1. INTERLIS 2 Reader/Writer for FME

The INTERLIS 2 reader and writer module (ili2fme) provides the Feature Manipulation Engine (FME) with access to INTERLIS 2 and INTERLIS 1 transfer files.

ili2fme is in stable state.

This documentation describes version 5.2.x of ili2fme. The current version of ili2fme can be found at http://www.eisenhutinformatik.ch/interlis/ili2fme/.

This chapter assumes you are familiar with FME and the INTERLIS 1 and 2 formats. For more information about FME, please read the FME documentation. For more information about INTERLIS, go to http://www.interlis.ch.

Please send comments about ili2fme to ce@eisenhutinformatik.ch.

1 Overview

Features read from an INTERLIS file consist of a series of attribute values. They may have no geometry. The attribute names are as defined in the INTERLIS model. The feature type of each INTERLIS feature is the qualified INTERLIS name (for INTERLIS 2: the qualified name of the class, for INTERLIS 1: the qualified name of the table). The mapping of the inheritance hierarchy is done with a super or sub-type strategy.

ili2fme supports the rich geometry model of FME.

ili2fme can read and write INTERLIS 1 and 2. However, in most cases you will need a FME script or workbench to write INTERLIS.

2 INTERLIS 2 Quick Facts

Format Type Identifier	ch.ehi.fme.Main
Long Format Name	Swiss INTERLIS (ili2fme)
Reader/Writer	Both
Dataset Type	File
Feature Type	Class name
Typical File Extensions	.xtf, .xml, .itf, .ili
Automated Translation Support	Automated reading
User-Defined Attributes	Yes
Coordinate System Support	No

Generic Color Support	No
Spatial Index	Never
Schema Required	Yes
Transaction Support	No
Geometry Type Attribute	xtf_geomtype

Geometry Support (classic geometry model)

Geometry	Supported	Geometry	Supported
aggregate	no	polygon	yes
circles	stroked	donut polygon	yes
circular arc	stroked	line	yes
elliptical arc	stroked	point	yes
ellipses	stroked	text	no
none	yes	3D	yes

Geometry Support (rich geometry model)

Geometry	Supported	Geometry	Supported
aggregate	no	polygon	yes
circles	stroked	donut polygon	yes
circular arc	yes	line	yes
elliptical arc	stroked	point	yes
ellipses	stroked	text	no
none	yes	3D	yes

3 Reading INTERLIS 1-Data

To read INTERLIS 1-data, the Model (.ili) must be known to FME.
It can be stored:

- in $(FME)\plugins\interlis2\ilimodels
- in a special model directory you specify
- in the same directory than your data

Then you can select an INTERLIS 1-datafile (.itf) and open it with FME (Viewer, Workbench, Universal Translator) and use it.

- All the enumerations from the ITFs will be converted to texts (values).
- If more than one geometry exists, the first geometry will be used as FME geometry, the other ones will be stored as Hex Well Known Binary in Attributes.

4 Reading INTERLIS 2-Data

Reading INTERLIS 2-data is essentially the same than reading INTERLIS 1-data with the following differences:

- The data comes in XTF-files (and not ITF-files)
- If your data models contain EXTENDS, FME will show all the data in a single "superstructure" – feature type. You will have to use an AttributeFilter on XTF_CLASS to separate the different classes in Workbench. Since ili2fme-4.4.0, the data model may also be imported with a "subclass"-strategy rather than a “superclass”-strategy. When "subclass" is chosen, a feature type is created for each concrete extended class, whereas one feature type is created per parent class when "superclass" is chosen.

5 Writing INTERLIS 1-Data

To write INTERLIS 1-data, the process is the following
Prerequisites: the INTERLIS model (.ili) has to exist before!

- Set up a Workbench
- Define an "Swiss INTERLIS (ili2fme)” destination dataset
- Import the feature type definitions from your ILI-model (Destination Data -> Import FeatureTypes -> Browse to your ILI-file)
- Define a transfer identification for each feature, by setting the format attribute "xtf_id" (e.g. generate it with a counter or map a format attribute like OBJECTID / FID or similar)
- Route your features to the destination feature types (connect the arrows)
- GO!

Please note, that:

- If the “xtf_class” format attribute is set, its value supersedes the name of the feature type. This may lead to unexpected results if your features come from an INTERLIS dataset, and “xtf_class” is still set (to the source class instead of the target class).

6 Writing INTERLIS 2-Data

To write out INTERLIS 2-data, you will have to follow these steps in addition to the ones explained for INTERLIS 1:

- Create one feature of feature type “XTF_BASKETS” for each TOPIC (With a Creator / NullGeometryCreator + AttributeCreator)
- Reference this basket in each feature type of the topic, by setting the format attribute "xtf_basket" (e.g. by attaching a constant).
- Write all herited classes to a "superstructure" feature type. (or choose a subclass-strategy)
- Define the qualified INTERLIS class name of each class, by setting the format attribute "xtf_class" in each feature type

Please note, that:

- “XTF_BASKETS” features must be created by hand in a common transformation with an INTERLIS 2 writer.
- “xtf_basket” format attributes must be set by hand in a common transformation with an INTERLIS 2 writer.
- “xtf_id” format attributes must be set/mapped in a common transformation with an INTERLIS 2 writer.

- You always need to provide fully qualified class names of the target INTERLIS model. For example, the correct parameter might be: "Fallbeispiel.Raumplanung.Bauzone".
- If the “xtf_class” format attribute is set, its value supersedes the name of the feature type. This may lead to unexpected results if your features come from an INTERLIS dataset, and “xtf_class” is still set (to the source class instead of the target class).

7 Writing GML-Data

Starting with version 5.0.0 ili2fme is able to write GML, according to the ILIGML specification. To write out GML, just follow the steps explained for INTERLIS 2, but select a file to write with extension ".gml".

8 Reading and writing INTERLIS

When you read and write INTERLIS data, read the sections on reading and writing. In addition, you always (even if writing INTERLIS 1) have to

- set the xtf_class format attribute on every destination feature type to the qualified INTERLIS class name (use an AttributeCreator transformer)!

9 Reader Reference

FME considers an INTERLIS transfer file to be a collection of features. The feature types are determined by scanning the transfer file and then reading the appropriate INTERLIS model/schema files. The model files have the extension .ili and should be located in the same folder as the transfer file and/or in the folder ${FME}/plugins/interlis2/ilimodels. A transfer file may need multiple model files. There are no DEF lines required.

9.1 Reader Keywords

The following table lists the keywords processed by the INTERLIS reader. The table shows only the suffixes prefixed by the current <ReaderKeyword> in a mapping file.

