Swiss INTERLIS (ili2fme) Reader/Writer

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

This documentation 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 https://www.interlis.ch/.

An alternative guide can be found here.

This documentation describes version 7.2.x of ili2fme. The current version of ili2fme is available at http://www.ili2fme.ch/.

NOTE

- Please note that Safe Software distributes the ili2fme format with FME as a convenience.
- The Licensing options for this format begin with FME Desktop Professional Edition.

Overview

Features read from an INTERLIS file consist of a series of attribute values. The attribute values 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 can read and write INTERLIS 1 and 2. However, in most cases you will need an FME script or FME Workbench to write INTERLIS.

Additional documentation is available at http://www.ili2fme.ch/.

The ili2fme is owned by Eisenhut Informatik AG and is licensed under the LGPL (Lesser GNU Public License). Safe does not provide warranties of any kind, express or implied, as to noninfringement of third party rights, merchantability, or fitness for any particular purpose with respect to the ili2fme. In no event shall Safe be liable for indirect, special, incidental, or consequential damages or loss of profit or business opportunity related to the use of the ili2fme. Please visit our website at http://www.safe.com/foss for further details. Please send comments about ili2fme to mailto:info@eisenhutinformatik.ch.

Quick Facts

About Quick Facts Tables

Format Type Identifier	ch.ehi.fme.Main
Long Format Name	Swiss INTERLIS (ili2fme)
Reader/Writer	Both
Licensing Level	Professional and above
Dependencies	None
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	xtf_geomtype

Geometry Support

Geometry	**Supported?**
aggregate	no
circles	stroked
circular arc	yes
donut polygon	yes
elliptical arc	stroked
ellipses	stroked
line	yes
none	yes
point	yes
polygon	yes
raster	no
solid	yes

Geometry	Supported?
surface	yes
text	no
z-values	yes

Use Cases

Reading INTERLIS 1-Data

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

- in a model repository on the internet
- 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.

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" â€“ 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.

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 the appropriate ili2fme parameters)
- 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!

NOTE

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).

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

NOTE

- you should define the appropriate ili2fme parameters when importing the feature types (such as "superclass" or "subclass" inheritance mapping strategy)
- `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).

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".

Reading and writing INTERLIS-Data

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)!

INTERLIS-Models

Normally ili2fme will read the required INTERLIS-Models as required by your data. Only when you "Import features types" (a FME Workbench menu item) you should specify a model file (a file with the extension ".ili"). You can specify the places that ili2fme should look after the required models by setting the parameter MODEL_DIR. If a file folder doesn't contain a file named "ilisite.xml" or "ilimodels.xml", ili2fme will scan all files with an extension ".ili". If the folder contains multiple files with extension ".ili" that contain an INTERLIS models with the same name, you will get unexpected results. If a file folder contains a file named "ilisite.xml" or "ilimodels.xml", ili2fme will use the folder as an INTERLIS model repository. "ilimodels.xml" lists models and associates them with files. "ilisite.xml" contains links to other model repositories.

Reader Parameters

Models

The required INTERLIS models to read the dataset (the model name, not the file of the model; so no extension .ili) and separated by semicolons (;). The default value %DATA is a placeholder and means that models are determined by inspecting the transfer file.

Models Directory

This is the folder that contains the .ili files. These files are scanned for INTERLIS models. You may use %XTF_DIR as a placeholder for the folder of the data file that you will read. Also model repositories might be specified (such as http://models.interlis.ch/). Multiple folders or repositories may be separated by semicolons (;).

Topics Filter

These are the qualified names of INTERLIS topics to read data from (for example, DM01.Bodenbedeckung). You can enter multiple topic names, separated by semicolons (;). If set, the data of other topics will be ignored.

This parameter can remain empty. If it is not set, data of all topics will be read.

NOTE	The Topics Filter parameter does not affect the number of FME Feature Types that are created during initial reader setup in FME Workbench. This aspect is based on the option *Inheritance Mapping Strategy* (see below), the Workflow Option in the *Add Reader* dialog and the selected Feature Types in the *Select Feature Types* dialog.

Check TID/OID Uniqueness

Yes

The reader will check if the TIDs/OIDs are unique.

No

It will bypass this check.

Validate

Yes

The reader will validate the data by using the ilivalidator.

No

It will bypass the validation completely.

Validate Attribute/Role Multiplicity

Yes

The reader will check for mandatory but missing values/references.

No

It will bypass this validation.

Validator Configuration

An ilivalidator configuration file to fine tune the validation. The value could also be in the form `ilidata:DatesetId` then the file will be downloaded from the repositories. See https://github.com/claeis/ilivalidator/blob/master/docs/ilivalidator.rst#konfiguration for further information.

Meta Configuration

A meta configuration file to setup ili2fme. The value could also be in the form `ilidata:DatesetId` then the file will be downloaded from the repositories.

ITF Ignore Polygon Building Errors

This setting is independent of an enabled or disabled validation and applies only to INTERLIS 1 datasets.

Yes

The reader will ignore AREA/SURFACE errors, that would result in incomplete data.

No

The reader will report AREA/SURFACE errors, that would result in incomplete data.

Geometry Encoding

Defines the encoding of geometry attributes, which are not used as FME geometry (only the first geometry attribute becomes an FME geometry).

