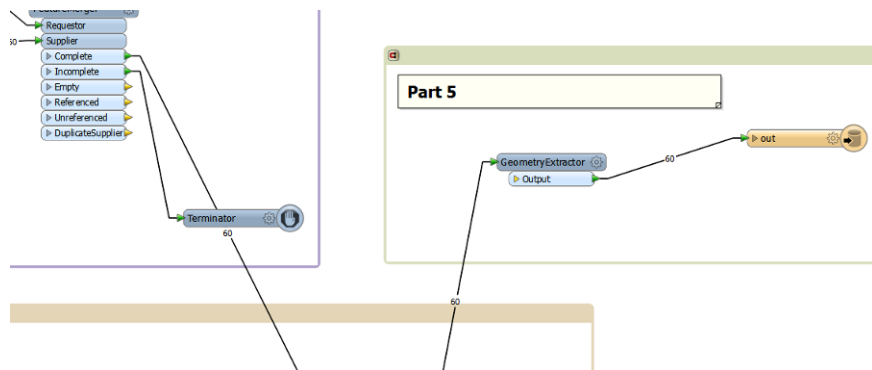
#### 5. Partie 5 : enregistrement dans un fichier CSV.

Nous allons maintenant enregistrer ces données dans un fichier CSV.

5.1. Afin de pouvoir stocker la géométrie, nous allons l'encoder en WKT.



5.2. Nous pouvons maintenant ajouter un writer CSV en y intégrant les attributs souhaités.



### 5.3.3 Enregistrement en fichier INTERLIS 2

L'objectif de cette seconde partie d'exercice consiste à récupérer le fichier CSV et à recréer un fichier INTERLIS.

#### 1. Partie 1 : Création de la liste sur la classe *Hochmoor*

La classe *Hochmoor* étant la classe principale du modèle (tous les objets y faisant référence), nous devons agréger tous les objets de cette classe présents dans le CSV.

1.1. Avant d'agréger les données, nous allons premièrement créer, en utilisant un *UUIDGenerator*, un identifiant unique pour chaque objet, nommons-le *xtf\_id\_object*.

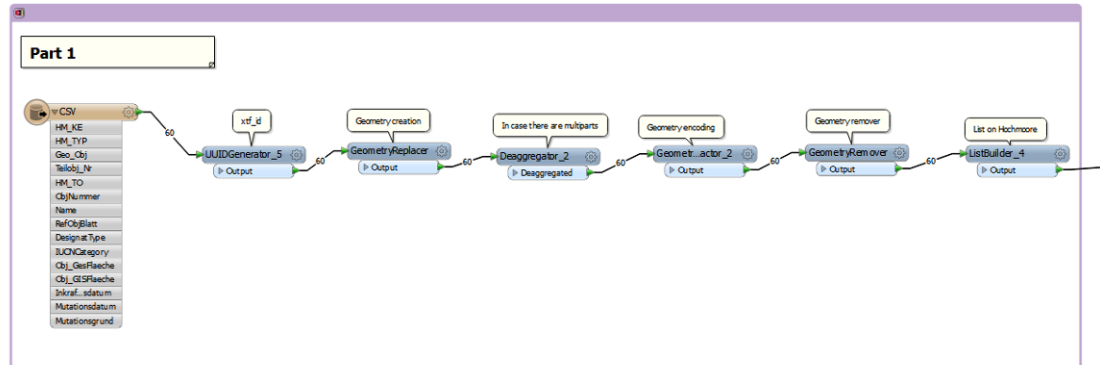
1.2. Puis nous pouvons régénérer la géométrie à l'aide d'un *GeometryReplacer* en sélectionnant le format FME Binary.

1.3. Puis, nous allons désagréger cette géométrie pour séparer les multipart.

1.4. Nous pouvons ré-encoder la géométrie de façon à ce qu'elle puisse être traitée comme un attribut normal. Nous pouvons choisir de nouveau du FME Binary et enregistrons-la dans un nouvel attribut nommé *\_geometry\_object*.

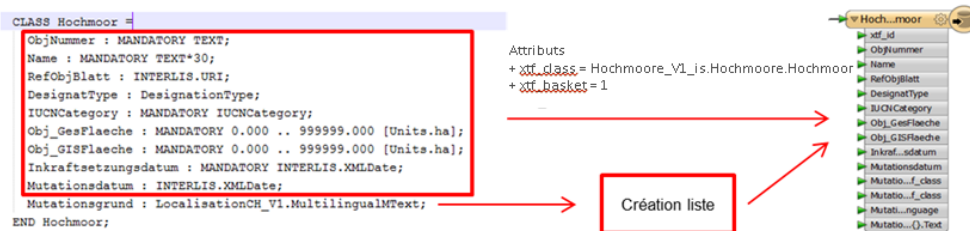
1.5. Supprimer la géométrie non-encodée

- 1.6. Finalement, nous pouvons créer les listes pour chaque objet de la classe hochmoor. Pour ce faire, nous devons les créer en les groupant (GROUP BY) par les attributs présents dans la classe. Dans le cadre de cet exemple, nous supposons qu'il n'y a pas deux objets possédant exactement les mêmes attributs dans cette classe.

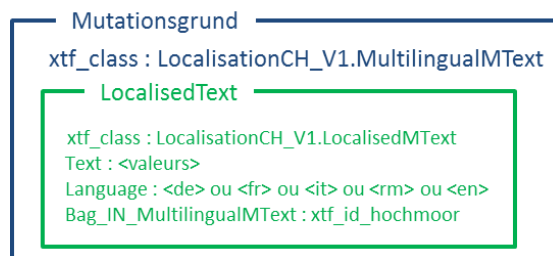


## 2. Partie 2 : écriture de la classe *Hochmoor*

La difficulté lors de l'écriture de cette classe est la contrainte concernant l'attribut *Mutationsgrund*.



Nous devons respecter la définition de la classe *Hochmoor*, et donc celle de *CHBASE Localisation\_V1*. De ce fait, une double liste sur le modèle suivant devra être créée.