Keyword Suffix	*Value*
MODELS	Required INTERLIS-models to read the dataset (the model name, not the .ili-filename; separated by semicolons ‘;’). Or the special value %DATA, in which case the models are determined by inspecting the transfer file. Default Value: %DATA
MODEL_DIR	Folder or (remote) model repositories containing the .ili-files. These files are scanned for INTERLIS-models. You may use %XTF_DIR as placeholder for the directory of the data file that you will read. Multiple directories/repositories may be separated by semicolons ‘;’. Default Value: unset/empty
GEOMETRY_ENCODING	This keyword defines the encoding of geometry attributes, which are not used as FME geometry (Only the first geometry

	attribute becomes a FME geometry). • FMEXML encodes as FME XML • FMEBIN encodes as FME Binary • FMEHEXBIN encodes as FME Hex Binary • OGCHEXBIN encodes as OGC Hex Binary Default Value: OGCHEXBIN
CHECK_UNIQUEOID	This keyword defines the checking of TIDs/OIDs. If set to True, ili2fme will check if TIDs/OIDs are unique. If set to False, ili2fme will by-pass this check. Default Value: True
CREATE_LINETABLES	This keyword applies only to INTERLIS 1 datasets with INTERLIS AREA or INTERLIS SURFACE attributes. If set to True, ili2fme will create two additional feature types for each INTERLIS SURFACE or AREA attribute. One with the ending "_MT" containing the main table data as it appears in the transfer-file. The other additional feature type with the ending "_LT" containing the line helper table as it appears in the transfer-file. If set to False, ili2fme will create no additional tables. Default Value: False
SKIP_POLYGONBUILDING	This keyword applies only to INTERLIS 1 datasets with INTERLIS AREA or INTERLIS SURFACE attributes. If set to True, ili2fme will not build polygons from the line tables as they appear in the transfer-file. If set to False, ili2fme will build polygons from the line tables and the (geo)-references as they appear in the transfer-file. Default Value: False
ILI1_ADDDEFVAL	This keyword applies only to INTERLIS 1 datasets. If set to True, ili2fme will parse the explanation at the end of attribute definitions that are optional. If there is no attribute value in the data, ili2fme will add the one given in the model. The syntax rule is: `'//' 'undefiniert' '=' Constant | 'letztes' 'Zeichen' '//'.` If the value in the model is `'letztes' 'Zeichen'` ili2fme will follow the first

	reference attribute of this table, and use the length of the value of the first text attribute in the target table. If set to False, ili2fme will not supply any default values to the data. OPEN: Example Default Value: False
ILI1_ENUMASITFCODE	This keyword applies only to INTERLIS 1 datasets. If set to True, ili2fme will read values of attributes of type enumeration as numeric code (the same code as it appears in the transfer file). This option is not recommended and exists only for backward compatibility reasons. If set to False, ili2fme will map the code from the transfer file to enumeration element name (the value as it would appear in an INTERLIS 2 transfer file). This option is recommended because it is less error prone and offers compatibility between INTERLIS 1 and 2. Default Value: False
ILI1_RENUMBERTID	This keyword applies only to INTERLIS 1 datasets. If set to True, ili2fme will renumber the objects, so that the TID becomes unique across the whole transfer. If set to False, ili2fme will read the TIDs unchanged. Default Value: False
INHERITANCE_MAPPING	This keyword applies only to INTERLIS 2 datasets. If set to “SuperClass” the superclass inheritance mapping strategy is applied. If set to “SubClass” the subclass inheritance mapping strategy is applied. See the section titled “Inheritance mapping strategy” below for an explanation of the different strategies. Default Value: “SuperClass”
ILI1_CONVERTAREA	This keyword applies only to INTERLIS 1 datasets and if SKIP_POLYGONBUILDING is set to False. The name of a FME pipeline definition file (.fmi), to be used to build the FME polygons from the line helper table features of INTERLIS AREA attributes and main table features as read from the ITF file. ili2fme will set the following macros: • $(lineTableName) name of FME

	• $(mainTableName) name of main FME feature type on input (point) and output (polygon/donut) • $(maxOverlaps) max overlaps as defined by datatype of attribute in INTERLIS model The features that come out of the pipeline should have $(mainTableName) as feature type and polygon or donut as geometry. If the filename is relative, the file is looked after in the folder of the workbench first, and then in $(FME)/plugins/interlis2/converter If the value is not set, ili2fme will use a built-in pipeline. Default Value: unset/empty
ILI1_CONVERTSURFACE	This keyword applies only to INTERLIS 1 datasets and if SKIP_POLYGONBUILDING is set to False. The name of a FME pipeline definition file (.fmi), to be used to build the FME polygons from the line helper table features of INTERLIS SURFACE attributes and from the main table features as read from the ITF file. ili2fme will set the following macros: • $(lineTableName) name of FME feature type with polylines on input • $(mainTableName) name of main FME feature type on input (point) and output (polygon/donut) • $(mainTableRef) name of reference attribute in feature type $(lineTableName) • $(maxOverlaps) max overlaps as defined by datatype of attribute in INTERLIS model The features that come out of the pipeline should have $(mainTableName) as feature type and polygon or donut as geometry. If the filename is relative, the file is looked after in the folder of the workbench first and then in $(FME)/plugins/interlis2/converter If the value is not set, ili2fme will use a built-in pipeline. Default Value: unset/empty
CREATEFEATURETYPE4ENUM	This keyword is used to control how ili2fme creates FME feature types for INTERLIS enumerations: If set to "No", ili2fme will create no feature types for enumerations. If set to "SingleType", ili2fme will create on

	single additional feature type "XTF_ENUMS" and provide each element of all enumeration types as a feature of this feature type. If set to "OneTypePerEnumDef", ili2fme will create one feature type for each enumeration type. This option is useful to setup a ValueMapper factory in the FME workbench. Default Value: No
TRACEMSGS	This setting controls the level of detail of log messages written by ili2fme. If set to True, ili2fme will write detailed progress messages to the log. If set to False, ili2fme will only write normal progress messages to the log. Default Value: False

10 Writer Reference

The INTERLIS writer module stores features into an INTERLIS transfer file. The required models are determined by scanning the features and then reading the appropriate INTERLIS model/schema files. The model files have the extension .ili and should be located in the same folder as the transfer file and/or in the folder ${FME}/plugins/interlis2/ilimodels. A transfer file may need multiple model files. There are no DEF lines required.

The appropriate feature types are expected by the writer, as if the same model would have been read by the INTERLIS 2 reader.