FMEXML

encodes as FME XML

FMEBIN

encodes as FME Binary

FMEHEXBIN

encodes as FME Hex Binary

OGCHEXBIN

encodes as OGC Hex Binary

Mapping of multiple Geometry Attributes

Defines the encoding of INTERLIS geometry attributes, in cases where the INTERLIS class defines multiple attributes of type geometry.

EncodeAsFmeAttribute

Only the first geometry attribute becomes an FME geometry. Any additional INTERLIS geometry attributes are mapped to FME attributes.

RepeatFeature

The reader creates multiple FME features for one single INTERLIS object; one feature per geometry attribute value of the single INTERLIS object (any non-geometry attribute is the same in all this cloned features).

ITF Linetable Mapping

Applies only to INTERLIS 1 datasets.

Polygon

The reader will create polygons for all SURFACE/AREA attributes; no linetable features are created. This option requires valid data.

Raw

The reader will read the data as it is in the ITF transfer file. No polygon building for SURFACE/AREA attributes will be done. This option enables to read invalid SURFACE/AREA data, and can be used for error analysis.

Polygon+Raw

The reader will create polygons for all SURFACE/AREA attributes, but will also create linetable features. AREA linetables will contain one or two references to the features with the polygons. This option requires valid data.

Inheritance Mapping Strategy

Applies only to INTERLIS 2 datasets.

NOTE	For more information, see the section titled Inheritance mapping strategies under 'Feature Representation' in the Swiss INTERLIS (ili2fme) Reader/Writer section of the Readers and Writers Manual.

SuperClass

The superclass inheritance mapping strategy is applied.

SubClass

The subclass inheritance mapping strategy is applied.

Trim Values

Yes

The reader will remove leading and trailing spaces from text attributes.

No

The reader will bypass this data cleaning.

ITF Add Default Values

Applies only to INTERLIS 1 datasets.

Yes

The reader will parse the explanation at the end of attribute definitions that are optional. If there is no attribute value in the data, it will add the one given in the model.

No

The reader will not supply any default values to the data.

ITF Renumber TIDs

Applies only to INTERLIS 1 datasets.

Yes

The reader will renumber the objects so that the TID becomes unique across the whole transfer. Any references to the renumbered objects are changed appropriately.

No

The reader will read the TIDs without making any changes.

ITF Read enum Values as Code

Applies only to INTERLIS 1 datasets.

Yes

The reader will read values of attributes of type enumeration as numeric code (the same code as it appears in the ITF transfer file). This option is not recommended and exists only for backward compatibility reasons.

No

The reader 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.

Create Feature Types For Enumerations

Controls how FME feature types are created for INTERLIS enumerations

No

No feature types are created for enumerations

SingleType

A single additional feature type called "XTF_ENUMS" is created and each element of all enumeration types is provided as a feature of this feature type.

OneTypePerEnumDef

One feature type is created for each enumeration type.

http Proxy Host

This is the proxy server that ili2fme will use to access model repositories.

http Proxy Port

This is the proxy server that ili2fme will use to access model repositories.

Enable Trace Messages

Controls the level of detail of log messages written by the reader.

Yes

details progress messages will be written to the log

No

only normal progress messages will be written to the log

Writer Parameters

Models

The required INTERLIS models to write the dataset (the model name, not the file of the model; so no extension .ili) and separated by semicolons (;). The default value %DATA is a placeholder and means that models are determined by inspecting the features.

Models Directory

This is the folder that contains the .ili files. These files are scanned for INTERLIS models. You may use %XTF_DIR as placeholder for the folder of the data file that you will write. Also model repositories might be specified (such as http://models.interlis.ch/). Multiple folders or repositories may be separated by semicolons (;).

Check TID/OID Uniqueness

Yes

Checks if the TIDs/OIDs are unique.

No

This check is bypassed.

Validate

Yes

The writer will validate the data by using the ilivalidator.

No

It will bypass the validation completely.

Validate Attribute/Role Multiplicity

Yes

The writer will check for mandatory but missing values/references.

No

It will bypass this validation.

Validator Configuration

An ilivalidator configuration file to fine tune the validation. The value could also be in the form `ilidata:DatesetId` then the file will be downloaded from the repositories. See https://github.com/claeis/ilivalidator/blob/master/docs/ilivalidator.rst#konfiguration for further information.

Meta Configuration

A meta configuration file to setup ili2fme. The value could also be in the form `ilidata:DatesetId` then the file will be downloaded from the repositories.

Inheritance Mapping Strategy

Applies only to INTERLIS 2 datasets.

SuperClass

The superclass inheritance mapping strategy is applied.

SubClass

The subclass inheritance mapping strategy is applied.

Geometry Encoding

Defines the encoding of geometry attributes which are not used as FME geometry (only the first geometry attribute becomes FME geometry).

FMEXML

encodes as FME XML

FMEBIN

encodes as FME Binary

FMEHEXBIN

encodes as FME Hex Binary

OGCHEXBIN

encodes as OGC Hex Binary

Trim Values

Yes

The writer will remove leading and trailing spaces from text attributes.

No

It will bypass this data cleaning.

Use Linetables

This field applies only to INTERLIS 1 datasets with INTERLIS AREA or INTERLIS SURFACE attributes.

Yes

The writer will expect one additional feature type for each INTERLIS SURFACE or AREA attribute. The additional feature type with the suffix _$(attributeName) contains the line helper features as they should appear in the transfer-file.

No

The writer will create the line helper table out of the polygons/donuts.

http Proxy Host

This is the proxy server that ili2fme will use to access model repositories.

http Proxy Port

This is the proxy server that ili2fme will use to access model repositories.

Enable Trace Messages

Controls the level of detail of log messages written out.

Yes

Detailed progress messages will be written to the log.

No

Only normal progress messages will be written to the log.

Meta Configuration

The following parameters are supported in the meta configuration file.

Configuration	Example	Description
baseConfig	[CONFIGURATION] baseConfig=ilidata:DatesetId	Base meta-configuration on which the current meta-configuration is built. Instead of `ilidata:DatesetId`, the form `file:/localfile` can also be used, in which case the corresponding local file is used. Several base configurations are separated with a semicolon ";".
org.interlis2.validator.config	[CONFIGURATION] org.interlis2.validator.config=ilidata:DatesetId	Validation configuration to be used. Instead of `ilidata:DatesetId`, the form `file:/localfile` can also be used, in which case the corresponding local file will be used. Multiple validation configurations are separated with a semicolon ";".

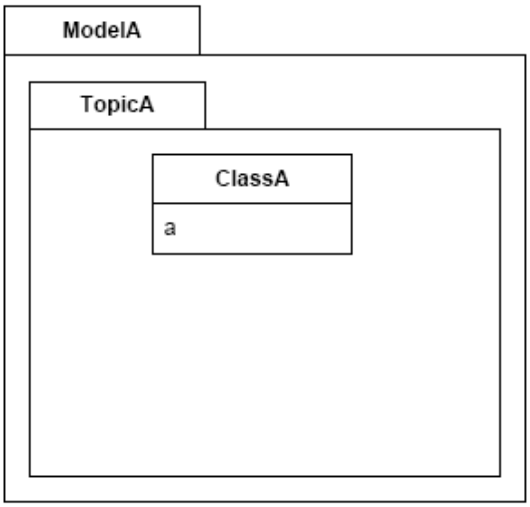
Example of a meta configuration file:

```
[CONFIGURATION]
org.interlis2.validator.config=ilidata:70340b5a-248c-4216-86e4-32e6a540d629
```

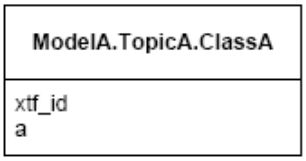
Feature Representation

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.

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 does not allow such a nesting; therefore, ili2fme maps INTERLIS class with their qualified name to FME feature types.

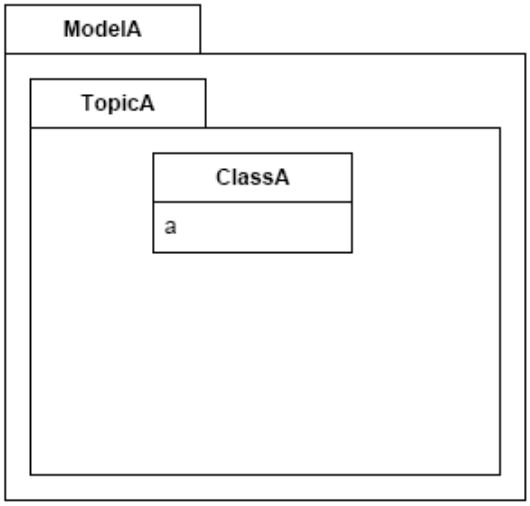


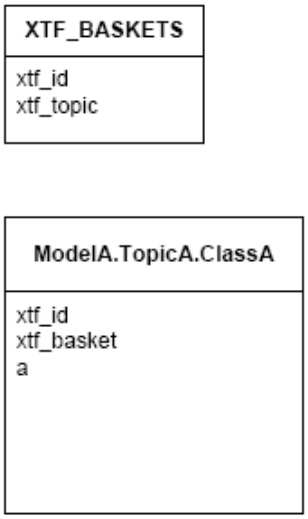
INTERLIS model



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.

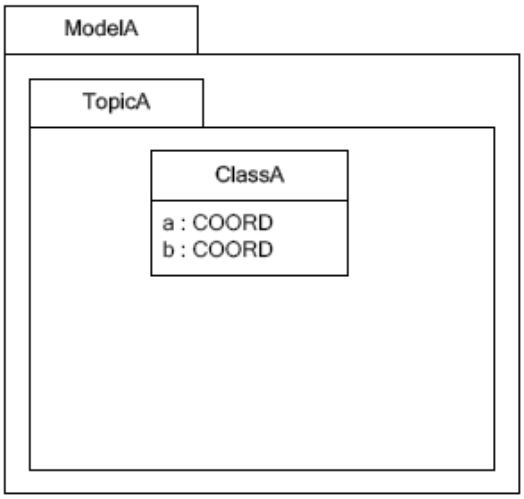




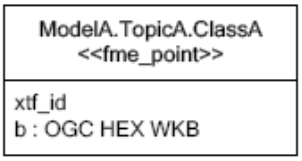
To know which feature belongs to which basket, each feature has a reference to the basket in the format attribute xtf_basket. Each basket is represented as an instance of the format feature type XTF_BASKETS. The attribute xtf_topic holds the qualified topic name that describes this basket (in this case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

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 existing FME attributes. The value of these attributes (attribute *b* in the diagram above) 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 GeometryReplacer transformer or set with the GeometryExtractor transformer.

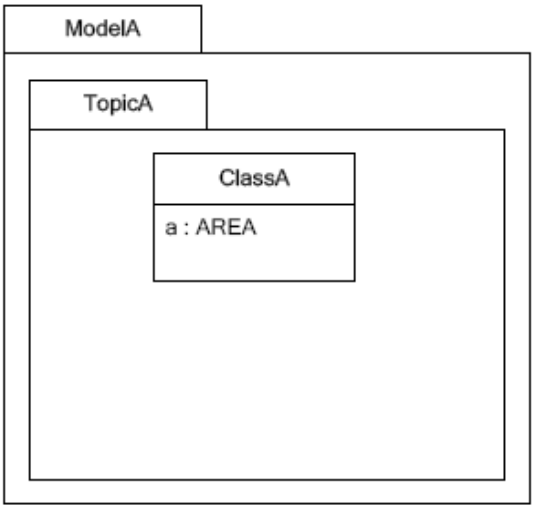
INTERLIS 1 Area

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

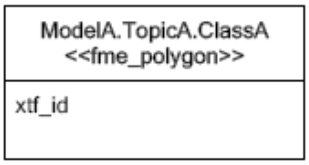
- build polygons/donuts automatically from the line table
- read the main table and the line table as they are in the transfer file
- combination of the two cases above

Automatic polygon building works only, if the AREA attribute is the first geometry attribute of the INTERLIS table.

With automatic polygon building, the mapping is as follows:

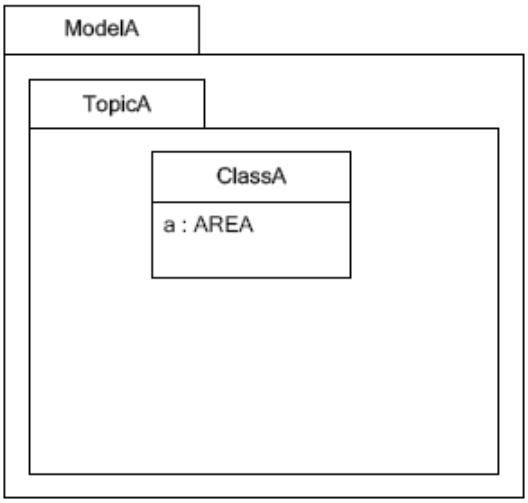


INTERLIS model

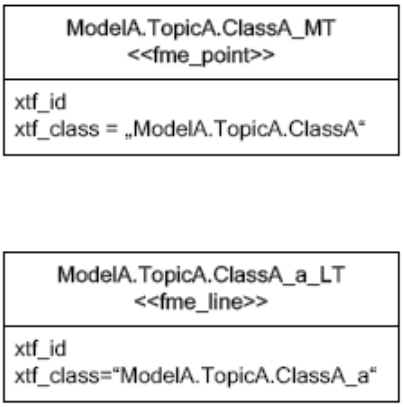


FME feature schema

With automatic polygon build disabled, the mapping is as follows:



INTERLIS model



FME feature schema

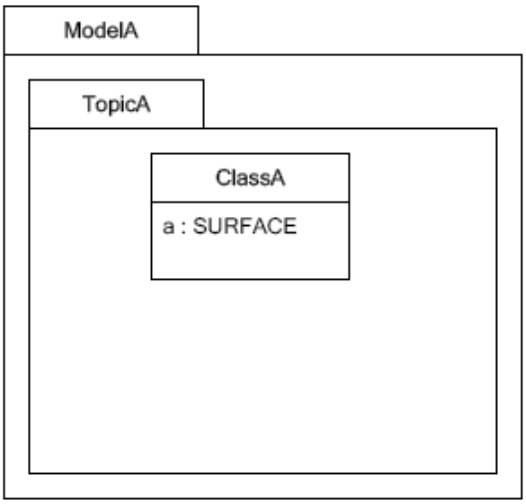
INTERLIS 1 Surface

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

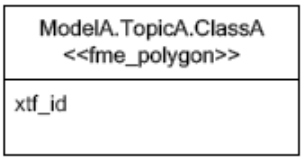
- build polygons/donuts automatically from the line table
- read the main table and the line table as they are in the transfer file
- combination of the two cases above

Automatic polygon building works only, if the SURFACE attribute is the first geometry attribute of the INTERLIS table.

With automatic polygon building the mapping is as follows:

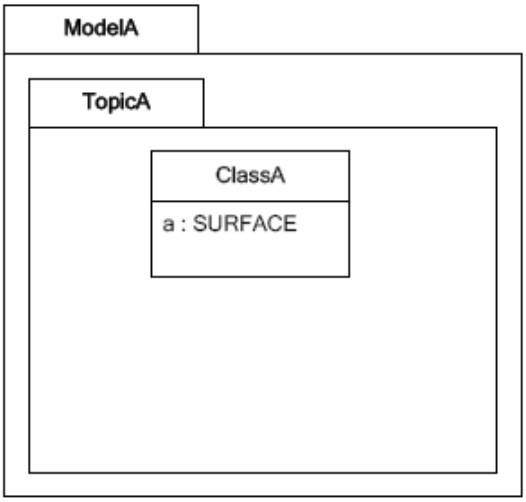


INTERLIS model

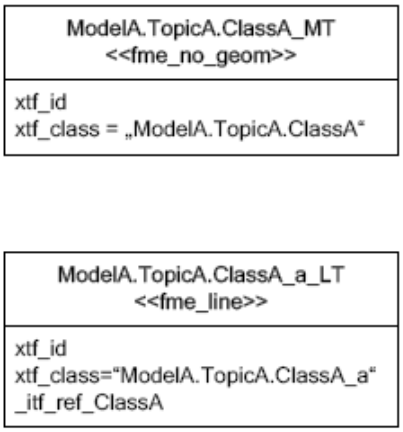


FME feature schema

With automatic polygon build disabled, the mapping is as follows:



INTERLIS model

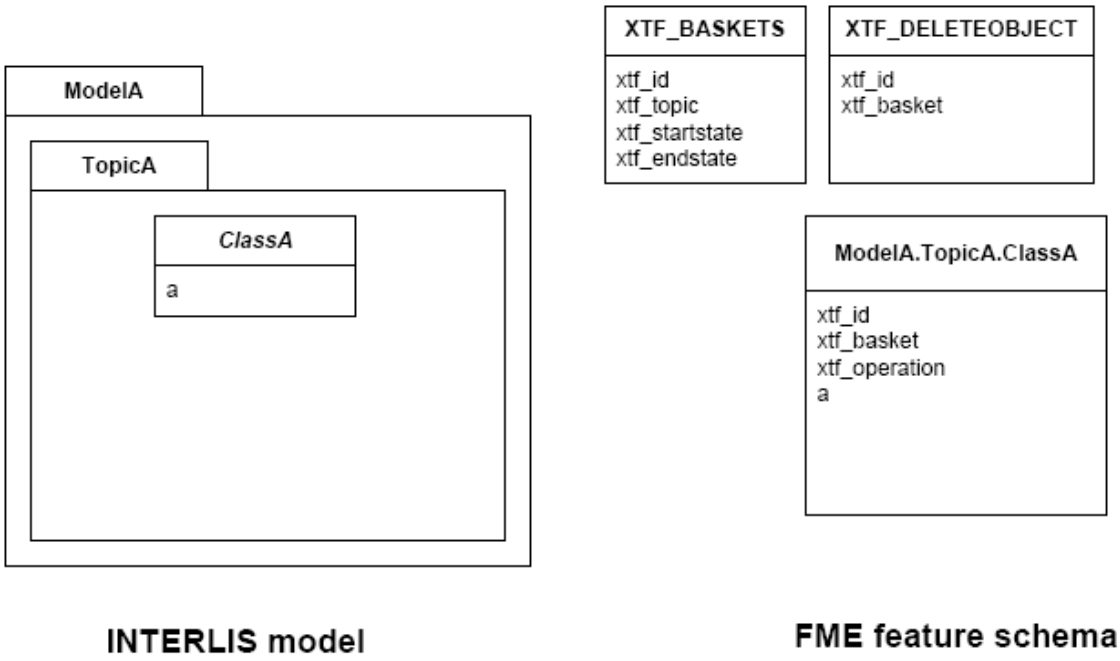


FME feature schema

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`)

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 series of transfers. It includes all objects. UPDATE is used for all succeeding transfers following 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.

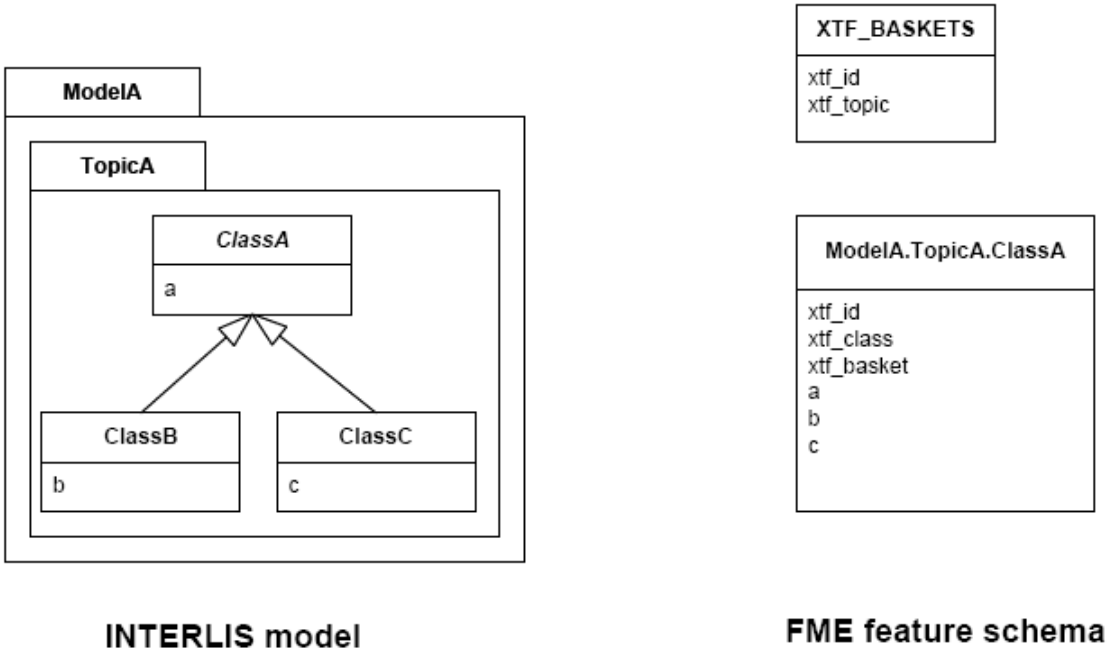
For an UPDATE data transfer, the XTF_BASKETS feature that represents the basket has a value in the `xtf_startstate` and the `xtf_endstate` attribute. The `xtf_startstate` value is the same as the `xtf_endstate` of the last transfer of that basket. The `xtf_operation` attribute should be set to `INSERT`, `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`).

Inheritance Mapping Strategy

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

Superclass Strategy

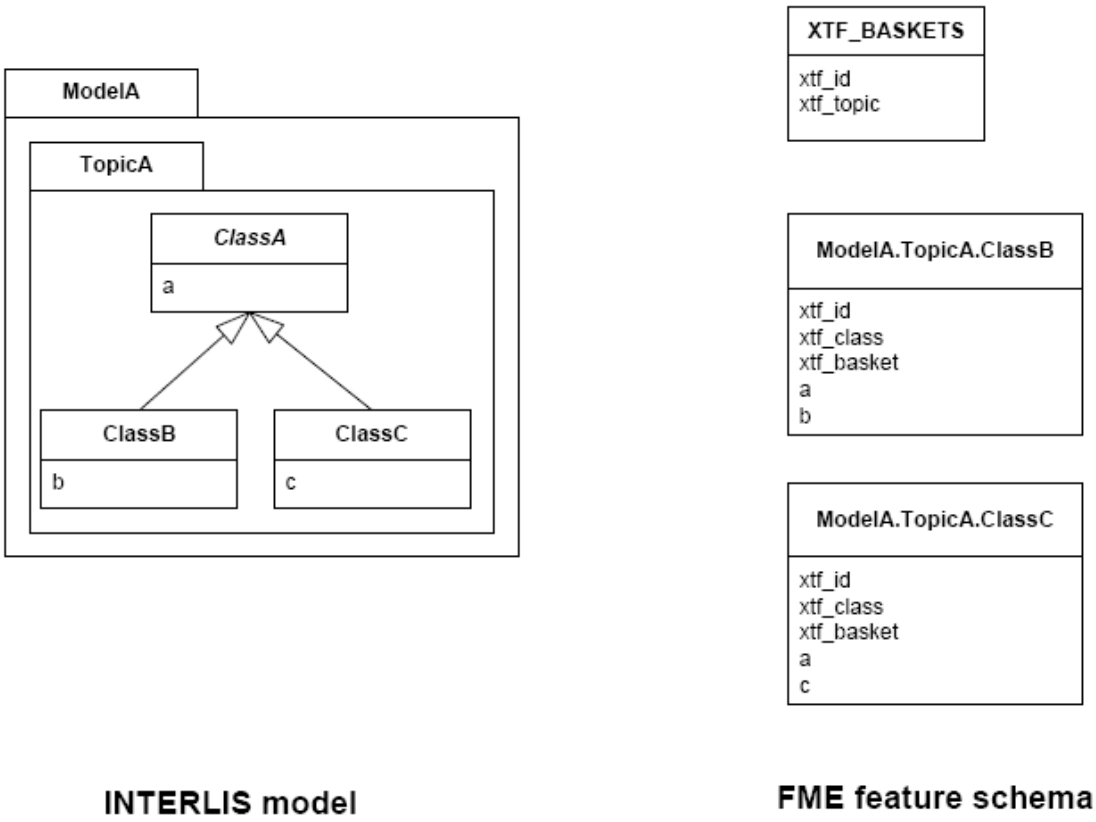
Attributes of non-root classes are shifted to the root, as illustrated by the following figure:



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`.

Subclass Strategy

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



There is no feature type `ModelA.TopicA.ClassA` because it's an abstract class in the INTERLIS model.

Enumerations

There are two modes to read enumerations:

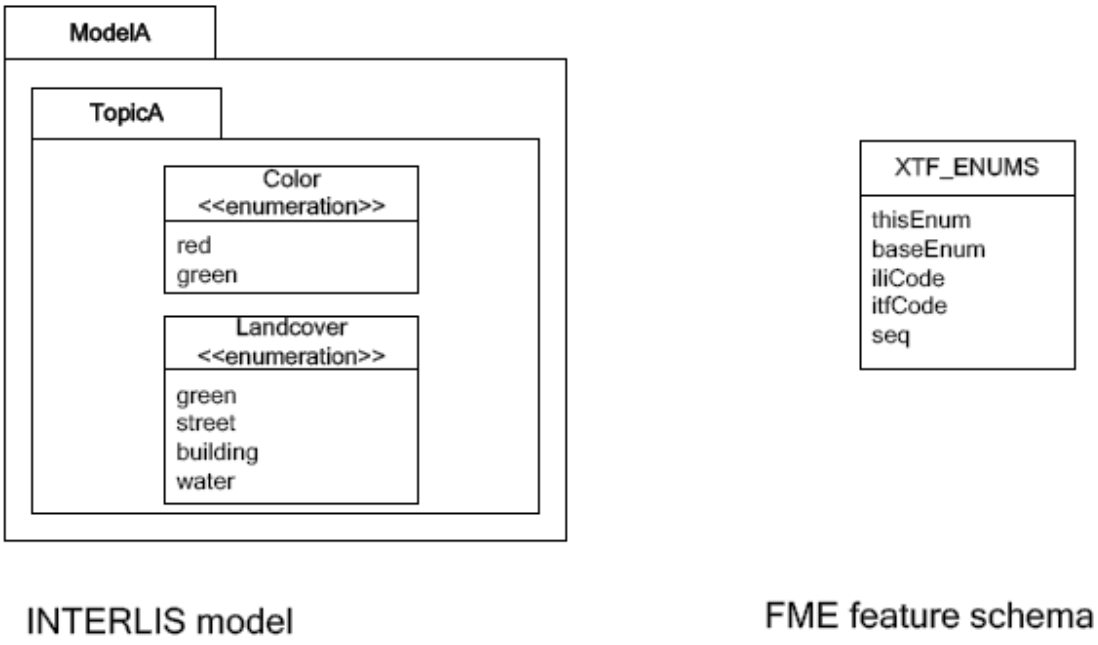
SingleType

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.

Enumerations as a Single Feature Type

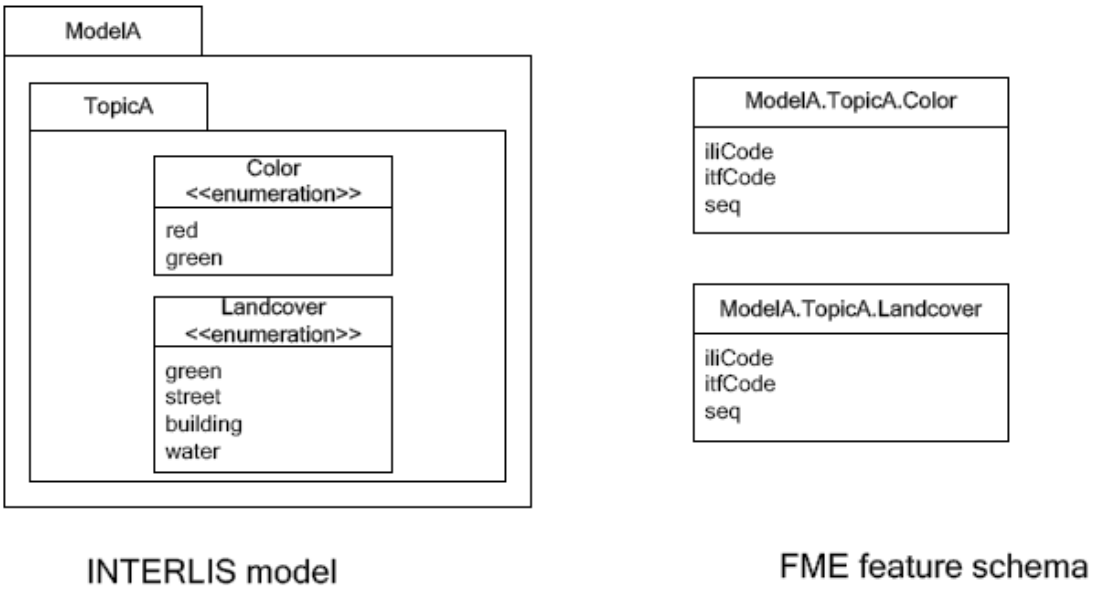


For the feature type `XTF_ENUMS`, the following features will be read:

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.

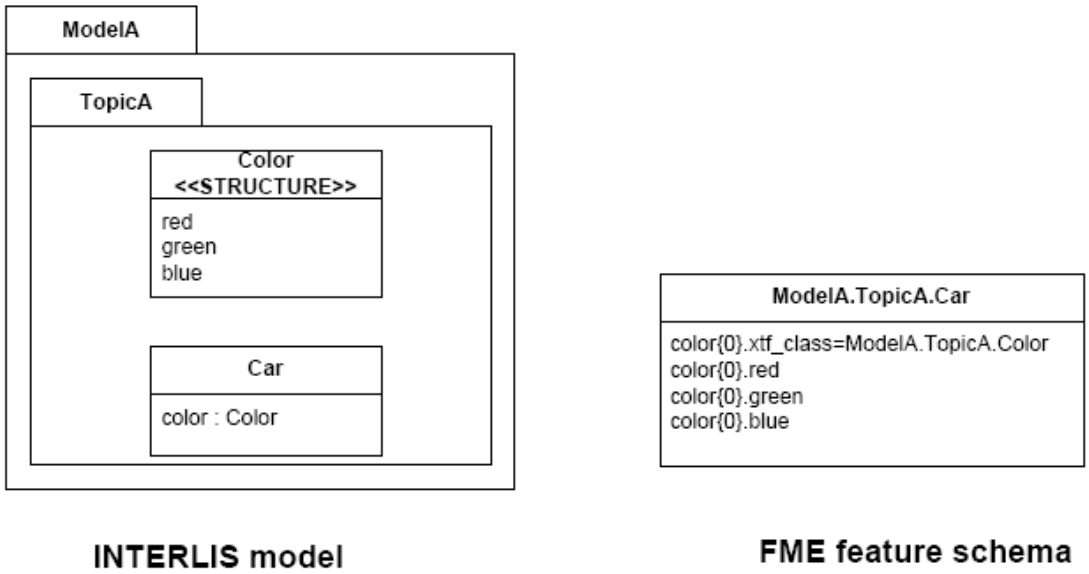
One Feature Type per Enumeration



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

iliCode	itfCode	seq
red	0	
green	1	

BAG/LIST OF



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. The type of the structure element is defined by the value of the attribute xtf_class (similar to the class type of objects; see sec. Superclass Strategy), which is mandatory to be set. In the example has the list attribute `color{0}.xtf_class` therefore the value `ModelA.TopicA.Color`.