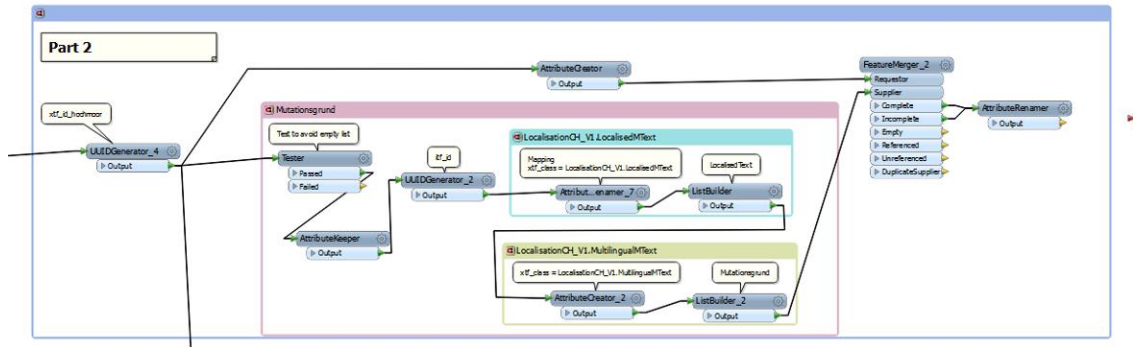


Pour plus d'informations théoriques concernant cette typologie, il faut se référer au chapitre 2.1.3.

- 2.1. Nous devons, avant de créer ces listes par objet, générer un *xtf\_id*. Nommons-le temporairement *xtf\_id\_hochmoor*. Celui-ci nous servira par la suite d'élément de jointure entre les listes qui vont être créées et les objets.
- 2.2. Pour l'écriture des objets dans la classe hochmoor, nous devons définir le *xtf\_class* ainsi que le *xtf\_basket* comme expliqué précédemment.
- 2.3. Les listes peuvent être parallèlement générées. Pour des raisons de simplicité, nous ne gardons que les attributs utiles à la génération des listes, à savoir : Language, Mutationsgrund (qui contient le texte) et *xtf\_id\_hochmoor*. Nous devons préalablement imposer un filtre pour ne générer des listes que pour les attributs qui possèdent des valeurs pour les remplir.

2.4. Puis nous devons créer le `xtf_id`, renommer les attributs pour correspondre au modèle, créer l'attribut `xtf_class` et générer les deux listes (`Mutationsgrund` et `LocalisedText`) en les regroupant par le `xtf_id_hoch_moor`.

2.5. Les listes générées peuvent maintenant être regroupées avec les objets grâce à un `FeatureMerger` avec comme attribut de liaison le `xtf_id_hoch_moor`.



### 3. Partie 3 : Géométrie - transformation de valeur selon le fichier XML

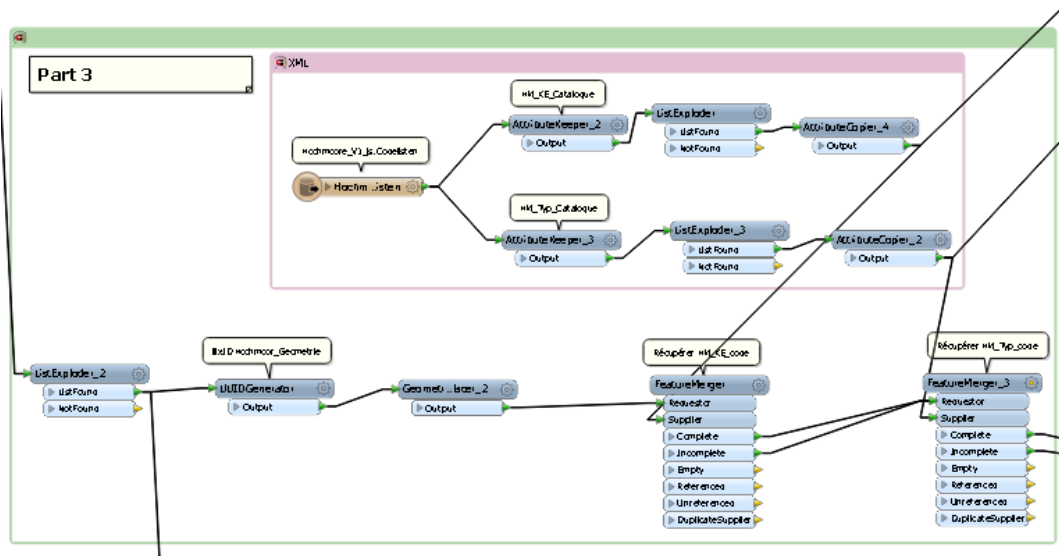
Nous pouvons maintenant traiter la classe *Hochmoor\_Geometrie*. Cela s'effectuera en deux étapes. Premièrement nous allons ajouter les listes de code depuis le fichier XML, puis nous allons traiter la géométrie et son enregistrement dans le fichier INTERLIS.

3.1. Récupérons nos objets après la définition de l'attribut `xtf_id_hochmoor`. Nous devons désagréger la liste qui a été faite pour le traitement de la classe *hochmoor* en utilisant un `ListExploder`.

3.2. Puis, nous pouvons régénérer un `xtf_id` pour la géométrie que nous appellerons temporairement `xtf_id_Hochmoor_Geometrie`.