This version of the Writer requires that the INTERLIS model already exists. The model can not be generated by the Writer. Therefore, in most cases you will need a FME script or workbench to prepare the expected feature types.

10.1 Writer Keywords

The following table lists the keywords processed by the INTERLIS 2 writer. The table shows only the suffixes which will be prefixed by the current `<WriterKeyword>` in a mapping file.

Keyword Suffix	*Value*
MODELS	Required INTERLIS-models to write the dataset (the model name, not the .ili-filename; separated by semicolons ';'). Or the special value %DATA, in which case the models are determined by inspecting the FME features. Default Value: %DATA
MODEL_DIR	Folder or (remote) model repositories containing the .ili-Files. These files are scanned for INTERLIS-Models. You may use %XTF_DIR as placeholder for

	the directory of the data file that you will write. Multiple directories/repositories may be separated by semicolons ';'. Default Value: unset/empty
CHECK_UNIQUEOID	This keyword defines the checking of TIDs/OIDs. If set to True, ili2fme will check if TIDs/OIDs are unique. If set to False, ili2fme will by-pass this check. Default Value: True
GEOMETRY_ENCODING	This keyword defines the encoding of geometry attributes, which are not used as FME geometry (Only the first geometry attribute becomes a FME geometry). • FMEXML encodes as FME XML • FMEBIN encodes as FME Binary • FMEHEXBIN encodes as FME Hex Binary • OGCHEXBIN encodes as OGC Hex Binary Default Value: OGCHEXBIN
INHERITANCE_MAPPING	This keyword applies only to INTERLIS 2 datasets. If set to "SuperClass" the superclass inheritance mapping strategy is applied. If set to "SubClass" the subclass inheritance mapping strategy is applied. See the section titled "Inheritance mapping strategy" below for an explanation of the different strategies. Default Value: "SuperClass"
USE_LINETABLES	This keyword applies only to INTERLIS 1 datasets with INTERLIS AREA or INTERLIS SURFACE attributes. If set to True, ili2fme will expect one additional feature type for each INTERLIS SURFACE or AREA attribute. The additional feature type with the ending "_$(attributeName)" contains the line helper features as they should appear in the transfer-file. If set to False, ili2fme will create the line helper table out of the polygons/donuts. Default Value: False
TRACEMSGS	This setting controls the level of detail of log messages written by ili2fme. If set to True, ili2fme will write detailed progress messages to the log. If set to False, ili2fme will only write normal progress messages to the log.

	Default Value: False

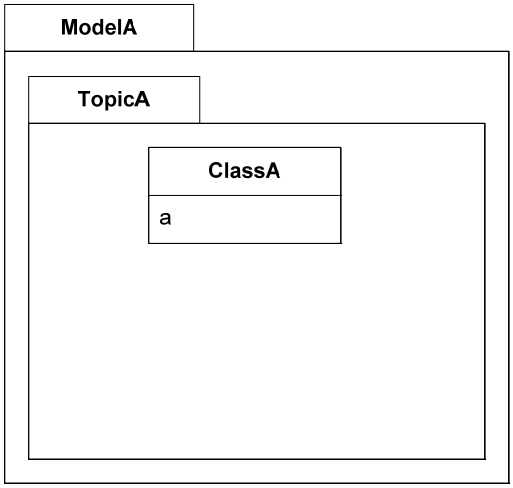
11 Feature Representation

The following clauses describe how ili2fme maps INTERLIS objects to FME features. Features written to the INTERLIS transfer file are expected to have the same structure, as they would have had when read.

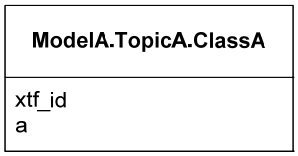
11.1 Overview

11.1.1 INTERLIS 1

INTERLIS allows for some nesting of type definitions. A class or table is defined in a topic. Several topics are grouped to a model. FME doesn't allow such a nesting. Therefore ili2fme maps INTERLIS class with their qualified name to FME feature types.



INTERLIS model

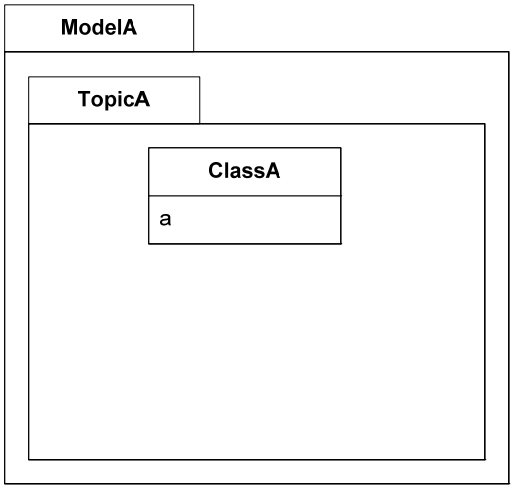


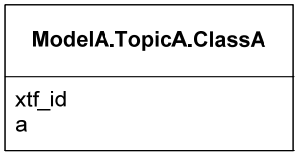
FME feature schema

Each FME feature type has a format attribute "xtf_id" that is the transfer identification of that feature in the ITF file.

11.1.2 INTERLIS 2 full transfer mode

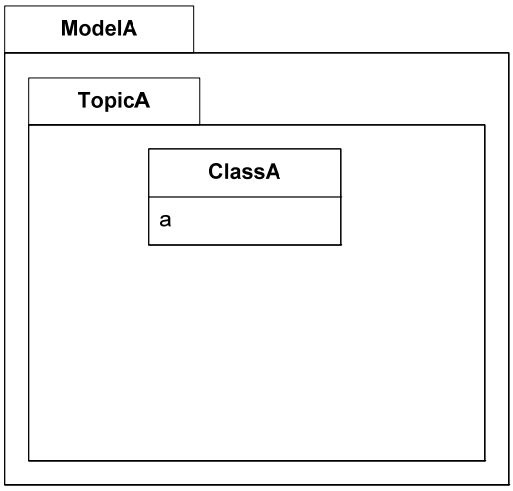
For INTERLIS 2 the mapping is the same as for INTERLIS 1, but only if there are no extended topics in the INTERLIS model and there is only one basket per topic in the transfer file.

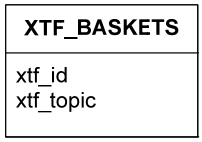


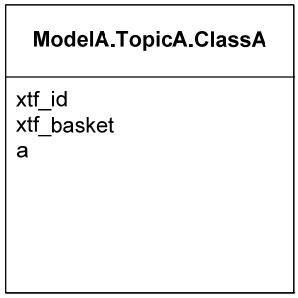


INTERLIS model **FME feature schema**

If an INTERLIS 2 data file has multiple baskets (instances of a topic; set of objects) of the same topic or the model has extended topics, additional format attributes are required.