Format Attributes

In addition to the generic FME feature attributes that FME Workbench adds to all features (see About Feature Attributes), this format also adds format-specific attributes.

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.

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.

XTF_TRANSFER

Content of the INTERLIS 2 transfer file header section.

Attribute	Description
oidspace{}	Content from the `<OIDSPACES>` element from the header section of the transfer file.

Attribute	Description
oidspace{}.name	For each OID domain used in this INTERLIS 2 transfer file, an alias name (as used in this transfer file).
oidspace{}.oiddomain	Qualified name of the INTERLIS 2 OID domain definition.
comment	Content of `<COMMENT>` element from the header section of the transfer file.

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_seq	index (starting with 1) of basket in transfer file (only INTERLIS 2).
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.
xtf_domains{}	List of assignments from generic to concrete coord domain names.
xtf_domains{}.generic	Qualified name of an INTERLIS 2 generic coord domain. In the figure above would `MultiCrs24.Coord` be a possible value.
xtf_domains{}.concrete	Qualified name of an INTERLIS 2 concrete coord domain. In the figure above would `MultiCrs24.Coord_LV03` be a possible value.

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.

XTF_ENUMS

This feature type is only created by the reader if the parameter Create Feature Types For Enumerations 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`.

XTF_ERRORS

Errors from the reader.

Attribute	Description
iliname	Qualified name of the INTERLIS 2 model element that is related to the message
message	Error message
tid{}	`TID`s of the objects related to the message

Limitations

- custom line forms

- XTF line attributes
- recursive structure attributes

License

- ili2fme is licensed under the LGPL (Lesser GNU Public License).
- Some libraries used by ili2fme are licensed under MIT/X.
- Some libraries used by ili2fme are licensed under Apache 2.0.
- Some libraries used by ili2fme are licensed under a library specific license.
- ili2fme includes software developed by The Apache Software Foundation (http://www.apache.org/).

Installation

NOTE

- Please note that Safe Software distributes the ili2fme format with FME as a convenience.
- The Licensing options for this format begin with FME Desktop Professional Edition.

Requirements

For the current version of ili2fme, you will need a JRE (Java Runtime Environment) installed on your system, version 1.6.0 or later. The JRE (Java Runtime Environment) can be downloaded for free from the Website http://www.java.com/.

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-core-1.14.0.jar
${FME}/metafile/ch.ehi.fme.Main.fme
${FME}/formatsinfo/interlis2.db
```

Configuration

To use ili2fme with the FME Universal Viewer, FME 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.

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` (from previous ili2fme versions). You must delete the file `jts-1.8.jar` and `jts-1.13.jar`. They are in conflict with `jts-core-1.14.0.jar` and result in a error

```
tried to access field com.vividsolutions.jts.geom.LineString.points from class
ch.interlis.iom_j.itf.impl.jtsext.geom.CompoundCurveRing
```

FAQ

Usage

1. *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.

2. *My destination format is INTERLIS and I'm getting the following 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).

3. *My destination format is INTERLIS and I'm getting the following 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.

4. *My destination format is INTERLIS and I'm getting the following 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.

5. *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.)

6. *Is the ordering of the model names as a value of the FME-keyword "Ili2fme_Models" significant?*

   No, any ordering will do.

7. *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.

8. *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.

9. *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.

10. *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.)

11. *Is it possible to merge the output of ili2fme with an existing file?*

    No, not directly. But you can read the existing file into the same workbench, setup the merging inside the workbench and write the result of the merging. In that way, you are able to fully control how the existing and the new data is merged.

Mapping

1. *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.

2. *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”.

3. *What is the purpose of the feature type "XTF_DELETEOBJECT"?*

   It’s a shortcut to signal “this object is no longer in the basket”.

4. *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.

5. *What is the purpose of the format attribute “xtf_consistency”? Which are the possible values?*

   Possible values are: COMPLETE, INCOMPLETE, INCONSISTENT, ADAPTED.

6. *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).)

7. *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.

8. *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.

9. *How are BAG/LIST-attributes mapped?*

   BAG/LIST-attributes are mapped as list attribute.

10. *How is inheritance mapped?*

    ili2fme uses a super or subclass strategy.

11. *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.

12. *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.

13. *My INTERLIS model contains a class with more than one geometry attribute. How is this class mapped to an FME feature type?*

    ili2fme uses the first geometry attribute of the INTERLIS class as geometry of the FME feature type. Any further geometry attributes are mapped as ordinary FME attributes. The encoding of the FME attributes containing geometry can be controlled by the ili2fme parameter GEOMETRY_ENCODING.

Configuration

1. *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").

2. *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 1.6.0)

- Wrong FME edition (normally ili2fme requires at least FME Professional)
- Maybe jvm.dll is not found by FME.
- Maybe a required JAR file is missing in $(FME)/plugins. The following JAR files are required: ili2fme.jar, ili2c.jar, jts-core-1.14.0.jar, pluginbuilder.jar FME uses standard registry entries to find JAVA. Check your JAVA installation (Open a command prompt and enter “java –version”).