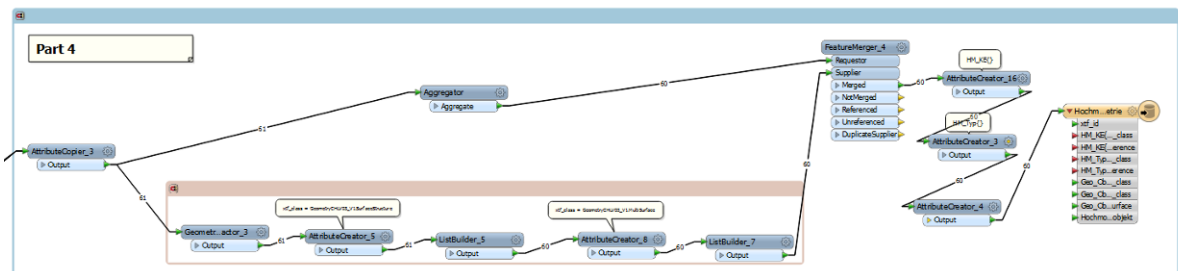
3.3. Nous pouvons recréer la géométrie. Nous avons actuellement autant d'objets que de géométries.

3.4. Les codes `HM_KE_code` et `HM_Typ_code` peuvent maintenant être récupérés du fichier XML, puis intégrés dans les objets à l'aide de deux `FeatureMerger` successifs. Ainsi, cette troisième partie doit ressembler à cela :



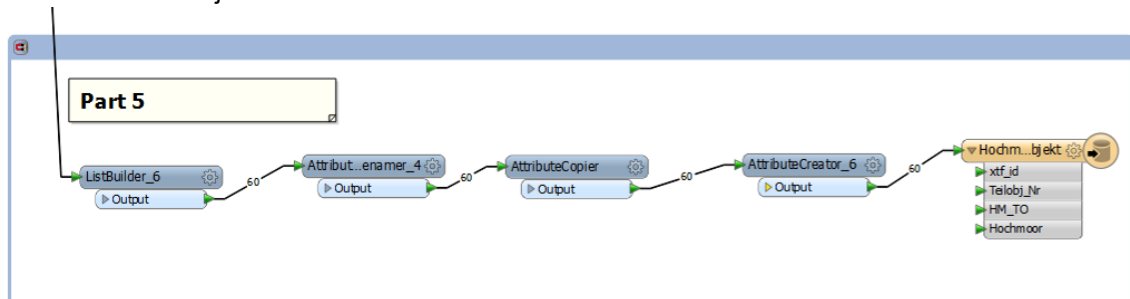
#### 4. Partie 4 : Géométrie - transformation des multi-parts

- 4.1. Nous pouvons créer les attributs qui seront utiles pour l'écriture de la classe géométrie dans le fichier INTERLIS (*xtf\_id*, *Hochmoor\_Teilobjekt*, *Geo\_Obj\_Structure*, *HM\_KE\_Structure* et *HM\_Typ\_Structure*).
- 4.2. Comme nous l'avons effectué dans l'exemple 5.1, nous devons créer deux listes pour correspondre au modèle de données et ainsi pouvoir gérer les géométries multi-parts.
- 4.3. L'attribut *Hochmoor\_Teilobjekt* est la clé étrangère qui est utilisée pour matérialiser l'association *AssociationDef194* définie dans le modèle INTERLIS. Il est automatiquement ajouté par le plugin ili2fme lors de l'ajout du writer.
- 4.4. Puis, ces listes seront ajoutées à l'agrégation des objets par l'attribut *xtf\_id\_object*.
- 4.5. Pour terminer, nous créons deux nouvelles listes pour *HM\_KE* et *HM\_Typ* en respectant le modèle de données.



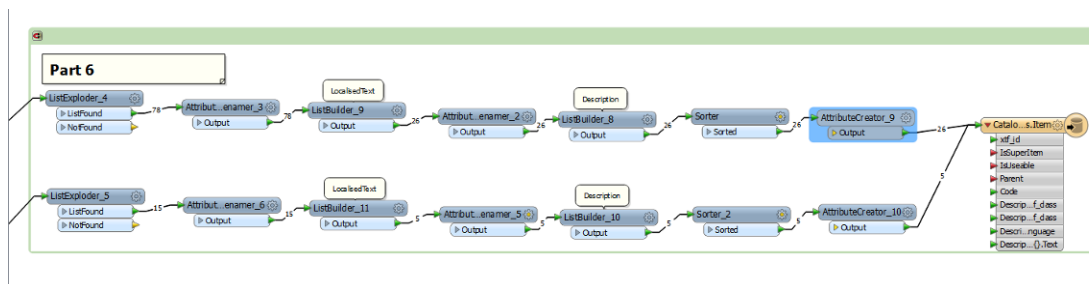
#### 5. Partie 5 : Ecrition de la classe *Hochmoor\_Teilobjekt*

- 5.1. Nous récupérons les objets non-agrégés en liste au début de la partie 3. Cette fois-ci, nous créons une liste sur les attributs de la classe *Hochmoor\_Teilobjekt* ainsi que le *xtf\_id\_object*.
- 5.2. Puis, nous créons les différents attributs *xtf\_class*, *xtf\_id* et *xtf\_basket* pour écrire les données.
- 5.3. L'attribut *Hochmoor* est la clé étrangère qui est utilisée pour matérialiser l'association *AssociationDef180* définie dans le modèle INTERLIS. Il est automatiquement ajouté par le plugin ili2fme lors de l'ajout du writer.



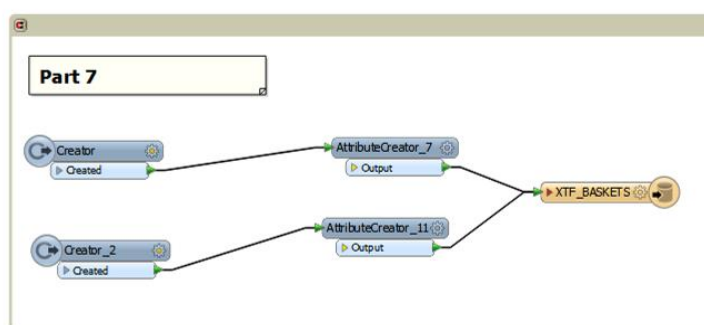
#### 6. Partie 6 : Ecrition des classes *HM\_KE\_Catalogue* et *HM\_Typ\_Catalogue*.