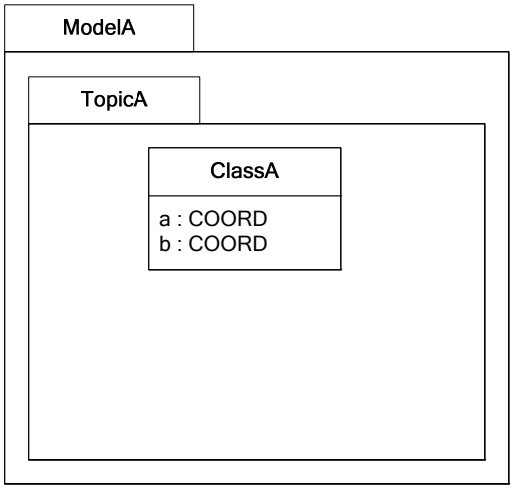


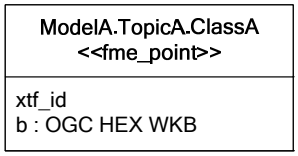
INTERLIS model **FME feature schema**

To know which feature belongs to which basket, each feature has a reference to the basket in the format attribute “xtf_basket”. Each basket is represented as an instance of the format feature type “XTF_BASKETS”. The attribute “xtf_topic” holds the qualified topic name that describes this basket (in this case that would be “ModelA.TopicA”). The attribute “xtf_id” of the feature type “XTF_BASKETS” is the transfer identification of the basket.

11.2 Multiple geometries per class

An INTERLIS class may define multiple attributes of type geometry.





INTERLIS model

FME feature schema

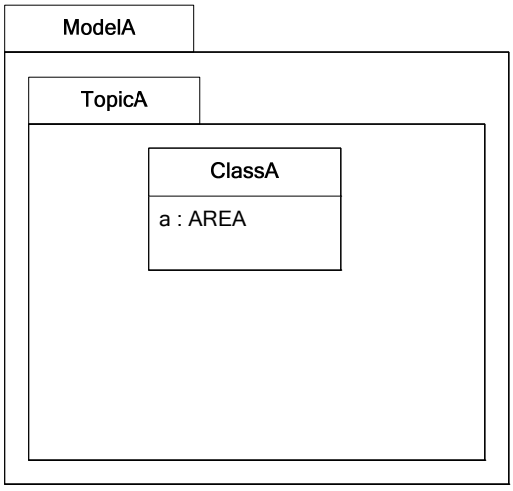
ili2fme maps the first geometry of the INTERLIS class to the FME geometry of the feature. Any additional INTERLIS geometry attributes are mapped to ordinary FME attributes. The value of these attribute (in this case the attribute “b”) is HEX encoded OGC WKB (this can be changed with the parameter GEOMETRY_ENCODING) and can be extracted from that attribute to the feature geometry with the transformer “GeometryReplacer” or set with “GeometryExtractor”.

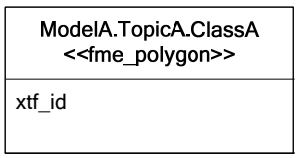
11.3 INTERLIS 1 AREA

INTERLIS 1 encodes attributes of type AREA in helper table prior to the main table. ili2fme can read these attributes in three modes:

- build polygons/donuts automatically from the line table
- read the main table and the line table as they are in the transfer file
- combination of the two cases above

With automatic polygon building the mapping is as follows

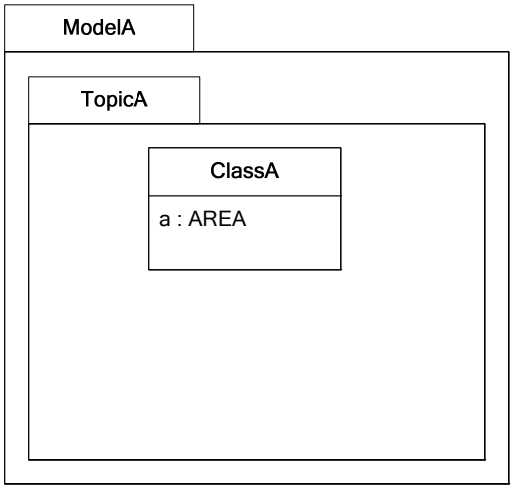


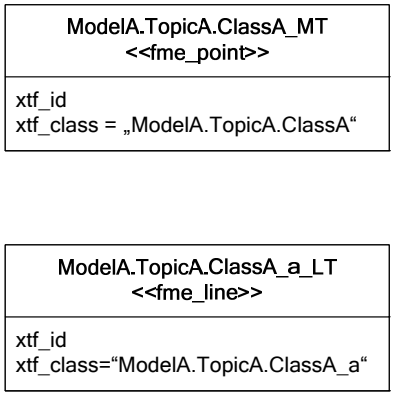


INTERLIS model

FME feature schema

With automatic polygon build disabled, the mapping is as follows





INTERLIS model

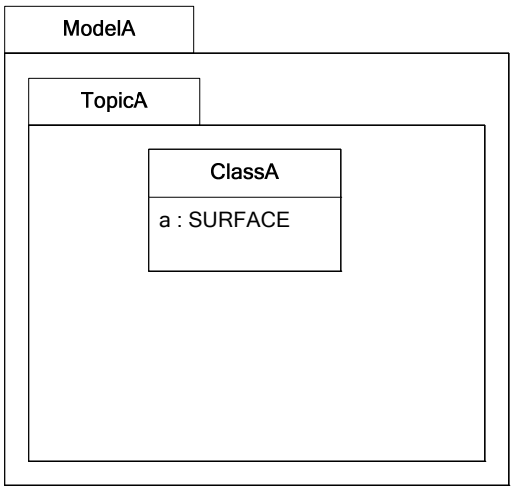
FME feature schema

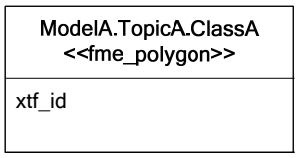
11.4 INTERLIS 1 SURFACE

INTERLIS 1 encodes attributes of type SURFACE in helper table following the main table. ili2fme can read these attributes in three modes:

- build polygons/donuts automatically from the line table
- read the main table and the line table as they are in the transfer file
- combination of the two cases above