- 6.1. Pour ce faire, nous allons récupérer les données provenant du fichier XML, en extraire les listes *LocalisedText* pour y ajouter les attributs *xtf\_class*. Nous créons donc la liste *LocalisedText* et *Description*.



## 7. Partie 7 : XTF\_BASKETS

7.1. De façon similaire à l'exercice 5.1, nous devons créer le xtf\_basket. Cependant, nous créons cette fois-ci deux baskets. Le premier pour les Codelisten, le second pour le reste des données.





## 5.4 Conversion INTERLIS $\leftrightarrow$ bases de données relationnelles

Mots-clés :

ESRI file Geodatabase

PostGIS

Modelisation

CHBase

### 5.4.1 Présentation

L'objectif de cet exemple est de donner quelques éléments permettant d'effectuer une conversion entre un jeu de données INTERLIS et une base de données relationnelle. Deux technologies de systèmes de gestion de bases de données ont été retenues : ESRI file Geodatabase et PostGIS.

Le modèle INTERLIS utilisé dans cet exemple est directement tiré du modèle minimum du Cadastre des sites contaminés sur les aérodromes civils (CASIP OFAC). Le modèle complet *KbS\_LV03\_V1\_2* est disponible à l'adresse suivante :

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

Pour les besoins de l'exercice, le modèle de base a été modifié, pour ne retenir qu'un exemplaire des particularités qu'il contient. Le modèle ainsi obtenu est nommé *model\_54.ili*

De plus, le deuxième modèle *model\_54bis.ili* est généré, qui contient la définition d'une classe abstraite, permettant d'introduire la notion d'héritage dans le premier modèle.

Le tableau suivant décrit les différents éléments des deux modèles.

model_54.ili	Commentaires
<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 &gt; 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>Deux domaines sont définis dans le thème <i>Belastete Standorte</i> :</p> <ul style="list-style-type: none"> <li><i>Polygon</i> fait référence à CH-Base pour la définition de la géométrie</li> <li><i>UntersMassn</i> contient une liste de valeurs</li> </ul> <p>La structure <i>UntersMassn_</i> est définie avec un attribut <i>value</i>, dont la valeur doit être contenue dans le domaine <i>UntersMassn</i>.</p> <p>Cette première classe étend la classe <i>ZustaendigkeitKataster</i> définie dans le modèle <i>model_54bis</i> en lui ajoutant l'attribut <i>Bemerkung</i>.</p> <p>Cette deuxième classe peut contenir deux types de géométrie : polygone et point.</p> <p>L'attribut <i>Untersuchungsmaßnahmen</i> est de type BAG. Cela signifie qu'il peut être constitué d'une ou plusieurs valeurs, qui sont structurées selon <i>UntersMassn_</i>.</p>

<pre> URL_KbS_Auszug : INTERLIS.URI; END Belasteter_Standort;  ASSOCIATION ZustaendigkeitKatasterBelasteter_Standort =   ZustaendigkeitKataster -- {1} ZustaendigkeitKataster;   Belasteter_Standort -&lt;- {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>Cette association de cardinalité 1:n définit un lien entre les classes <i>ZustaendigkeitKataster</i> et <i>Belasteter_Standort</i>.</p> <p>Ce deuxième thème contient les significations dans les différentes langues des valeurs du domaine <i>UntersMassn</i>.</p>
model_54bis.ili	Commentaires
<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>Cette classe abstraite définit 2 attributs, qui peuvent être étendus dans une ou plusieurs autres classes d'un autre modèle.</p>

## 5.4.2 Implémentation du modèle conceptuel

La première étape de cet exercice est de modéliser la base de données relationnelle à partir du modèle conceptuel INTERLIS 2. Un rapport de swisstopo [5] traite de cette thématique et met en évidence des particularités de la modélisation INTERLIS et les manières de les implémenter dans une base de données relationnelles.

### 5.4.2.1 ESRI file GDB

ArcGIS 10.2.2 a été utilisé pour générer le modèle de cet exemple. La base de données *data\_54.gdb* est disponible annexe.

Quatre tables et deux *feature class* géométriques sont nécessaires pour convertir le modèle conceptuel INTERLIS en un schéma file GDB ESRI.



Le tableau suivant donne une description de chacune des tables :

KbS_LV03_V1_2_is.gdb	Commentaires																						
<p>Belasteter_Standorte :</p> <table border="1" data-bbox="309 394 919 667"> <thead> <tr> <th>Field Name</th><th>Data Type</th></tr> </thead> <tbody> <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> </tbody> </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>Table principale, qui ne contient pas de géométrie. Elle contient notamment une clé primaire <i>ilixID</i> ainsi qu'une clé étrangère <i>ZustaendigkeitKataster_FK</i>. Cette dernière matérialise l'association <i>ZustaendigkeitKatasterBelasteter_Standort</i> définie dans le modèle INTERLIS.</p> <p>Les attributs <i>InBetrieb</i> et <i>Nachsorge</i> sont liés à un domaine de valeur de type booléen.</p> <p>On remarque que l'attribut <i>Untersuchungsmassnahmen</i> n'apparaît pas dans cette table (voir ci-dessous).</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 et Belasteter_Standort_Geo_Lage_Punkt :</p> <table border="1" data-bbox="280 804 940 949"> <thead> <tr> <th>Field Name</th><th>Data Type</th></tr> </thead> <tbody> <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> </tbody> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Belasteter_Standort	Text	SHAPE	Geometry	<p>Concernant les géométries, les File GDB ESRI ne permettent de stocker qu'un type de géométrie par <i>feature class</i>. Dans notre cas, il est donc nécessaire de créer deux <i>feature class</i>, une de type polygone et l'autre de type point.</p> <p>Ces deux tables ont une structure semblable. Elles contiennent une clé primaire <i>ilixID</i> et une clé étrangère <i>Belasteter_Standort</i> qui fait référence à la clé primaire de la classe <i>Belasteter_Standorte</i>.</p>												
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
Belasteter_Standort	Text																						
SHAPE	Geometry																						
<p>Rel_UntersMassn:</p> <table border="1" data-bbox="272 1072 951 1218"> <thead> <tr> <th>Field Name</th><th>Data Type</th></tr> </thead> <tbody> <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> </tbody> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Bag_IN_Belasteter_Standort	Text	value	Short Integer	<p>Cette table de relation est nécessaire pour prendre en compte l'attribut <i>Untersuchungsmassnahmen</i>.</p> <p>En effet, une entité de la classe <i>Belasteter_Standorte</i> peut être liée à plusieurs <i>value</i> de la structure <i>UnterMassn</i>.</p> <p>Ainsi, cette table met en relation l'attribut <i>Bag_IN_Belasteter_Standort</i>, qui fait référence à la clé primaire de la classe <i>Belasteter_Standorte</i>, avec une ou plusieurs <i>value</i>, selon le domaine <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 border="1" data-bbox="269 1442 956 1621"> <thead> <tr> <th>Field Name</th><th>Data Type</th></tr> </thead> <tbody> <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> </tbody> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	Code	Short Integer	Text	Text	Sprache	Text	<p>Pour chaque code du domaine <i>UntersMassn</i>, on définit dans cette table une signification dans chacune des langues allemande, française et italienne.</p>										
Field Name	Data Type																						
OBJECTID	Object ID																						
ilixID	Text																						
Code	Short Integer																						
Text	Text																						
Sprache	Text																						
<p>ZustaendigkeitKataster:</p> <table border="1" data-bbox="280 1805 956 1980"> <thead> <tr> <th>Field Name</th><th>Data Type</th></tr> </thead> <tbody> <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> </tbody> </table>	Field Name	Data Type	OBJECTID	Object ID	ilixID	Text	URL_Behoerde	Text	URL_Kataster	Text	Bemerkung	Text	<p>Cette classe est liée à la table principale via une association de cardinalité 1:n. Autrement dit, un élément de <i>ZustaendigkeitKataster</i> peut être associé à n éléments de <i>Belasteter_Standorte</i>. Une clé étrangère est donc ajoutée à la classe <i>Belasteter_Standorte</i>.</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 et PostGIS 2.1.1 ont été utilisé pour générer le modèle de cet exemple. Le code SQL permettant de générer le modèle est disponible en annexe.

La définition du modèle de données pour la base PostgreSQL/PostGIS s'effectue de manière similaire à celle définie dans le chapitre 5.4.2.1 pour une fGDB ESRI. Quatre tables sans géométries sont créées ainsi qu'une table géométrique de type polygone (*belasteter\_standort\_geo\_lage\_polygon*) et une de type point (*belasteter\_standort\_geo\_lage\_punkt*).

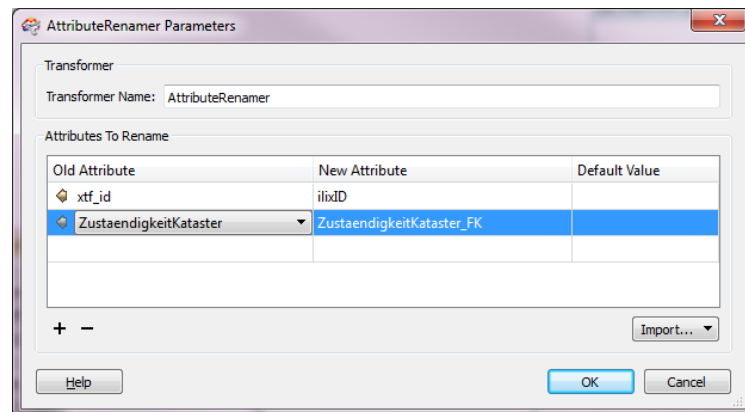


Notons toutefois que PostgreSQL impose l'utilisation de minuscules dans les noms de classe et d'attribut.

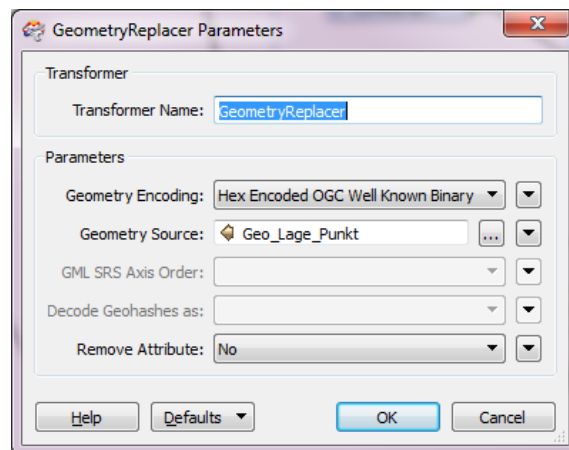
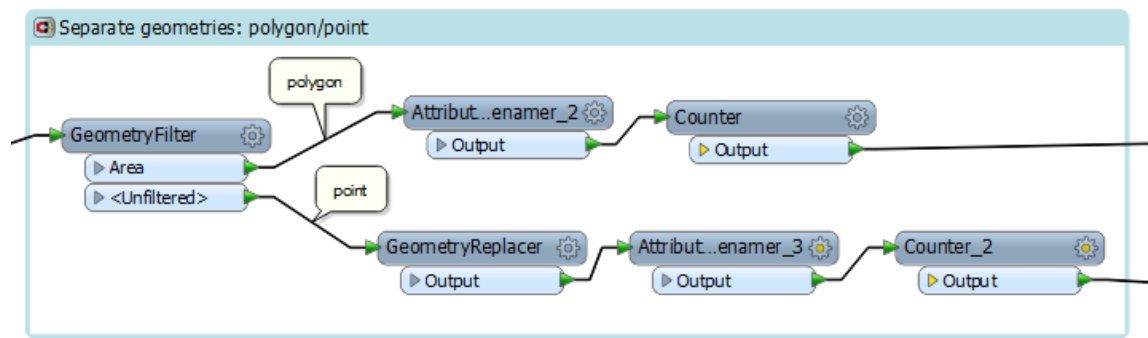
### 5.4.3 Transformation INTERLIS → fGDB

Ce chapitre fait référence au workspace *w-Doc\_ili2fme\_Ex4\_ili2fGDB-isb.fmw*.