With automatic polygon building the mapping is as follows

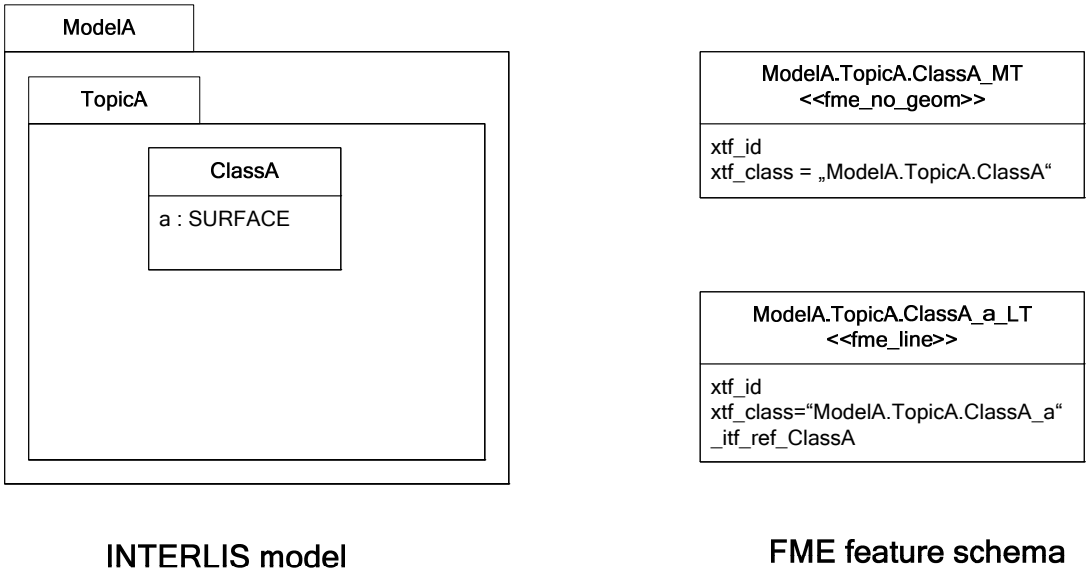




INTERLIS model

FME feature schema

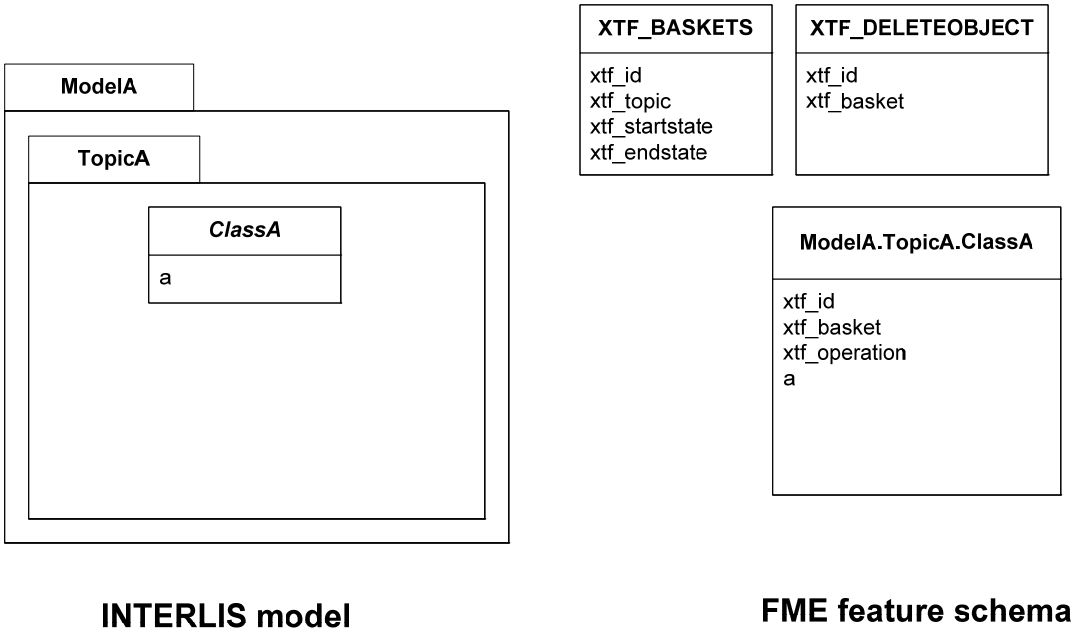
With automatic polygon build disabled, the mapping is as follows



The line table (“ModelA.TopicA.ClassA_a_LT”) gets an additional attribute (with the name of the main class; in this case “_itf_ref_ClassA”) that is a reference from the lines to the feature in the main table (“ModelA.TopicA.ClassA_MT”)

11.5 INTERLIS 2 incremental transfer

INTERLIS 2 supports incremental transfers (change only transfers). Incremental transfer happens per basket. There are two kind of incremental transfers: INITIAL and UPDATE. INITIAL ist the first transfer in a serie of transfers. It includes all objects. UPDATE is used for all succeeding transfers follwing INITIAL and includes only changed objects since the last transfer. Both kinds require additional format attributes.



For an INITIAL data transfer, the XTF_BASKETS feature that represents the basket has a value in the “xtf_endstate” attribute. The “xtf_startstate” attribute should not be set. There are no “XTF_DELETEOBJECT” features. The “xtf_operation” attribute should not be set.

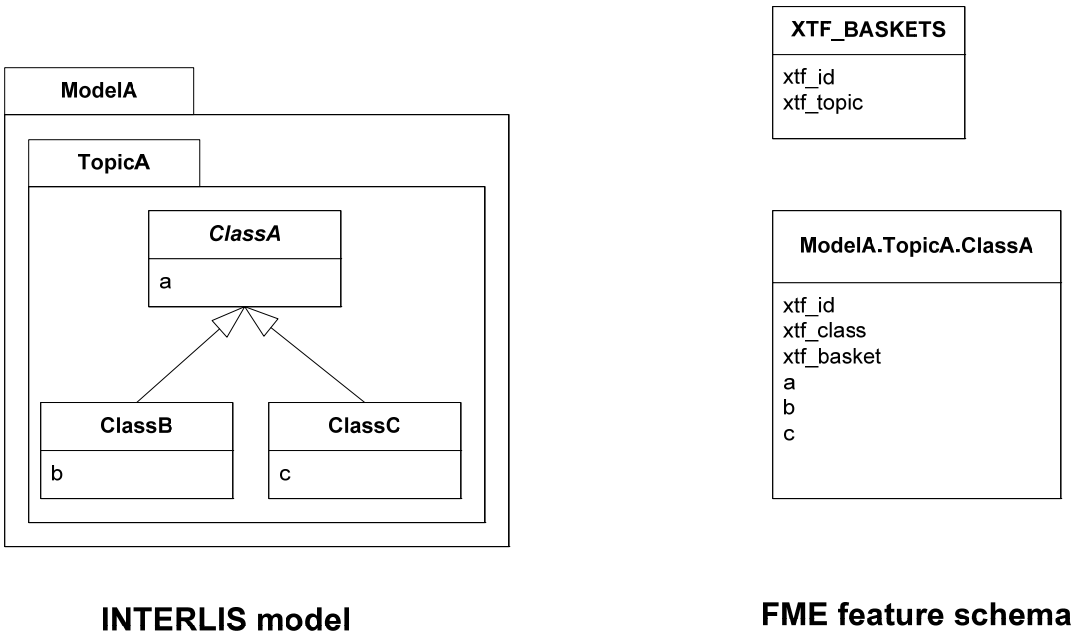
For an UPDATE data transfer, the XTF_BASKETS feature that represents the basket has a value in the “xtf_startstate” and the “xtf_endstate” attribute. The “xtf_startstate” value is the same as the “xtf_endstate” of the last transfer of that basket. The “xtf_operation” attribute should be set to “INSERT”, “UPDATE” or “DELETE”. Instead of mapping deleted objects to ordinary features with “xtf_operation” set to “DELETE”, they may alternatively be mapped to instances of the format feature type “XTF_DELETEOBJECT” (without any INTERLIS attribute values; just “xtf_id” and “xtf_basket”).

11.6 Inheritance mapping strategy

ili2fme supports to inheritance mapping strategies. Depending on you INTERLIS model, one or the other is appropriate.

11.6.1 Superclass strategy

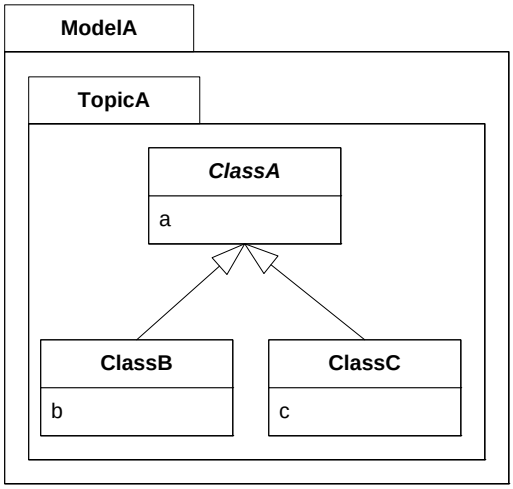
Attributes of non-root classes are shifted to the root, as illustrated by the following figure:

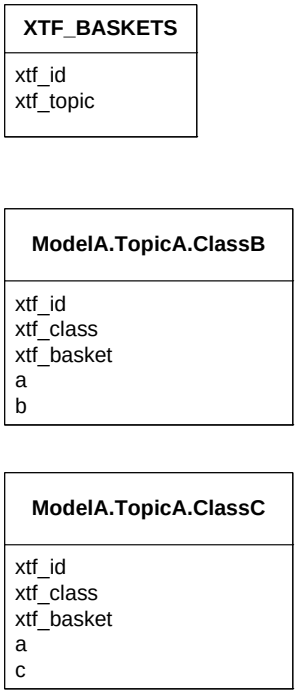


The format attribute „xtf_class“ may be used to determine if a feature is an instance of class „ModelA.TopicA.ClassB“ or class „ModelA.TopicA.ClassC“.

11.6.2 Subclass strategy

Attributes of base classes are shifted to leafs, as illustrated by the following figure:





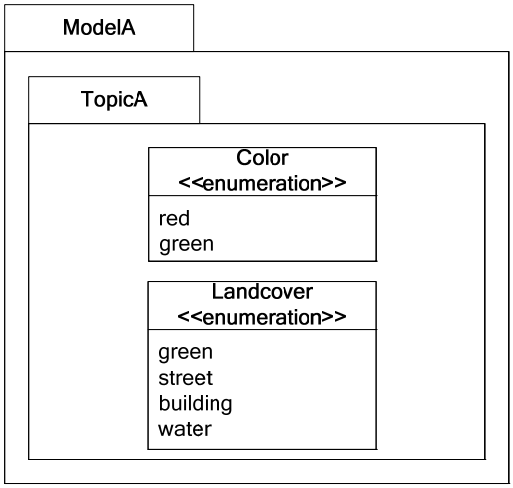
INTERLIS model **FME feature schema**

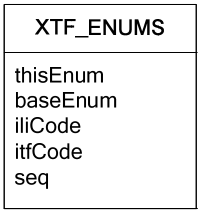
There is no feature type “ModelA.TopicA.ClassA” because it’s an abstract class in the INTERLIS model.

11.7 Enumerations

There are two modes to read enumerations:
"SingleType" will read all elements of all enumerations with the same FME feature type XTF_ENUMS.
"OneTypePerEnumDef" will create one FME feature type for each enumeration type.

11.7.1 Enumerations as one single feature type





INTERLIS model

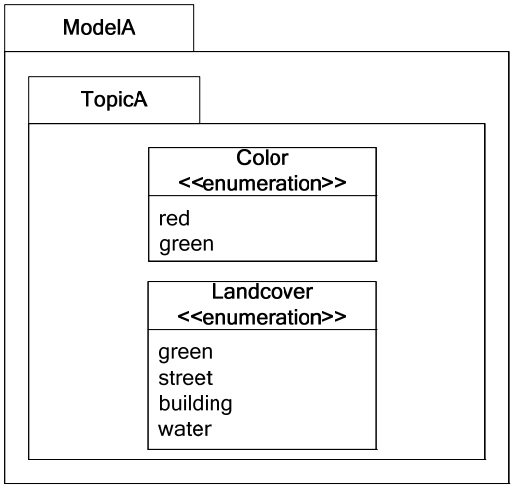
FME feature schema

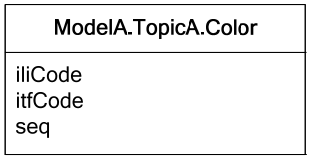
For the feature type "XTF_ENUMS", the following features will be read:

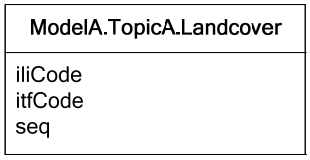
thisEnum	*baseEnum*	*iliCode*	*itfCode*	*seq*
ModelA.TopicA.Color		red	0	
ModelA.TopicA.Color		green	1	
ModelA.TopicA.Landcover		green	0	
ModelA.TopicA.Landcover		street	1	
ModelA.TopicA.Landcover		building	2	
ModelA.TopicA.Landcover		water	3	

The property "baseEnum" is only defined, if the enumeration is an extended one.
The property "seq" is only set, if the enumeration is ordered.