1. Dans la feature class *Belasteter\_Standort* du reader INTERLIS, on observe que l'attribut *ZustaendigkeitKataster* a été automatiquement ajouté par le plugin *ili2fme*. Cet attribut est la clé étrangère qui matérialise l'association *ZustaendigkeitKatasterBelasteter\_Standort*. Cet attribut doit être renommé en *ZustaendigkeitKatasterBelasteter\_Standort\_FK* pour correspondre à l'attribut défini dans la base de données en sortie.

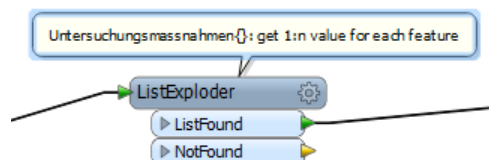


2. A partir de la feature class *Belasteter\_Standort* du reader INTERLIS, on a vu précédemment que les géométries doivent être séparées pour être chargées dans deux *feature class* ESRI, selon que la géométrie soit de type point ou polygone.



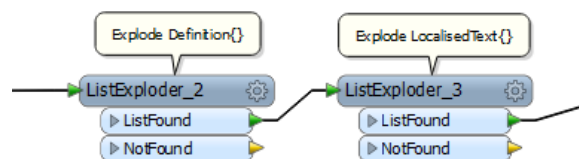
Le *GeometryFilter* permet d'effectuer cette séparation. Le *GeometryReplacer* permet ensuite de récupérer la géométrie de type point qui est stockée dans l'attribut *Geo\_Lage\_Punkt*.

- On a vu que chaque entité de la classe *Belasteter\_Standort* peut avoir plusieurs valeurs pour l'attribut *Untersuchungsmassnahmen*. Afin de récupérer toutes les valeurs, il faut exploser la liste *Untersuchungsmassnahmen* qui a été créée dans le reader par le plugin *ili2fme* :



On obtient ainsi les relations qui permettent de renseigner la table *Rel\_UntersMassn*.

- Finalement, le processus récupère les différentes significations des codes selon la langue, qui ont également été stockées dans des listes de la feature class *Untersuchungsmassnahmen\_Definition* par le plugin *ili2fme*.



Le nom des listes (*Definition* et *LocalisedText*) qui sont créées par *ili2fme* proviennent directement de la structure qui est définie dans les modèles INTERLIS. Ainsi, dans *KbS\_LV03\_V1\_2\_is.ili*, on retrouve :

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

Puis, en allant consulter la structure MultilingualText dans le modèle LocalisationCH\_V1 :

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

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

Le manuel d'utilisateur, publié par swisstopo [10], donne une explication détaillée de la manière d'intégrer les modèles minimaux dans une modélisation INTERLIS.

#### 5.4.4 Transformation INTERLIS → PostGIS

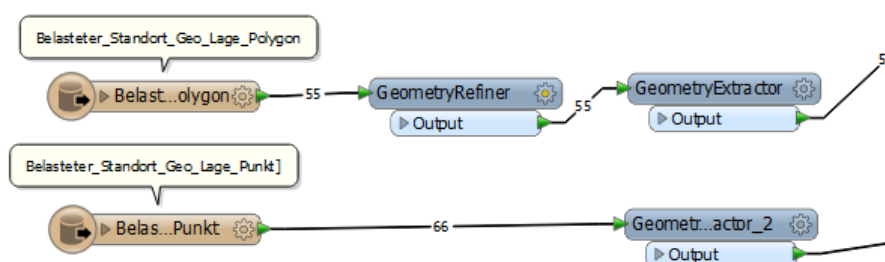
Ce chapitre fait référence au workspace *w-Doc\_ili2fme\_Ex4\_ili2PostGIS-isa.fmw*.

La procédure de transfert vers une base PostGIS est très similaire à celle développée ci-dessus. Seuls les writers changent. De ce fait, le workspace n'est pas décrit dans le présent rapport, mais est laissé à la disposition du lecteur.

#### 5.4.5 Transformation fGDB → INTERLIS

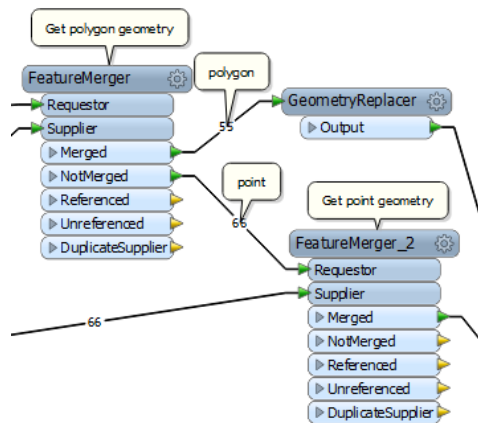
Ce chapitre fait référence au workspace *w-Doc\_ili2fme\_Ex4\_fGDB2ili-isb.fmw*.

1. A partir des deux feature class géométriques de la fGDB (*Belasteter\_Standort\_Geo\_Lage\_Polygon* et *Belasteter\_Standort\_Geo\_Lage\_Punkt*), le processus commence par stocker les géométries dans des attributs à l'aide de deux *GeometryExtractor*.