11.7.2 One feature type per enumeration







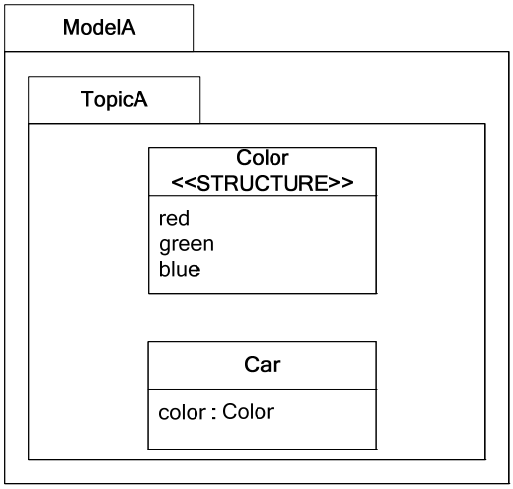
INTERLIS model

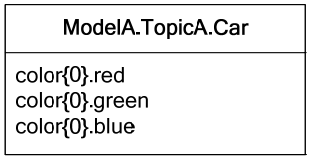
FME feature schema

For the feature type "ModelA.TopicA.Color" the following features will be read:

iliCode	*itfCode*	*seq*
red	0	
green	1	

11.8 BAG/LIST OF





INTERLIS model

FME feature schema

INTERLIS structure attributes (in the example the attribute "color" in the class "Car") are mapped to FME lists. The definition of the INTERLIS structure (in the example the structure "Color") is not mapped as a FME feature type.

11.9 Reference

11.9.1 Format Attributes

In addition to the generic FME feature attributes that FME Workbench adds to all features, this format adds the format-specific attributes described in this section.

Attribute	*Description*
xtf_id	Value of the TID XML-attribute out of the INTERLIS transfer file. Unique across all feature types.
xtf_class	Qualified name of the INTERLIS class name. This is different from the feature type name in the case of non base classes. In the figure above would ModelA.TopicA.ClassB be a possible value. If this value is not set, the feature type name is used as the qualified INTERLIS class name.
xtf_basket	Value of the BID XML-attribute out of the INTERLIS transfer file. May be used as foreign key to a feature of the feature type XTF_BASKET (see below). On writing, this may be used to write multiple baskets of the

	same topic. If writing INTERLIS 1 transfer files, this attribute is not required.
xtf_operation	Only used for incremental INTERLIS 2 transfer. Possible values are: INSERT, UPDATE, DELETE.
xtf_consistency	Only used for somehow modified data. Not yet fully supported.
xtf_geomattr	*Deprecated: Name of the geometry attribute read (e.g. "Geometrie"). An INTERLIS class may define multiple geometry attributes.*

11.9.2 Format features

The reader creates additional feature types, and the writer expects this feature types as well. If writing INTERLIS 1 transfer files, these feature types are not required.

11.9.2.1 Format feature type XTF_BASKETS

Attribute	*Description*
xtf_id	For each basket in the INTERLIS 2 transfer file, the value of the BID XML-attribute.
xtf_topic	Qualified name of the INTERLIS 2 topic name. In the figure above would ModelA.TopicA be a possible value.
xtf_startstate	Only used for incremental INTERLIS 2 transfer. If set, it indicates an UPDATE transfer. It indicates an INITIAL transfer, if it is not set. If it is not an incremental transfer, the value is ignored.
xtf_endstate	Only used for incremental INTERLIS 2 transfer. If set, it indicates an incremental transfer. If it is not set, this is not an incremental transfer.
xtf_consistency	Only used for somehow modified data. Not yet fully supported.

11.9.2.2 Format feature type XTF_DELETEOBJECT

Attribute	*Description*
xtf_id	Value of the TID XML-attribute out of the INTERLIS transfer file. Unique across all feature types.
xtf_basket	Value of the BID XML-attribute out of the INTERLIS transfer file. May be used as foreign key to a feature of the feature type XTF_BASKET. On writing, this may be used to write multiple baskets of the same topic.

11.9.2.3 Format feature type XTF_ENUMS

This feature type is only created by the reader, if the keyword is "CREATEFEATURETYPE4ENUM" is set to "SingleType".

Attribute	*Description*
thisEnum	Qualified INTERLIS name of the enumeration definition of this element.
baseEnum	Qualified INTERLIS name of the base enumeration definition of this element. This is only set, if the enumeration is EXTENDED.
iliCode	Qualified INTERLIS Name of the enumeration element. Same as it would appear in an INTERLIS 2 transfer file (XTF).
itfCode	Code of the enumeration element as it would appear in an INTERLIS 1 transfer file (ITF).
seq	Ordering position of the element. Only set, if this enumeration is ORDERED.

12 Limitations

- custom line forms
- XTF line attributes
- recursive structure attributes

13 Installation

13.1 Requirements

For the current version of ili2fme, you will need a JRE (Java Runtime Environment) installed on your system, version 1.4.1 or later.
The JRE (Java Runtime Environment) can be downloaded for free from the Website http://www.java.com/.
ili2fme was tested with FME 2009 (Build 5676) and FME 2010 (Build 6192).

13.2 Files

To install ili2fme, choose a directory and extract the distribution file there.

Copy the files and subdirectories of "${ili2fme}/FME Suite" to your FME directory.

Add your standard INTERLIS models to the directory "${FME}/plugins/interlis2/ilimodels".

At runtime, ili2fme requires the following files:

```
${FME}/plugins/ili2c.jar
${FME}/plugins/ili2fme.jar
${FME}/plugins/jts-1.8.jar
${FME}/metafile/ch.ehi.fme.Main.fme
${FME}/formatsinfo/interlis2.db
```

13.3 Configuration

To use ili2fme with the FME Universal Viewer, FME versions prior to FME 2008 requires you to set an environment variable: FME_VIEWER_THREADING=SINGLE.

ili2fme doesn't use or require any windows registry entries or user settings file.

13.4 How to migrate/update an existing ili2fme installation

Just copy the files and subdirectories of the new "${ili2fme}/FME Suite" to your FME directory.

Starting with ili2fme version 4.0, there is no longer a native part required. You may delete the files iom_fme.dll and `xerces-c_2_6-interlis2.dll`.