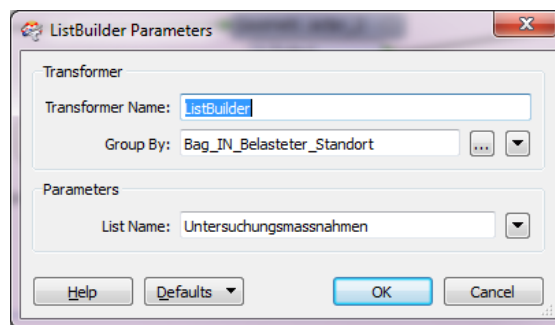


Dans cet exemple, nous utilisons le type d'encoding *FME\_XML*. On aperçoit également que le transformer *GeometryRefiner* a été ajouté. Il permet d'effectuer une simplification de la définition de la géométrie lorsque celle-ci peut poser problème lors de traitements ultérieurs.

2. Ces attributs contenant la géométrie (*Geo\_Lage\_Polygon* et *Geo\_Lage\_Punkt*) doivent ensuite être joints à l'objet auquel ils font référence dans la classe principale *Belasteter\_Standort*. Les deux transformers *FeatureMerger* et *FeatureMerger\_2* effectuent cette opération.



Par ailleurs, il faut reconstruire la liste *Untersuchungsmassnahmen*{}, à partir de la table de relations *Rel\_UntersMassn*. En ayant pris soin d'avoir préalablement renseigné correctement l'attribut *xtf\_class*. Un *ListBuilder* permet de reconstruire cette liste. On effectue un *GroupBy* selon la clé étrangère, qui fait référence à la clé primaire de la classe principale *Belasteter\_Standort*. On obtient ainsi une liste par entité de cette classe. Un *FeatureMerger* permet ensuite de joindre la liste à l'entité en question.



3. Finalement, il faut également reconstruire les listes pour correspondre à la structure définie pour la classe *Untersuchungsmassnahmen\_Definition*. En faisant exactement l'inverse qu'effectué au point 3 du chapitre 5.4.3, on commence par reconstruire la liste *LocalisedText*{}, qui groupe pour chaque *Code* les significations dans les trois langues. Puis, on remonte la structure en construisant la liste *Definition*{}, qui est incluse dans la classe *Untersuchungsmassnahmen\_Definition*. C'est sous cette forme que le plugin *ili2fme* pourra correctement écrire dans le fichier XML. Ci-dessous, on voit un extrait du fichier *.xtf* qui a été écrit par le processus.

```
<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>
```

On remarque que la construction dans FME est faite « depuis le centre ». On commence par construire les listes *LocalisedText{}* en rouge, puis *Definition{}* en vert. A chaque étape, il est important de renseigner l'attribut *xtf\_class* qui permet de faire le lien avec le modèle conceptuel.

#### 5.4.6 Transformation INTERLIS →PostGIS

Ce chapitre fait référence au workspace *w-Doc\_ili2fme\_Ex4\_PostGIS2ili-isa.fmw*.

La procédure d'écriture en INTERLIS depuis une base PostGIS est très similaire à celle développée ci-dessus. Seuls les writers changent. De ce fait, le workspace n'est pas décrit dans le présent rapport, mais est laissé à la disposition du lecteur.



## 6. FAQ

1. Que sont les BASKET ? Et comment doivent-ils être utilisés ?  
*Ce sont des conteneurs qui sont utilisés pour structurer le fichier XML INTERLIS 2. Lorsqu'on écrit en INTERLIS 2, il faut donc créer un BASKET par TOPIC. Puis, il faut que chaque feature type soit liée à un BASKET à l'aide de l'attribut xtf\_basket.*
2. Est-ce possible de convertir un modèle INTERLIS 1 en INTERLIS 2 ? Et des données ?  
*Oui, il existe l'outil ITF2XML, développé par infoGrips GmbH, qui permet d'obtenir un fichier XML ainsi que le modèle en INTERLIS 2 à partir d'un fichier ITF et un modèle en INTERLIS 1.*  
<http://www.interlis.ch/general/itf2xml.php?language=f>
3. Est-ce qu'il existe en INTERLIS 2 des contraintes sur les noms d'attribut ou les noms de table ?  
*INTERLIS 2 ne supporte que les noms ne contenant pas de symbole. De plus, certains noms réservés ne doivent pas être utilisés, comme BOOLEAN, END, IN, NAME, MODEL, NULL, OID, PARAMETER, THIS, SURFACE. Il existe une liste exhaustive de noms réservés en INTERLIS 2, disponible au chapitre 2.2.7 du manuel de référence pour INTERLIS 2 [1].*
4. Quelles géométries sont supportées en INTERLIS 2 ?  
*Les types de géométrie qui existent en INTERLIS 2 sont les suivants :*

- Les points, qui peuvent être définis en 2D ou 3D :

```
COORD
480000.000 .. 840000.000 [m],
7000.000 .. 300000.000 [m],
200.000 .. 5000.000 [m];
```

- Les lignes sont définies en POLYLINE. Le mot clé ARCS permet d'autoriser des lignes de type arc :

```
POLYLINE WITH (STRAIGHTS, ARCS) VERTEX Coorddef;
```

- Les polygones peuvent être de type SURFACE ou AREA. Le type SURFACE autorise les superpositions d'objets au sein d'une classe, au contraire de AREA.

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

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

*Le mot clé ARCS permet également d'autoriser les arcs.*

*L'extension WITHOUT\_OVERLAPS > «Tolerance» permet de définir une tolérance maximum de superposition au sein d'une classe. A partir d'INTERLIS 2.3, il est obligatoire de définir une valeur de tolérance non nulle pour ces types de géométrie.*

5. Comment sont gérées les géométries multi-parts ?  
*Les géométries multi-part ne sont nativement pas supportées en INTERLIS. Afin de remédier à cette limitation, un type de géométrie multi-part est défini dans les briques de modélisation CH-Base, qui est une agrégation de types de géométries de base.*
6. Comment peut-on s'assurer qu'aucun polygone n'a été perdu pendant un traitement ?

De manière générale, il est d'abord conseillé de vérifier les messages de type warning dans le log du traitement FME.

Par ailleurs, il est possible de comparer le nombre d'occurrences qui existent dans une classe particulière du fichier xtf (en ouvrant le fichier avec un éditeur de texte) avec le nombre de polygones créés dans FME.

7. Est-ce que la géométrie des objets peut être modifiée lors d'un traitement FME ?  
Concernant le plugin ili2fme : non, les reader et writer INTERLIS 2 ne modifient pas la géométrie des objets.

Dans le cas où un traitement convertit des données INTERLIS dans un autre format, il est possible que des contraintes liées au format d'écriture (précision, résolution, tolérance, ...) puissent affecter les géométries initiales.

8. Est-ce qu'INTERLIS permet de définir des règles de topologie ?  
Dans le modèle INTERLIS 2, il est possible de définir des règles de non superposition au-delà d'une certaine tolérance pour des objets de type SURFACE :

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

Cependant, le plugin ili2fme ne vérifie pas que les données en lecture ou en écriture soient conformes à cette définition. Un checker INTERLIS remplit toutefois ce rôle.

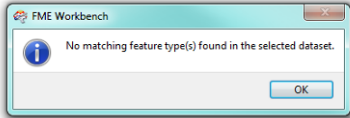
9. Lorsqu'un modèle INTERLIS fait appel à d'autres modèles (Units, CoordSys, Time, Symbology, CHBase), comment faire pour que ces modèles soient accessibles ?

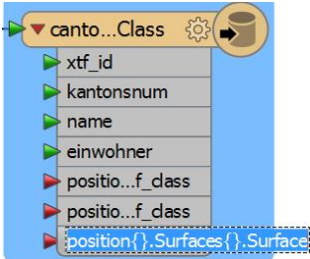
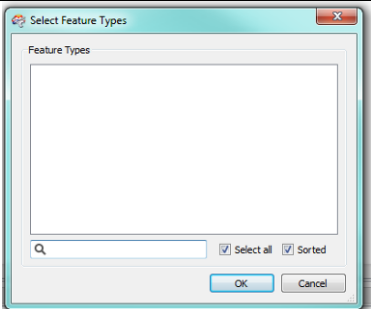
Deux méthodes sont à différencier :

- Online : si une connexion Internet est disponible et qu'aucune contrainte de sécurité ne l'empêche, tous les modèles minimaux sont accessibles à l'adresse suivante : [models.interlis.ch](http://models.interlis.ch).
- En local : les modèles minimaux sont stockés dans des répertoires particuliers.

Le paramètre « Models directory » du plugin ili2fme permet de définir l'emplacement des modèles minimaux (c.f. chapitre 3.3).

10. Exemple de problèmes fréquemment rencontrés avec ili2fme :

Problème	Cause	Solution
<i>Model(s) not found</i> ou : 	<ul style="list-style-type: none"><li>- Non correspondance entre les modèles inscrit dans le fichier de données XTF (entête du document) et le nom des modèles de référence (changement de nom, de version)</li><li>- Modèles non présents dans les répertoires de référence</li></ul>	<ul style="list-style-type: none"><li>- Vérifier les noms de modèles en ouvrant les fichiers XTF et ILI dans un éditeur de texte</li><li>- Modifier les chemins d'accès dans <i>Parameters</i> de FME (cf. chapitre 3.3)</li></ul>
Type de données non respecté	<ul style="list-style-type: none"><li>- Le processus développé dans FME pour générer le fichier de données XTF n'a pas pris en compte ces contraintes. Attention, le plugin ili2fme écrit les données, sans vérifier leurs correspondances avec le modèle.</li></ul>	<ul style="list-style-type: none"><li>- Appliquer des transformations dans le workspace FME pour correspondre au modèle</li><li>- Vérifier le fichier de données avec le checker infoGrips</li></ul>

Données obligatoires non enregistrées	<ul style="list-style-type: none"> <li>- Certains attributs obligatoires (MANDATORY) n'ont pas été renseignés dans le workspace. Attention, le plugin <i>ili2fme</i> écrit les données, sans vérification.</li> </ul>	<ul style="list-style-type: none"> <li>- Créer ces attributs dans le workspace FME pour correspondre au modèle.</li> <li>- Vérifier le fichier de données avec le checker infoGrips.</li> </ul>
Non-respect de la structure de classe dans les listes FME 	<ul style="list-style-type: none"> <li>- Le workspace doit contenir la création successive de listes pour correspondre aux classes. Si tel n'est pas le cas, les données ne seront pas inscrites. Attention, il arrive parfois que certains attributs du writer restent rouges, tant bien même que la structure des données a été correctement effectuée. Dans ce cas, les données s'enregistrent correctement.</li> </ul>	<ul style="list-style-type: none"> <li>- Respecter la structure du modèle en créant une liste par classe et en y intégrant le <i>xtf_id</i>, <i>xtf_class</i>, ainsi que les autres attributs constituant la classe. Commencer par la classe qui a le niveau le plus élevé afin de l'inclure dans les classes suivantes. Dans l'exemple présent Surface doit être créée en premier, puis être incluse dans position.</li> </ul>
Translation was SUCCESSFUL mais le fichier XTF est vide	La cause peut être multiple (liste non exhaustive) : <ul style="list-style-type: none"> <li>- Pas de création des attributs <i>xtf_basket</i> par classe</li> <li>- Pas de création de la classe <i>XTF_BASKETS</i></li> <li>- Mauvaise définition de l'attribut <i>xtf_basket</i></li> </ul>	<ul style="list-style-type: none"> <li>- Créer ces attributs ou classe</li> </ul>
Problème de géométrie <i>@Geometry function could not parse ...</i>	<ul style="list-style-type: none"> <li>- l'encodage de la géométrie à l'aide d'un <i>GeometryExtractor</i> ne correspond pas à l'encodage spécifié dans les paramètres du writer</li> </ul>	Faire correspondre les deux types d'encodage
	Lors de l'import d'un modèle de données, la fenêtre de sélection des classes est vide. Le problème provient d'une erreur dans le modèle de données <i>ili</i> .	Corriger le modèle