14 FAQ

14.1 Usage

I am getting the following error: "missing model Roads"
In the folder of your data-file or your folder ${FME}/plugins/interlis2/ilimodels there is no .ili-file containing a "MODEL Roads". Move the file Roads.ili to the folder of your data-file or the folder ${FME}/plugins/interlis2/ilimodels.

My destination format is INTERLIS and I'm getting the follwing error: "model name not specified"
You must change the Parameter "MODELS" to "%DATA" or the name of the INTERLIS model (without extension .ili) that you intend to write (on the Destination Dataset).

My destination format is INTERLIS and I'm getting the follwing error: "missing mandatory attribute xtf_class."
The appropriate feature types are expected by the writer, as if the same model would have been read by the INTERLIS 2 reader. That means: Every feature type must have the Attributes xtf_id, xtf_class, xtf_basket. There must be a feature type XTF_BASKET with attributes xtf_id and xtf_topic.

My destination format is INTERLIS and I'm getting the follwing error: "missing mandatory attribute xtf_basket."
The appropriate feature types are expected by the writer, as if the same model would have been read by the INTERLIS 2 reader. That means: Every feature type must have the Attributes xtf_id, xtf_class, xtf_basket. There must be a feature type XTF_BASKETS with attributes xtf_id and xtf_topic.

I have an INTERLIS model "Roads.ili". Should I place into the folder ${FME}/plugins/interlis2/ilimodels?
Yes, if you read or write data according to that model more than once. (ili2fme will also look in the folder of your data-file for INTERLIS models.)

Is the ordering of the model names as a value of the FME-keyword "Ili2fme_Models" significant?
No, any ordering will do.

If a model imports other models (like "Units" or "CoordSys"), should I name all models as value of the FME-keyword "Ili2fme_Models"?

No, but all required models (including indirectly imported ones), all required .ili-files, should be in the folder of your data-file or the folder $(FME)/plugins/interlis2/ilimodels.

If a model imports other models (like "Units" or "CoordSys"), which one should I name as value of the FME-keyword "Ili2fme_Models"?
Use the most specific one (the one that imports directly or indirectly all the other ones). The imported models will be used automatically.

If a model extends another one, which one should I name as value of the FME-keyword "Ili2fme_Models", the base model or the extended one?
Use the extended one. The base model will be used automatically.

I would like to convert to a particular INTERLIS model. How can I import the feature types?
Import the INTERLIS model file (file with the extension .ili), instead of a INTERLIS data file. (You have to change the Filetype in the file selector dialog to "All Files" to see the ili-files.)

14.2 Mapping

How to map XTF_ID, XTF_CLASS, XTF_BASKET if INTERLIS is the destination format?
XTF_ID is the XML attribute TID and should be unique across all feature types. Typically the value of the primary key of the source feature.
XTF_CLASS the qualified name of the destination INTERLIS-class. Typically a constant like "ModelName.TopicName.ClassName" (the actual value depends on your INTERLIS model).
XTF_BASKET is the foreign key of a feature of type XTF_BASKETS.

How to specify at export the kind of transfer (FULL, INITIAL, UPDATE) and the kind of feature operation (INSERT, UPDATE, DELETE)?
The kind of transfer is indicated by values in the attributes "xtf_startstate" and "xtf_endstate" of the format feature type "XTF_BASKETS". If "xtf_endstate" is not set, its an FULL transfer. If "xtf_endstate" is set and "xtf_startstate" is not set, it's an INITIAL transfer. If "xtf_endstate" and "xtf_startstate" are set, it's an INITIAL transfer.
The values INSERT, UPDATE, DELETE are required for incremental transfer. Use the format attribute "xtf_operation".

What is the purpose of the feature type "XTF_DELETEOBJECT"?
It's a shortcut to signal "this object is no longer in the basket".

What is the purpose of the format attribute "xtf_operation"? Which are the possible values?
This format attribute indicates the kind of change to the object. Possible values are: INSERT, UPDATE, DELETE. It's only used with incremental transfer mode.

What is the purpose of the format attribute "xtf_consistency"? Which are the possible values?
Possible values are: COMPLETE, INCOMPLETE, INCONSISTENT, ADAPTED.

If an attribute is of type enumeration (like „color: (red,green,blue);": Is it possible to get the values (0,1,2,...) instead of the resolved names?
XTF file: No. In INTERLIS 2 the resolved name is the value. In INTERLIS 2 there is no mapping of an enumeration to a numeric.
ITF file: Yes. Set the parameter ILI1_ENUMASITFCODE to "Yes". (But this is not recommended, because it is more difficult to detect errors in the mapping script, and the mapping script becomes incompatible to INTERLIS 2 (XTF).)

How are foreign keys mapped?
The value of the REF XML-attribute of the role (association end) gets the property value of the feature, that contains the role.

How are 1-1 associations mapped?
Like defined by the INTERLIS 2-encoding rules. The end class of the second role (association end) gets the property with the reference/foreign key. The property gets the name of the first role.

How are BAG/LIST-attributes mapped?
BAG/LIST-attributes are mapped as list attribute.

How is inheritance mapped?
ili2fme uses a super or subclass strategy.

My INTERLIS model contains a lot of classes, but in FME, I see only a few of them as feature types. Why?
ili2fme uses by default a super type strategy to map the inheritance tree of the INTERLIS classes. Only root classes in INTERLIS become feature types in FME. You may consider changing the mapping strategy to subclass.

How can I read the TID of records out of INTERLIS 1 transfer files (files with extension .itf)?
The TID is accessible threw the XTF_ID attribute in each feature type.

14.3 Configuartion

Which version of ili2fme is installed?
Run FME Viewer and open an INTERLIS data file. The version of ili2fme will be written to the log window of FME (e.g. "ili2fme-5.1.0-20090311").

Why does FME report: “No Reader named `ch.ehi.fme.Main' is available in this FME version”?
This may have several reasons:

- No JAVA installed
- Wrong Version of JAVA installed (ili2fme requires at least JAVA 1.4.1)
- Wrong FME edition (normally ili2fme requires at least FME Professional)
- Maybe jvm.dll is not found by FME.

FME uses standard registry entries to find JAVA. Check your JAVA installation (Open a command prompt and enter “`java –version`”).

15 Changes
See the file CHANGELOG.txt in the distribution of ili2fme.

16 License

Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library. In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices. Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy. This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange. If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation,

is not a derivative work of the Library, and therefore falls outside the scope of this License. However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables. When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law. If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.) Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications. You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy. For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself

accompanies the executable.It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library. If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice. This section is intended

to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS