cs2bim

FHNW Institut Digitales Bauen

26.03.2026

Table of contents

Introduction

The service **cs2bim** transforms GIS-based cadastral survey (CS) data to IFC instances. The service offers sophisticated options for data model transformation between GIS and IFC, and it provides various methods for geometric conversions, particularly from 2D to 3D.

The service contains the following major components:

- Reading 2D GIS feature types and systematically converting them to IFC entities, supporting key concept templates of IFC.
- Processing of the terrain model to create the resulting 3D surfaces with geometric projection methods.
- Processing of CityGML data (buildings) to convert it to IFC entities.
- Processing of 2.5D utility network data to convert it to IFC entities with geometric extrusion methods.
- Exporting of the objects to IFC format using the IfcOpenShell component.
- API access

To run the service, a ready-to-use Docker-based setup is provided.

These pages describe the technical aspects and configurations of the application, as well as the concepts required and used for it.

The source code is available on github

Project

This application was originally developed as part of the **cs2bim** project. The project has been launched by the Conference of Cantonal Geoinformation and Cadastral Offices (KGK) as *Cadastral Surveying Data to Building Information Modeling (CS2BIM)*. The *Institute of Virtual Design and Construction* and the *Institute of Geomatics* at the University of Applied Sciences Northwestern Switzerland (FHNW) have developed the service based on open source libraries.

Authors

Institut Digitales Bauen
Fachhochschule Nordwestschweiz /
University of Applied Sciences and Arts Northwestern Switzerland, Institute of Virtual Design and Construction

Project team:

- Lukas Schildknecht
- Oliver Schneider
- Joel Gschwind
- Jonas Meyer
- Christian Gamma

If you use this project for your research, please cite:

```
@inproceedings{schildknecht2025cs2bim,
  author={Schildknecht, Lukas and Schneider, Oliver and Meyer, Jonas, and Gamma, Christian
  title={Integration of land administration data into BIM/IFC - an open source approach fo
  year={2025},
  booktitle={Dreiländertagung der DGPF, der OVG und der SGPF in Muttenz, Schweiz},
  series={Publikationen der DGPF},
  volume={Band 33},
  editor={Kersten, Thomas P. and Tilly, Nora},
  publisher={Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation (
  address={Stuttgart, Germany},
  pages={294--310}
}
```

License

Part I

Concepts

1 Concepts

On this page some basics and concepts of the cs2bim project are documented.

- IFC basic principles
- Transformation GIS –> IFC
- Projection and triangulation
- CityGML / 3D city model
- Extrusions

1.1 IFC basic principles

The "Industry Foundation Classes" (IFC) is an open, international standard that defines a conceptual data model for buildings. It is developped and maintained by buildingSmart International and documented in an open HTML based documentation (buildingSmart International, 2023). IFC is also published as an ISO standard (ISO 16739-1, 2024), that is identical to the open standard.

IFC defines a large data model. In the context of cs2bim, the core of the IFC data model can be described (simplified) with the following structures (see also (Schildknecht, 2023)).

- **IfcElement**
  IfcElement is an abstact entity that can be specialised with a lot of different, conrete "business" entities, e.g. IfcDoor, IfcWall etc. The individual semantics of all enitities is definied in (buildingSmart International, 2023). In the context of cs2bim, IfcGeographicElement is currently the main candidate to be used.

- **IfcPropertySet**
  IfcPropertySet (with Properties) is a generic strucutre within IFC that allows the assignment of arbitrary properties to an IfcElement.

- **IfcShapeRepresentation**
  An IfcElement can have zero, one or multiple geometric representations. The "RepresentationIdentifier" defines the type of representation. Possible identifiers could be "Body" (for a 3D representation) or "Axis", amongst others.
  The geometry type is defined by the "RepresentationType" and could be Point, Curve, Surface, SweptSolid and others.

- **IfcSpatialStructureElement**
  An IfcElement can be assigned to a spatial structure element. IfSpatialStructureElements are used to define a spatial-logical structure (typically building-storey-space in buildings; typically segments and cross-sections in infrastructure constructions). In the context of cs2bim, the IfcSpatialStructure is also "misused" for a functional grouping of the elements (without spatial-logical structure).
- **IfcGroup**
  In IFC, any groups can be defined for any subject/functional grouping of elements. The groups can also be structured hierarchically. An IfcElement can be assigned to any group.
- **IfcClassificationReference**
  As an alternative to IfcGroup, an IfcElement can be assigned to a classification value. Classification values belong to classifications that are typically defined outside of IFC.

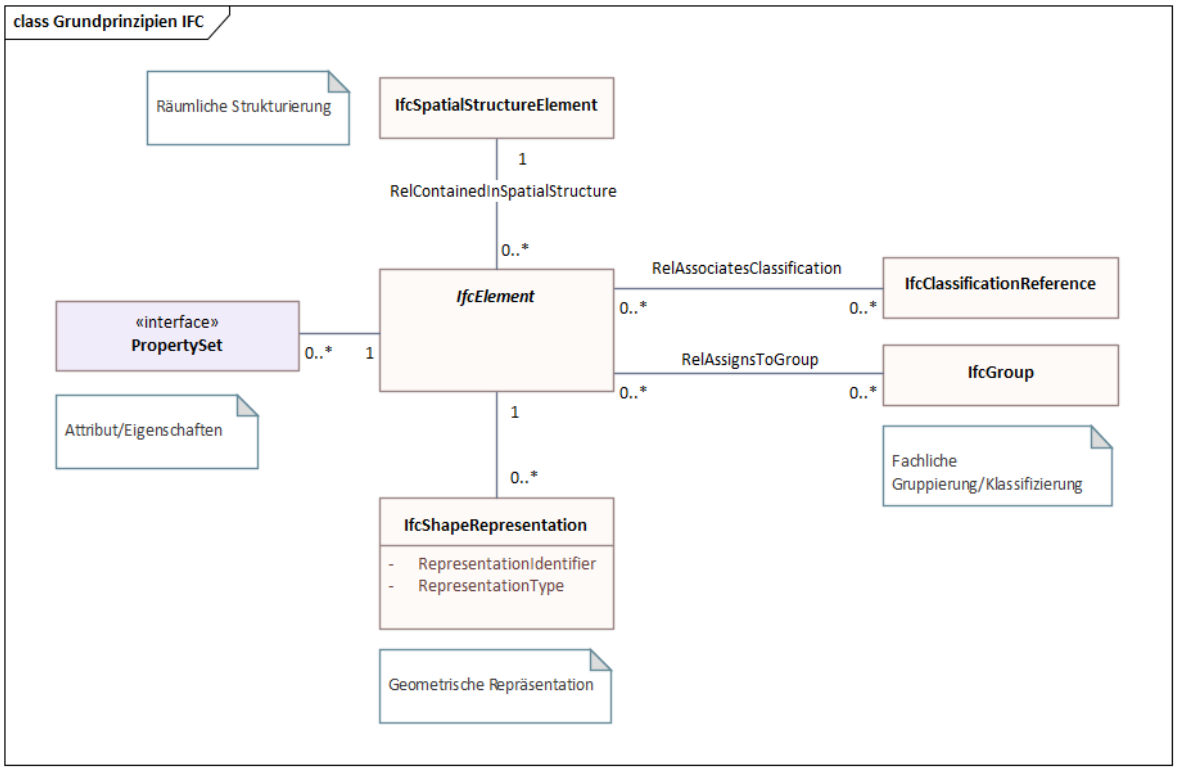


Figure 1.1: IFC principles (simplified), according to (Schildknecht, 2023)

The data model shown above is simplified and conceptualised. In fact, the data model of IFC is much more structured and uses, amongst other things, inheritance relationships and relationship classes. The figure below shows the same core elements again, but taking into account the most important inheritance relationships from IFC (note: this figure is also a simplified representation).

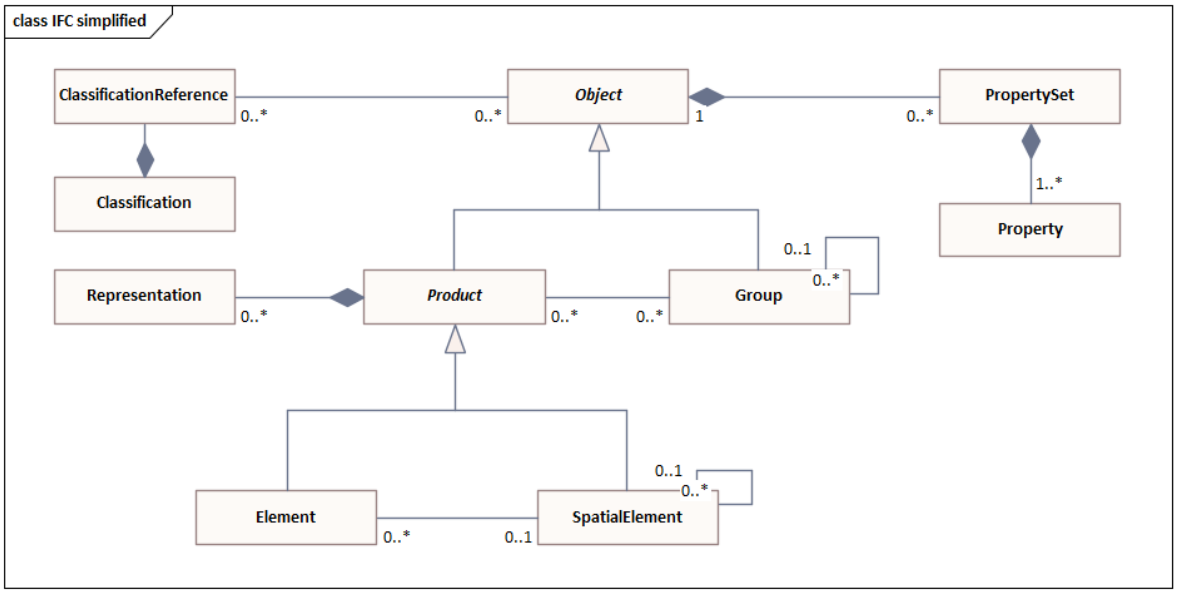


Figure 1.2: IFC schema (extraction, simplified)

1.2 Transformation GIS –> IFC

The transformation of a GIS feature type into the IFC schema takes place as shown in the following figure.

The entity mapping determines for which IFC entity an instance is created for each feature of the feature type.
The attribute values of a feature can be transformed into

- an attribute value of the entity instance
- a property value of the entity instance
- in a group assignment of the entity instance
- in a classification value assignment of the entity instance (not implemented, not shown in figure above)

The GIS geometry is transformed into a “Body” geometry of IFC. In the current implementation, only 2D surface geometries are supported in the source geometry. These are transformed into 3D surfaces (preferably of the tesselation type). In future developments, it is planned to support different geometry types in the feature types and to be able to convert them into different geometry types of IFC.

The GIS geometry is expected to be in WKT format (ISO 19125-1, 2006). If the geodata source is in INTERLIS format (eCH-0031 iliRefMan, 2024), a preprocess must be run to transform it to the WKT format (e.g. ili2pg).

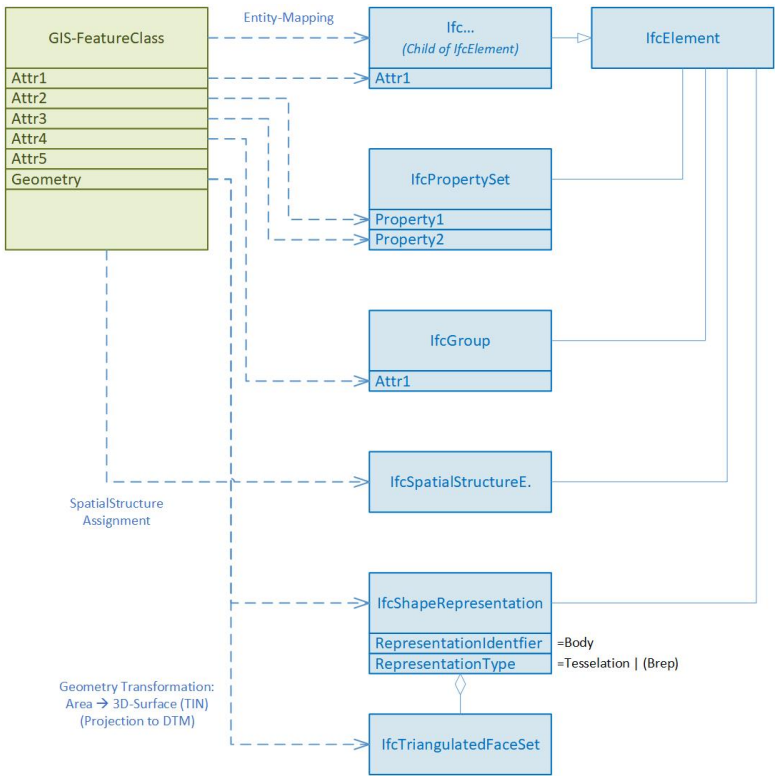


Figure 1.3: Transformation GIS–> IFC, conceptual view

1.3 Projection and triangulation

Based on the available digital terrain model (DTM) represented as uniformly sampled grid points any 2D polygon object is converted into a 3D surface object. The 2D polygon object is assumed to be represented as WKT-string and to have ***no circular arcs***.
First, all grid points within a specific buffer (user-definable argument) around the 2D polygon object are extracted. A 2D Delaunay triangulation is applied to the retrieved subset of grid points to obtain a triangulated irregular network (TIN). Then, the vertices of the polygon object are projected onto the surface by using raytracing along the z-unit vector (0,0,1). For each line segment of the polygon object a vertical plane is defined and intersection points of all triangle edges are calculated. The new surface object with all grid points within the polygon object and a boundary consisting of all vertices and intersection point is defined. The new surface is again triangulated using a 2D Delaunay triangulation. To reduce the number of triangles it is possible to apply a simplification of the TIN by specifying the maximum acceptable height error (user-definable argument). The following figure shows the geometry conversion schematically.

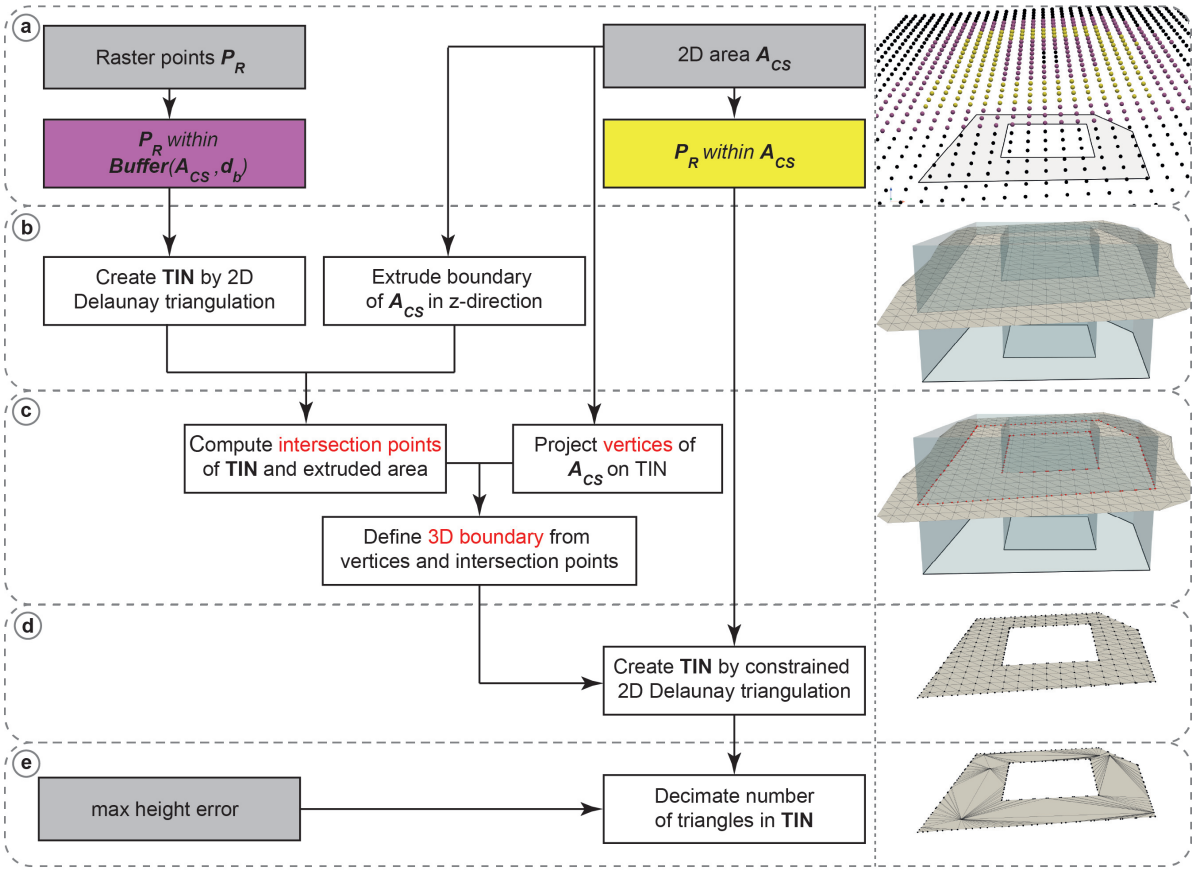


Figure 1.4: Schematic illustration of geometry conversion

The resulting 3D surfaces fulfill the 2D area constrains which are relevant for land coverage and property layer. However, since for every 2D polygon object a subset of grid points is triangulated by a 2D Delaunay triangulation, which does not find an optimal solution in the case of uniformly distributed grid points, there may be some small holes between two consecutive objects. This problem can be mitigated by using a 3D triangulation method instead, which is, however, much more computationally expensive.

Note: The conversion process was reworked to fix problems with holes on shared edges. Key changes include:

- **Two-tier raster point extraction**: Separate buffer zone points (for grid structure) and strictly internal points (for triangulation)
- **Grid-based boundary densification**: Polygon edges are densified using intersections with horizontal, vertical and diagonal grid lines instead of raytracing
- **Constrained Delaunay triangulation**: Single triangulation with boundary constraints instead of the two-step projection approach
- **Post-triangulation height assignment**: Z-coordinates computed via barycentric interpolation after 2D triangulation, rather than raytracing projection

1.4 CityGML / 3D city model

In this project the transformation of 3D city models is based on CityGML, version 2 (Gröger et al., 2012).
The data model of CityGML comprises different thematic modules. In addition to the Core module, only the Building module is processed for the transformation of the 3D city model.

1.4.1 Building structure

Within the Building module, the two classes (features) `Building` and `BuildingPart` are taken into account, including their bounding surface objects `_BoundarySurface`, such as, for example, `RoofSurface`, `WallSurface`, `GroundSurface` etc.
Finer structuring of the building, such as `BuildingInstallation`, `BuildingFurniture` or `Room`, has not been considered in this project, since no data for these classes are available in the used source dataset (SwissBuildings3D).

For the mapping between CityGML and IFC the aggregation hierarchy between `Building` and `BuildingPart` of CityGML is converted into a hierarchy between building elements (child elements of `IfcBuiltElement`) and spatial structure elements (child elements of `IfcSpatialStructureElement`) using the spatial containment concept (`IfcRelContainedInSpatialStructure`).

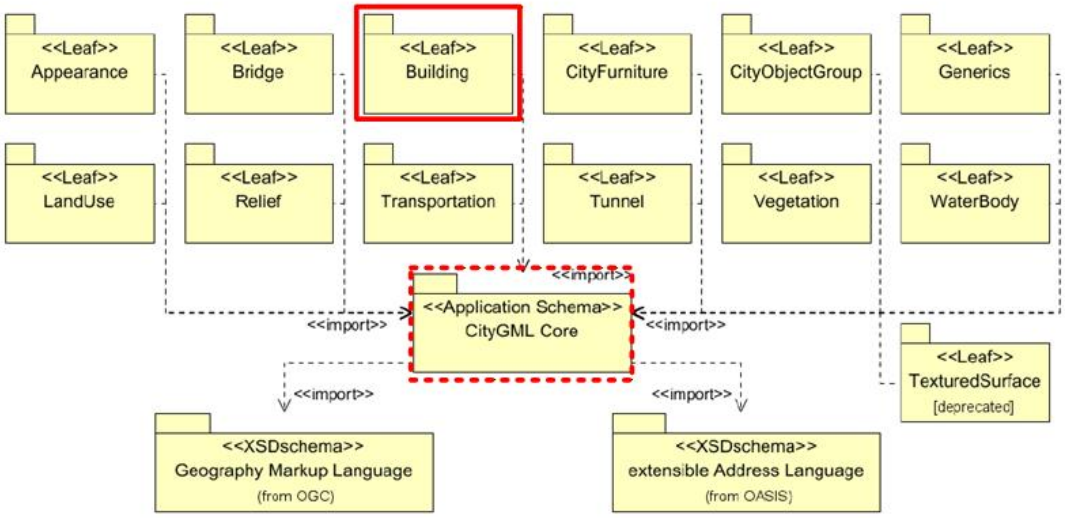


Fig. 8: UML package diagram illustrating the separate modules of CityGML and their schema dependencies. Each extension module (indicated by the leaf packages) further imports the GML 3.1.1 schema definition in order to represent spatial properties of its thematic classes. For readability reasons, the corresponding dependencies have been omitted.

Figure 1.5: CityGML v2, Modules (based on (Gröger et al., 2012))

1.4.2 Geometry types

CityGML supports different geometry types of GML. For the transformation of 3D city models with a focus on buildings, the support of solid and simple surface geometries is sufficient. The following figure shows the geometry types relevant and supported for the transformation of buildings in this project (i.e. the simple geometries `Solid`, `Polygon` with `LinearRing` and the composite geometries `CompositeSolid`, `CompositeSurface` and `MultiSurface`)

The conversion from the GML geometry type to the IFC geometry type is defined according to the following table:

Table 1.1: CityGML geometry IFC mapping

GML Geometry	IFC Geometry
Solid	Brep
CompositeSolid	Brep
Multisurface	Tessellation

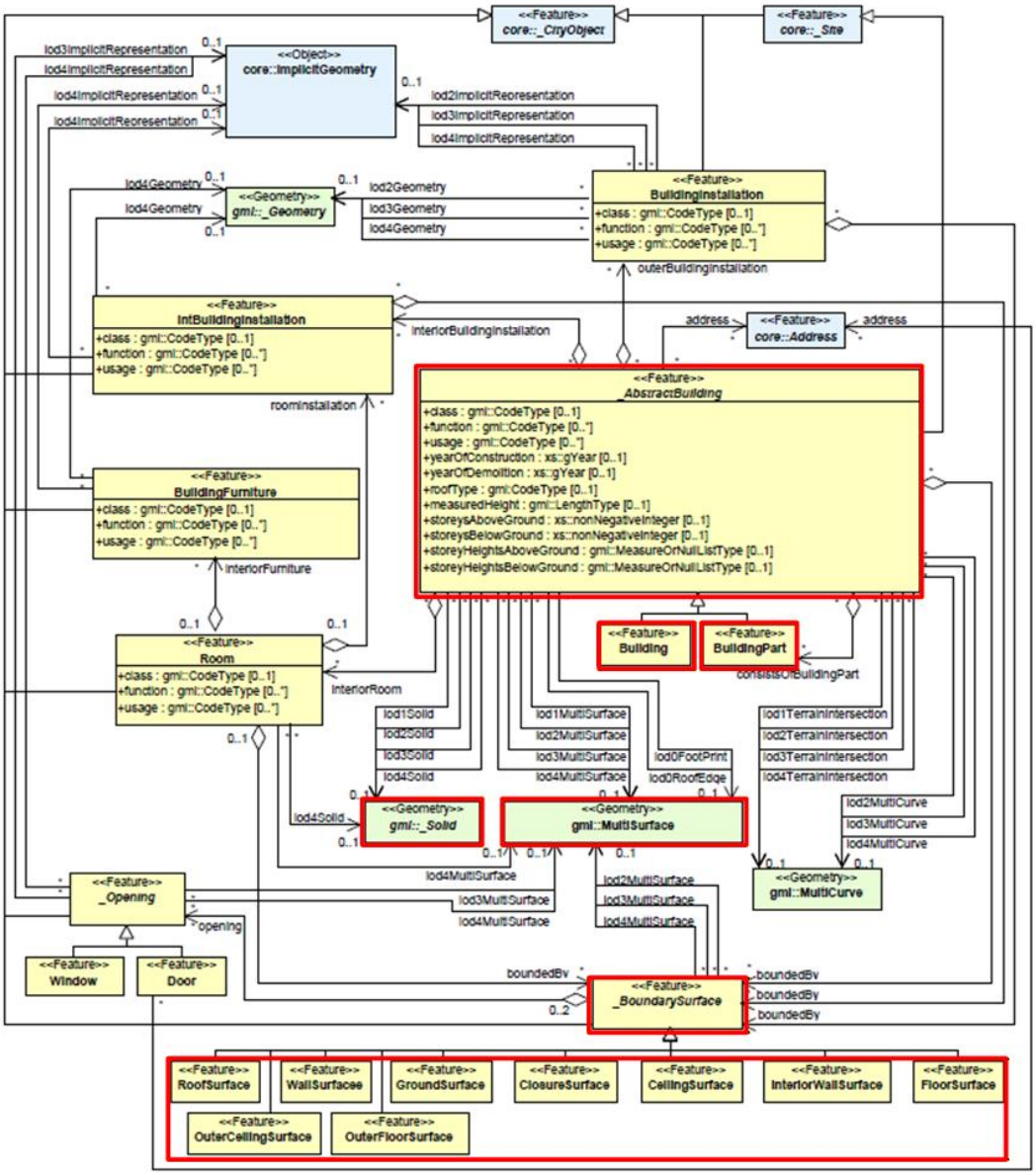


Fig. 27: UML diagram of CityGML's building model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Building* module.

Figure 1.6: CityGML v2, Building model (based on (Gröger et al., 2012))

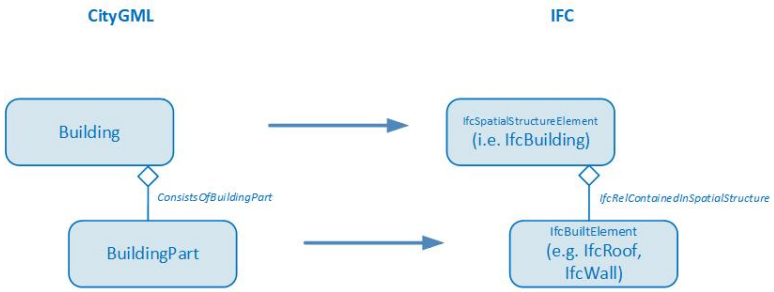


Figure 1.7: CityGML hierarchy mapping to IFC

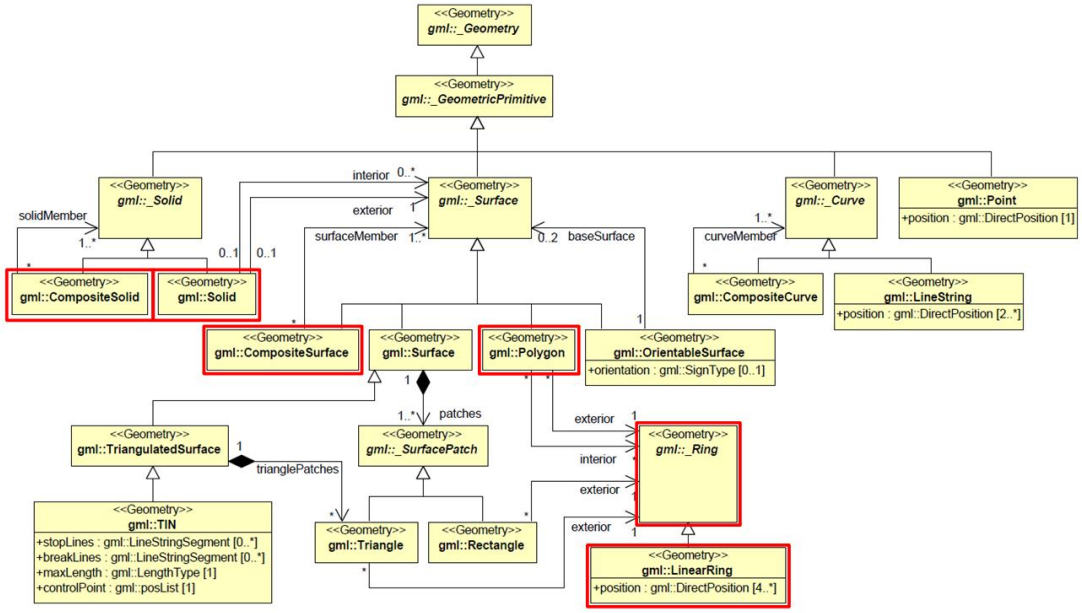


Fig. 9: UML diagram of CityGML's geometry model (subset and profile of GML3): Primitives and Composites.

Figure 1.8: CityGML v2, geometry primitives (based on (Gröger et al., 2012))

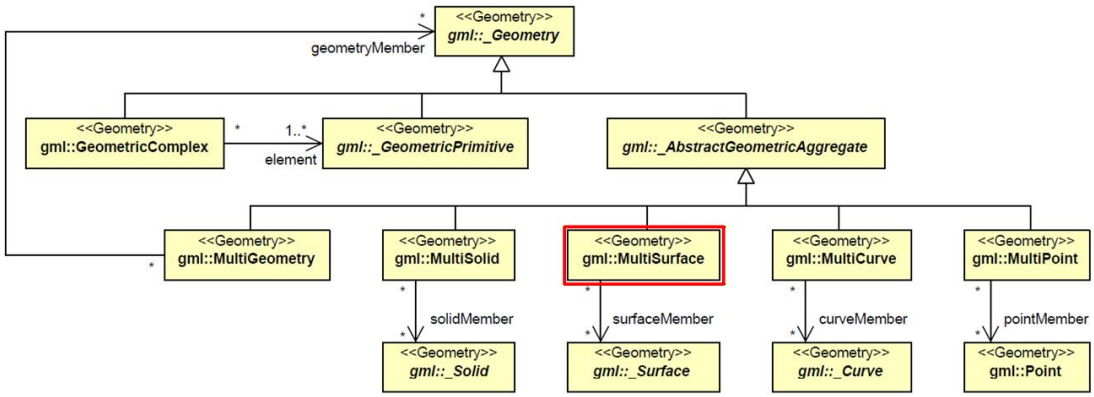


Fig. 10: UML diagram of CityGML's geometry model: Complexes and Aggregates

Figure 1.9: CityGML v2, geometry complexes (based on (Gröger et al., 2012))

1.5 Extrusions

A typical use case for transforming geodata into IFC format involves utility cadastre data. In Switzerland there is a standard data model for utility cadastre called "LKMap", which is specified in (SIA 405, 2025) and (SIA 4008, 2025). This information is often referred to as 2.5D, since it is primarily defined in 2D with additional height data.
To enable the conversion of pipe elements to IFC, the transformation needs special geometry conversion functions that can be used to generate extruded geometries. In accordance with the principles of SIA 405, two main methods can be distinguished for creating solid geometries from the 2.5D data in LKMap using extrusion:

- Extrusion along a polyline:
  A cross-section is extruded along a 3D polyline. This method is used specifically for pipes (LKLine).
  In LKMap, the standard cross-sectional shapes are circle, ellipse, and rectangle. In specialized building information models, it is generally possible to specify freely definable cross-sectional shapes that deviate from the standard profiles. However, this is not possible with LKMap.
- Vertical extrusion:
  A base area is extruded vertically between two elevation points. This method is used specifically for shaft structures (LKPunkt, LKFlaeche). The base area can be defined by any 2D polygon. However, circular base areas (shafts) are also common in the utility cadastre.

For vertical extrusion of utility cadastre objects, two main cases can be distinguished:

- Point objects with standardized cross-sections
- Area shaped objects (surface)

In the GIS cadastre, all geometric data is defined in global coordinate systems (e.g. LV95). In GIS cadastres, point objects are defined solely as point geometry in global coordinates (e.g. LV95). Symbols can be used to represent the point object with a geometric shape. For area objects, however, the entire geometric shape (the surface geometry) is defined in global coordinates. The shape of the surface can be defined arbitrarily. This fact is also taken into account in the transformation functions of the extrusion, so that three basic methods can be distinguished:

- POLYGON: For the extrusion of pipes along an axis (polygon).
- POINT: For the extrusion of point objects.
- SURFACE: For the extusion of area/surface objects.

The following figure shows a schematic representation of the different types of extrusion:

Depending on the type of extrusion and the cross section, different additional parameters are required to define the extrusion algorithm. The following table lists the extrusion parameters required for each type of extrusion, while the table below explains each parameter in detail.

Table 1.2: Extrusion parameter usage

extrusion_type	cross_section_type	polyline	point_start	point_end	height	width	orientation	polygon
POLYLINE	CIRCLE, EGG	x				x		
POLYLINE	RECTANGLE	x			x	x		
POLYLINE	POLYGON_LOCAL	x						x(1)
POINT	CIRCLE		x	x		x		
POINT	RECTANGLE		x	x	x	x	x	
POINT	POLYGON_LOCAL		x	x			x	x(1)
SURFACE	POLYGON_GLOBAL		x	x				x(2)

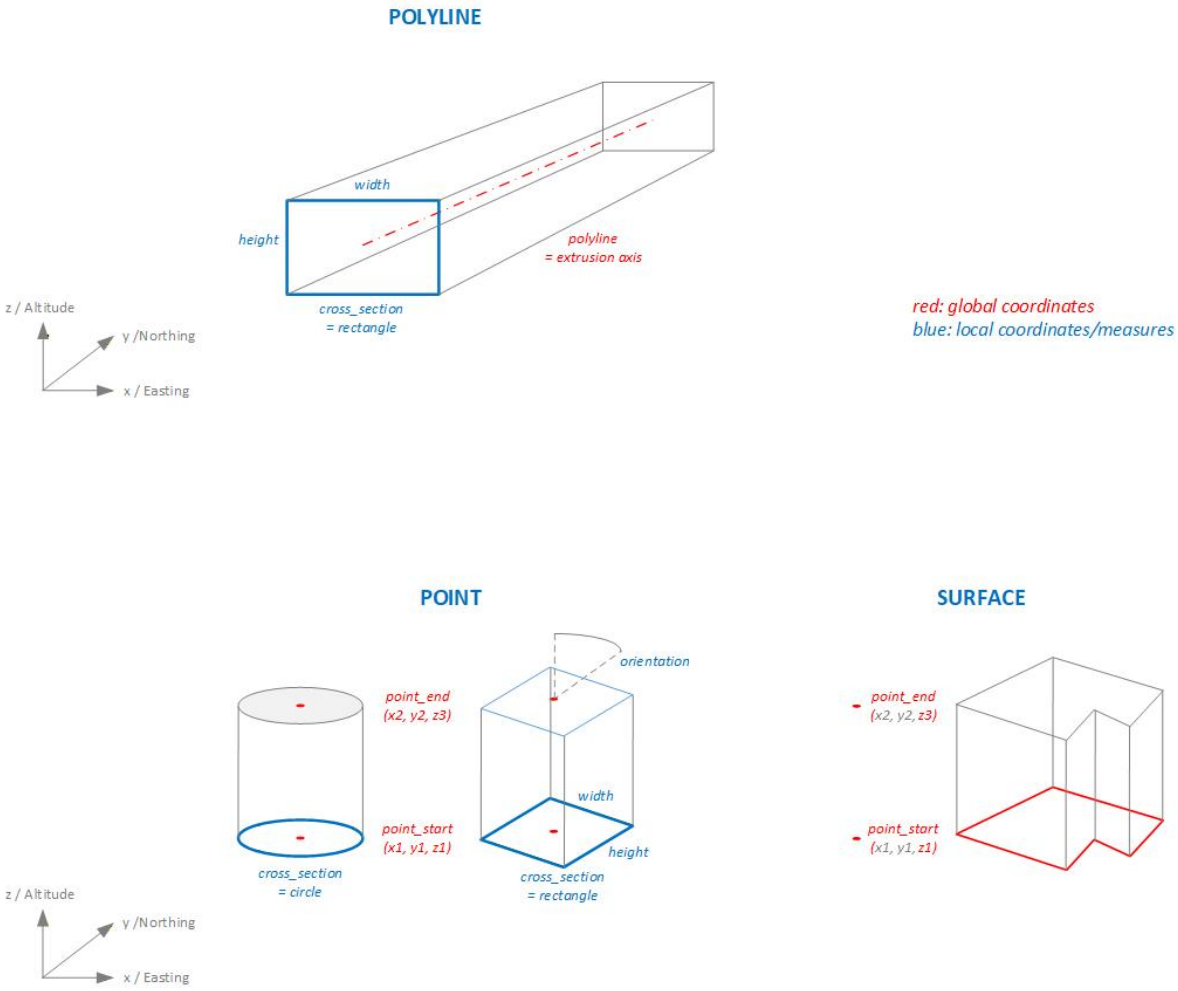


Figure 1.10: Extrusion types

(1) Polygondefinition in lokalem Koordinatensystem

(2) Polygondefinition in LV95

Table 1.3: Extrusion parameters

Parameter	Beschreibung
extrusion_type	POLYLINE extruding along 3D polyline.POINT extruding point geometry between two points.SURFACE extruding surface geometry. Extrusion vector and lenght is defined by two points
cross_section_type	CIRCLEEGGRECTANGLEPOLYGON_LOCAL arbitrary cross sections defined in local coordinate systemPOLYGON_GLOBAL arbitrary cross sections defined in global coordinate system
polyline	Extrusion axis for extrusion_type = POLYLINE;3D polyline in WKT format.The extrusion axis is centered in the cross-section with respect to both height and width.
point_start	Starting point for extrusion for extrusion_type = POINT, SURFACE;3D point in WKT format, global coordinate system (e.g. LV95)
point_end	Ending point for extrusion for extrusion_type = POINT, SURFACE;3D point in WKT format, global coordinate system (e.g. LV95)
height	Height of the cross section in [m]
width	Width of the cross section in [m]
orientation	If extrusion_type = POINT and cross_section_type = RECTANGELE, POLYGON. Orientation of the cross section.Degrees 0.0 .. 359.9

Parameter	Beschreibung
polygon	Cross section, 2D polygon in WKT format in [m].If extrusion_type = *POLYLINE*: A cross-sectional area defined in a local coordinate system in which the extrusion axis passes through the origin (0,0). The cross-sectional area is always defined as being orthogonal to the extrusion axis.If extrusion_type = *POINT*: A cross-sectional area in local coordinate system. The area is always defined in the xy-plane of the parent, absolute coordinate system (e.g. LV95).The cross-sectional area is extruded between the points point_start and point_end without being rotated relative to the xy-plane.If extrusion_type = *SURFACE*: Surface in absolute coordinate system (e.g. LV95). The surface is always defined in the xy-plane of the parent coordinate system (e.g. LV95). The surface is extruded between the z-values of the points point_start and point_end and in the direction defined by point_start and point_end, without being rotated relative to the xy-plane.

The conversion of the extrusion types described above into IFC geometry types is performed according to the following mapping table:

Table 1.4: Extrusion types IFC mapping

extrusion type	cross section type	IFC geometry type
POLYLINE	CIRCLE	IfcSweptDiskSolid
POLYLINE	EGG, RECTANGLE, POLYGON	IfcFixedReferenceSweptAreaSolid
POINT, SURFACE	[all]	IfcExtrudedAreaSolid

2 Mapping Principles

2.1 Basics

To convert a typical geodataset to IFC, both semantic mappings and geometric conversions must be defined, see figure below.

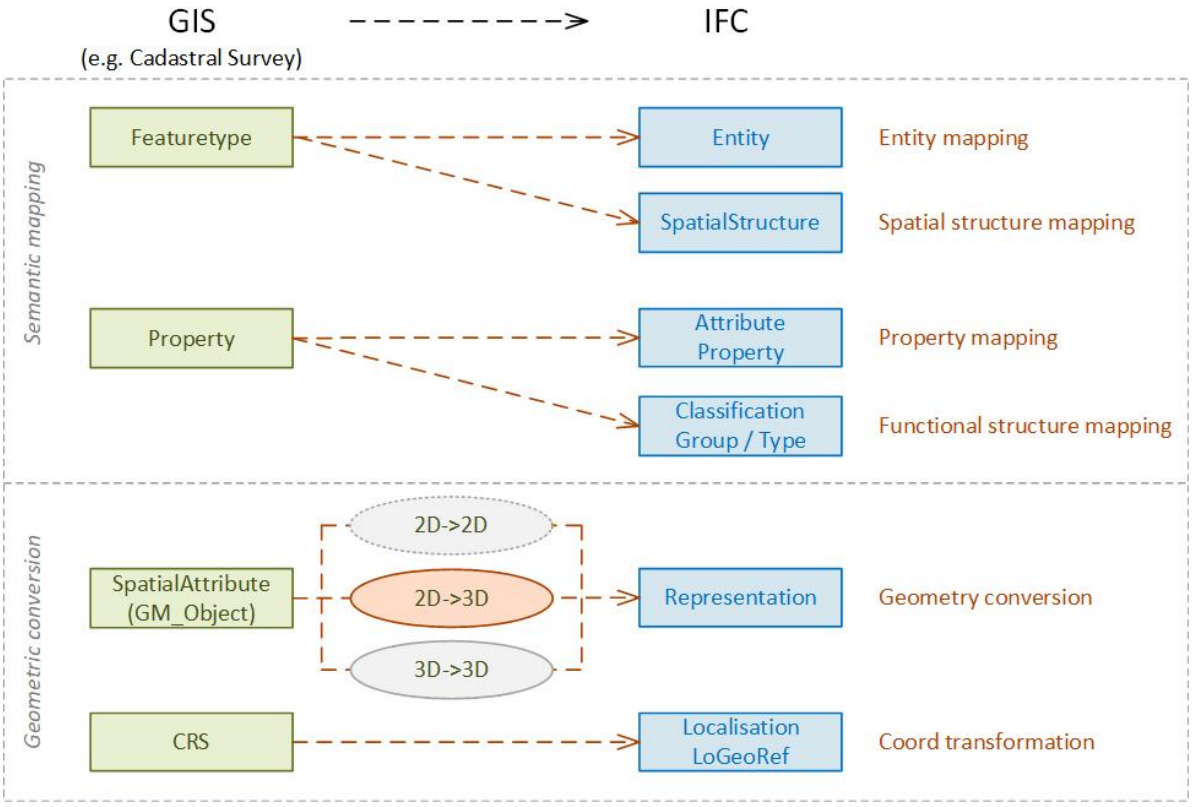


Figure 2.1: Mapping Principles Overview ((Schildknecht et al., 2025a))

In (Schildknecht et al., 2025a), the various semantic and geometric mappings are described and discussed in detail.

The semantic mapping implemented with cs2bim comprises five mapping levels (see also figure below):

- Entity mapping
- Property mapping
- EntityType mapping
- Spatial structure mapping
- Group mapping

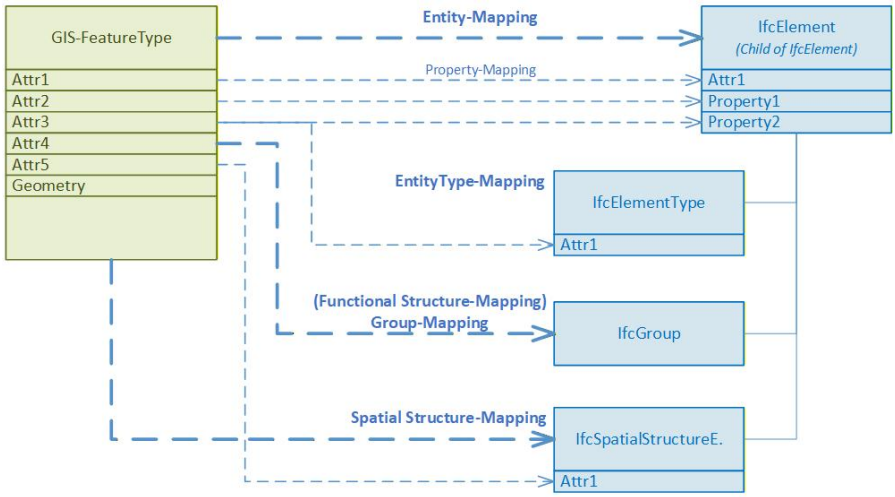


Figure 2.2: Mapping Principles Entities

The geometric conversion is shown schematically in the figure below. Based on the defined conversion type, an IFC geometry is generated from the geometry of the geodata instance and assigned to the generated IFC instance. This process always involves at least one geometry type conversion from a WKT definition to an IFC definition. Depending on the source dataset, additional processing of the geometry may also occur — e.g., from 2D to 3D.

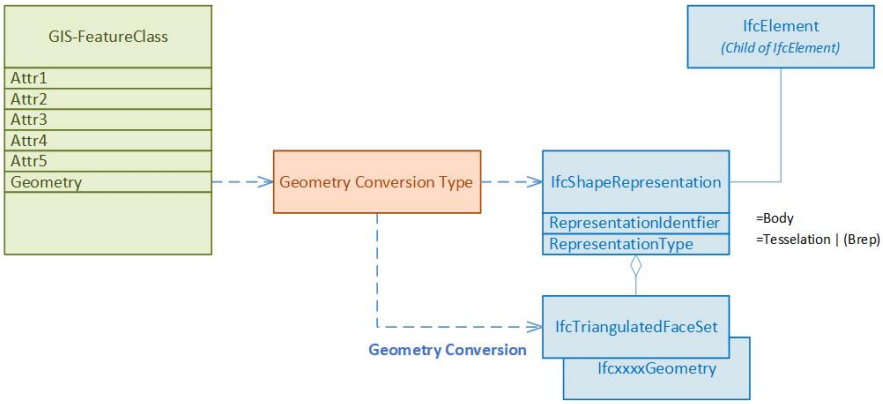


Figure 2.3: Mapping Principles Geometry

2.2 Principles

Part II

Application

3 Architecture

3.1 System architecture

3.1.1 Context

The service cs2bim is designed with the following conceptual components:

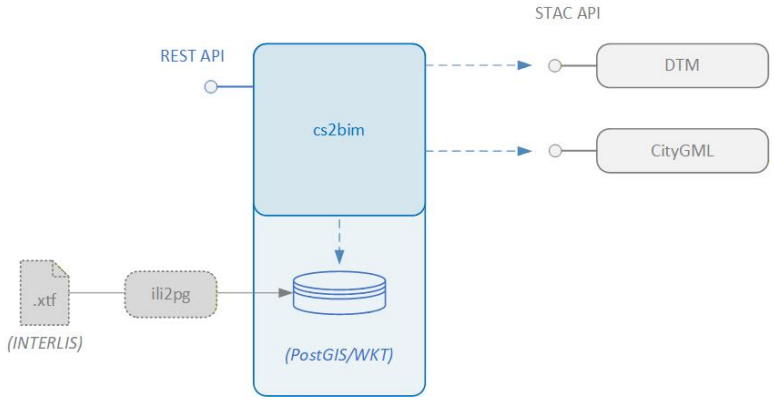


Figure 3.1: CS2BIM System Architecture

Component	Description
cs2bim	Main component to transform feature types into IFC instances.Including transformation algorithms, IFC serialisation, service task management etc.
PostGIS/WKT	Database holding the GIS feature types. In current state, this is a PostGIS database.The component cs2bim has a postgres connection to the database.
DTM	Data store with Digital Terrain Model data. DTM data is expected in format XYZ. The cs2bim component gets the DTM data over a STAC API request.
CityGML	Data store with CityGML data. CityGML data is expected in version 2 of CityGML. The cs2bim component gets the CityGML data over a STAC API request.
REST API	REST API to interact with cs2bim: Start processing, get status of processing, get generated IFC file.
ili2pg	External component to transform INTERLIS data into PostGIS. This must be done as a standalone pre process.

The following architectural enhancements are already being discussed and ***proposed as potential next steps*** for the further development and improvement of the service:

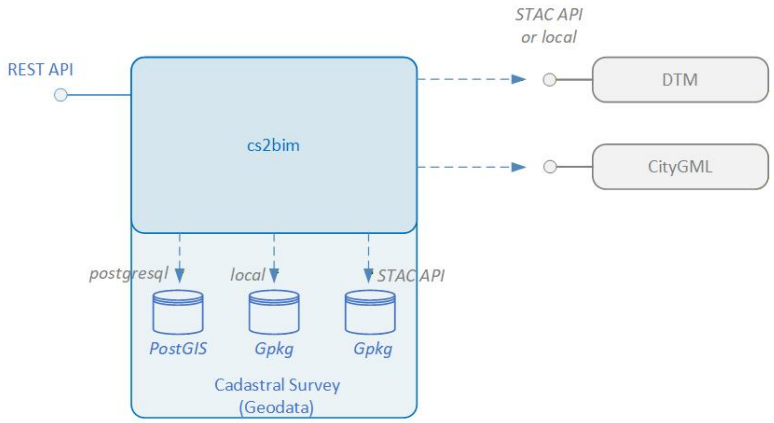


Figure 3.2: CS2BIM System Architecture, look ahead

- Geopackage as additional data source format for GIS feature types.
  - Geopackage files could be stored locally (on the server) or could be accessed via STAC API
- DMT files stored locally (on the server)
- CityGML files stored locally (on the server)

These improvements would simplify the "local" execution of the service as a "standalone" application, as originally intended.

3.1.2 Internal

3.1.2.1 Services

The system architecture is set up using three Docker services.

The api and worker services are built from a custom Docker image based on `python:3.10`, which contains the application code base and all required dependencies. Both services share the same image but use different entry points. The redis service is created using the standard `redis:7` image.

3.1.2.1.1 api

The entry point for this service is the `api.app` module. It is responsible for routing user requests. There are three endpoints that let the user interact with the application:

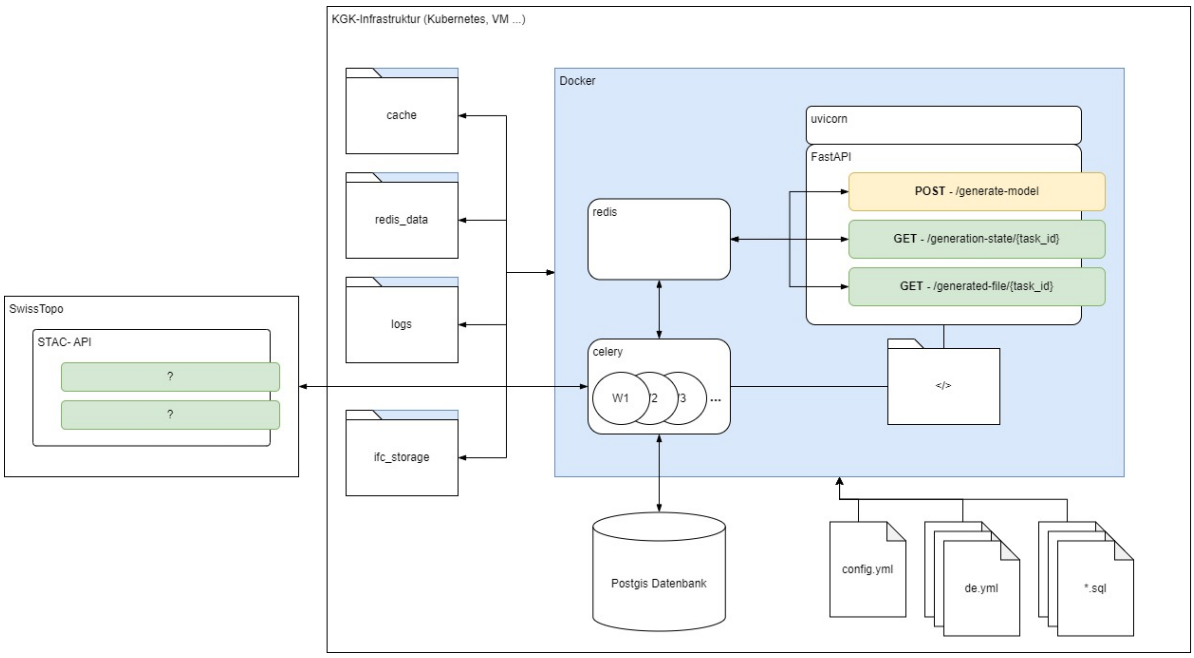


Figure 3.3: CS2BIM System Architecture

- `POST /generate-model` – triggers the generation of an IFC model. Returns a `task_id` that can be used to track progress and retrieve the result.
- `GET /generation-state/{task_id}` – returns the current state of a generation task.
- `GET /generated-file/{task_id}` – returns the generated IFC file once the task is completed.

The service uses FastAPI as the web framework and Uvicorn as the ASGI server to host it.

3.1.2.1.2 worker

The entry point for the worker service is the `worker.app` module. The service runs a Celery instance that manages and executes IFC model generation tasks. It connects to the required data sources (PostGIS database and SwissTopo STAC API) that provide the data needed for model generation.

3.1.2.1.3 redis

The redis service serves two purposes: as the message broker for the worker service (managing task queues between api and worker) and as the cache management layer, tracking which files are cached and whether cached files need to be refreshed.

3.1.3 Docker Volumes

To store data produced during run-time, the docker-compose file defines four Docker volumes:

- **redis_data** – persists all data stored in Redis. Prevents data loss on service rebuilds.
- **ifc** – stores generated IFC files.
- **logs** – stores log files for the api and worker services.
- **cache** – stores cached data fetched from external data sources.

3.1.4 Volume Mappings

All configurable files are mapped from the host machine into the containers. This includes translation files (i18n), SQL query files, and the application configuration (`config.yml`). This way they can be easily edited without rebuilding the containers.

4 Configuration Overview

The application is configured using a YAML file. Below, you'll find an overview of the main configuration concepts. You can find the full and more technical documentation of the configuration file here.

The configuration is divided into several sections:

- **Logging**
  Set the application's log level (e.g., DEBUG, INFO).
- **Connections**
  Provide connection settings for Redis and Database access.
- **Internationalization (i18n)**
  Specify translation files to support multiple languages.
- **External Data (STAC, TIN)**
  Configure where the application remote data sources (STAC APIs) can be found.
- **IFC Export**
  Define how data will be structured and exported to IFC.

The concepts and principles of the configuration and the most important parameters are described in the following sections.

4.1 Internationalization Support

The configuration supports referencing translation files for multiple languages. The values affected by translation are fixed and not configurable:

- Attribute values
- Property set names
- Property names
- Property values
- Group names

If a translation key cannot be found, the original value is written unchanged. The following characters are allowed for keys in the table: letters (lowercase), numbers and underscore (_).

Keys derived from dynamic values are generated as follows:

1. key = lower_case(key)
2. key = remove_special_characters_except_spaces(key)
3. key = trim_leading_and_trailing_spaces(key)
4. key = replace_sequences_of_spaces_with_single_underscore(key)

Example: “Year (1990)” → “year (1990)” → “year 1990” → “year 1990” → “year_1990”

4.2 External Data

The projection feature types require a DTM file source provided via a STAC API. The building feature types require a CityGML file source. These URLs must be defined in the STAC configuration.

4.2.1 DTM

Needs to be set if there are projection feature types configured. Expected asset properties:

- type = application/x.ascii-xyz+zip
- eo:gsd / gsd = config.tin.grid_size

4.2.2 Buildings

Needs to be set if there are building feature types configured. Expected asset properties:

- type = application/x.gml+zip

4.3 IFC Export

4.3.1 Geo referencing

You can provide the so-called “Level of Georeferencing” (LoGeoRef), according to (Clemen&Görne, 2019) [^LoGeoRef].
The different levels represent different methods of defining information about georeferencing in IFC.
Supported values are:

- LO_GEO_REF_30
  - IfcObjectPlacement of an IfcSpatialStructureElement contains georeferencing
  - Suitability for local projects on a smaller scale

  - IFC 2.3

- LO_GEO_REF_40
  - IfcGeometricRepresentation context of IfcProject contains georeferencing
  - Suited for larger infrastructure projects
  - IFC 2.3

- LO_GEO_REF_50
  - IfcMapConversion defines georeferencing of the “SurveyPoint”, including coordinate system parameters
  - Suited for large-scale and linear project expansions
  - IFC 4.0

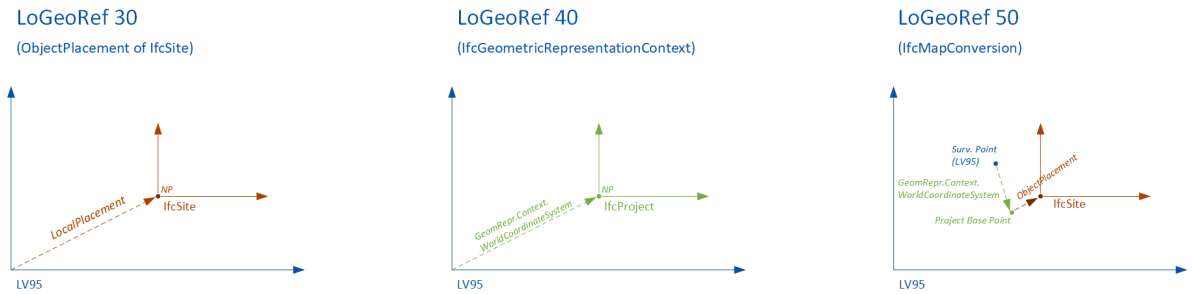


Figure 4.1: Levels of Georeferencing LoGeoRef

Coordinates and Offsets
You can provide a project origin in LV95 coordinates (Easting, Northing, Height). The project origin can also be set to (0,0,0).
If not provided, the system sets a project origin in the lower left corner on a minimum bounding box of the perimeter.

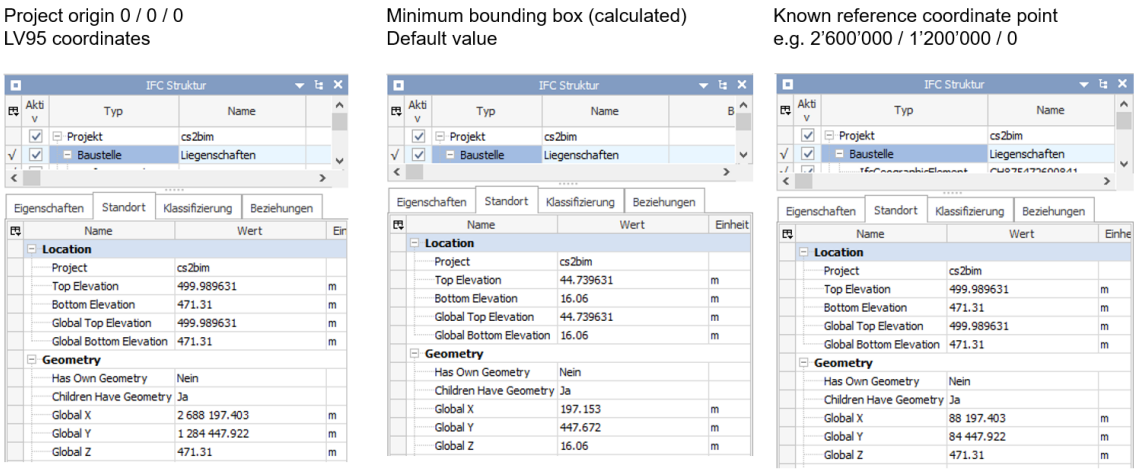


Figure 4.2: Levels of Georeferencing LoGeoRef

4.3.2 Feature types

A "feature type" is the definition of a set of objects that are exported as instances of an IFC entity with common definitions. Each feature type is defined through a configuration that describes how data from the GIS database is transformed into IFC instances. For every configured feature type, a set of instances is created. Currently, three different kinds of feature types are supported. They differ according to the type of geometry conversion.

- **Projection** (projection_feature_types)
  Projections are feature types that generate IFC structures by projecting 2D areas onto a 3D surface (in this case the terrain model).
- **Building** (building_feature_types)
  Buildings are feature types that are generated by transforming 3D building geometries from CityGML into IFC structures.
- **Extrusion** (extrusion_feature_types)
  Extrusions are feature types that are generated by extruding 2D areas along a polyline. Five different cross-section types (CIRCLE, EGG, RECTANGLE, POLYGON_GLOBAL, POLYGON_LOCAL) and three different extrusion types (POLYLINE (horizontal) , SURFACE (vertical) and POINT (vertical)) are supported.

4.3.3 SQL

The basis of a feature type is a SQL statement, that selects data from a geodata source. The SQL statement is stored in a separate file. It is executed at the beginning of constructing of

the feature type instances. It determines what objects to create for the feature type. The input parameter "%(polygon)s" is the input WKT string of the application. It is expected that this input parameter is used to select the relevant feature type instances for the request. Each result row is converted into an IFC instance. The required return columns differ from feature type to feature type.

- Projection: Expects a column with the name `wkt`[2d_wkt_polygon] that represents the area to be projected.
- Building: Expects a column with the name `egid`[number] that contains the egid number to identify the building.
- Extrusion: Extrusions are a bit more complicated because there are different cross-sections and extrusion types that require different columns. The columns `cross_section` [cross_section_type] and `extrusion_type` [extrusion_type] are always mandatory and used to identify the cross-section and extrusion type. Depending on the `cross_section` and `extrusion_type` additional columns are needed. See this section for more details.

Additional columns can be included to provide values that can be referenced in the attribute, property, or group configurations, see this section for more details.

Selecting the data by SQL is flexible and can be done in various ways. Specifically use geometry functions (e.g. intersections) and aggregation functions. There are some examples in the sql folder.

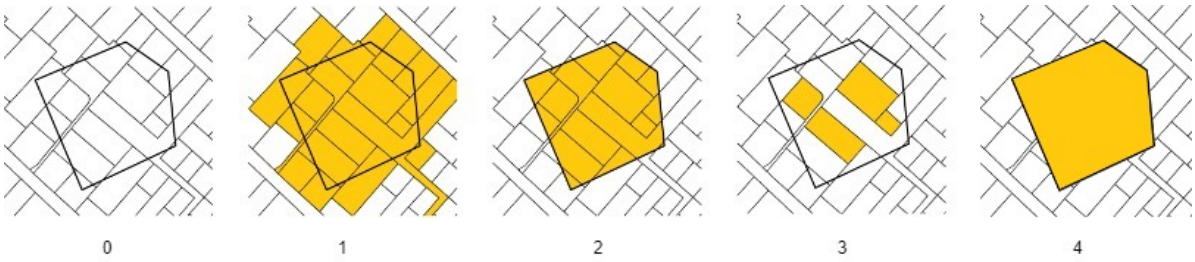


Figure 4.3: Polygon

1. The input polygon
2. All areas that intersect with the polygon are included (Parcels, Land covers, Land covers buildings)
3. All areas are cut off at the border of the polygon (Parcels intersection, Land covers intersection)
4. All areas that are fully contained by the polygon are included (Parcels contains, Land covers contains)
5. The entire area of the input polygon without subdivisions (Polygon)

Useful postgis functions

ST_GeomFromText: Constructs a PostGIS ST_Geometry object
ST_AsText: Returns the OGC WKT representation of the geometry
ST_CurveToLine: Converts a given geometry to a linear geometry
ST_Intersects: Returns true if two geometries intersect. Geometries intersect if they have any point in common. **ST_Contains**: Returns true if the first geometry contains the second.

Hint: The wildcard character '%' needs to be escaped like this '%%'.

4.3.4 Entity Mapping

Specifies the IFC entity used to create instances for feature types. The configuration expects a string containing the exact name of the entity as defined in the IFC schema.

Not all entities are supported by all IFC versions, so only entities that are available across the supported versions should be used. An exception to this rule applies to IfcBuiltSystem / IfcBuildingSystem. If either of these entities is configured, the system will automatically create the appropriate entity for the selected IFC version.

Incorrect or unsupported entity names will result in errors. The responsibility for providing a valid and compatible entity name lies with the user.

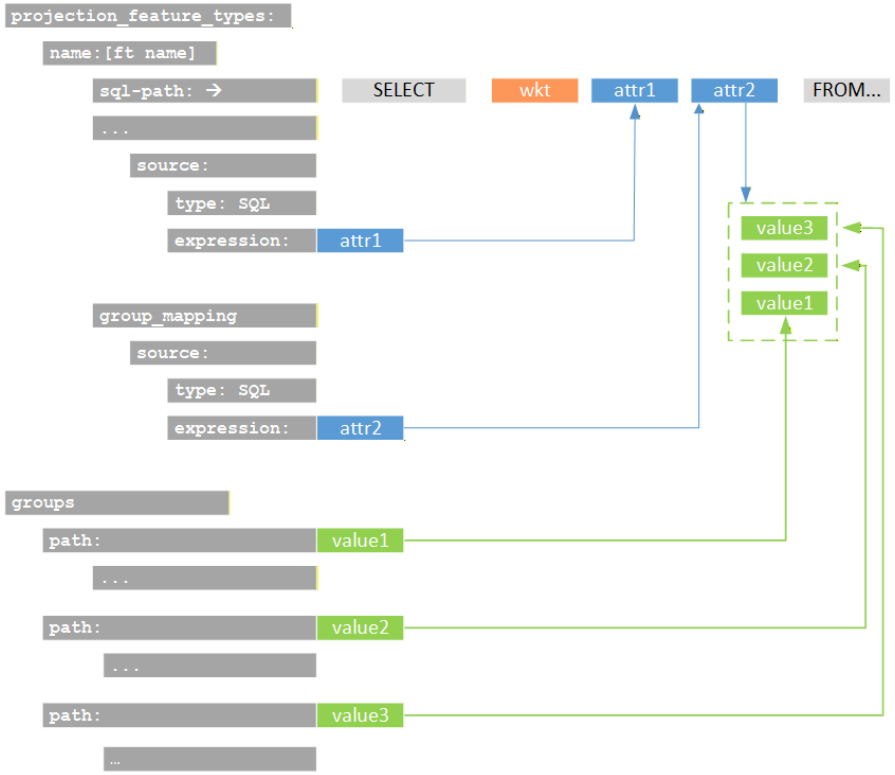
4.3.5 Attributes, properties and group mappings

To define a value for an attribute, property or group name, the source needs to be defined. The `source` is defined by a `type` and an `expression`. There are three different types of sources to evaluate the values of attributes, properties and group names:

- `SQL`: The value is determined by an attribute of the SQL statement.
- `STATIC`: The value is static and defined directly in the configuration
- `CITY_GML` (for buildins feature types only): The value is determined by an XPath expression

It is important that the expression matches the type being used. An SQL source expects a column name to retrieve a value from the SQL result set. A CityGML source expects an XPath expression that selects the desired value (starting from bldg:Building). The static source does not resolve any value and therefore has no requirements for the expression.
Attributes cannot be freely selected and depend on the ifc entity that is selected. If a configured attribute does not exist, it is ignored. Properties that are configured but not found as output column in the sql result are also ignored.

The following figure shows schematically the attributes of the sql statement and their referene as parameters in the configuration.



4.3.6 Spatial Structure Mapping

Specifies the IFC spatial structure element that a created instance is linked to. Spatial structures are shared among instances if their attribute or property values are identical — even across different feature types. As a result, the total number of spatial structures ranges from one up to the number of feature type instances.

4.3.7 Entity Type Mapping

Specifies an IFC entity type linked to each created instance. Entity types are shared among other instances of this feature type if their attribute or property values are identical. Entity types are only supported by certain IFC entities. The responsibility for configuring this correctly lies with the user. Configured entity types that do not exist are ignored. Attributes cannot be freely selected and depend on the ifc entity that is selected. If a configured attribute does not exist, it is ignored.

4.3.8 Groups

Each feature type instance can be assigned to one or more groups. This configuration is optional—if omitted, no group assignment will be made. For every group assignment, the system creates an IFC group based on the specified group configuration, including its parameters such as entity and any number of attributes or properties. If no configuration exists for a given assigned value, the system will generate a basic IFC group entity without additional attributes or properties. When defining groups, the "." character can be used to create nested group structures.

For example, the group "Amtliche Vermessung.Bodenbedeckung.befestigt" results in the creation of three nested groups. The feature type instances are assigned to the final group in this hierarchy.

- Amtliche Vermessung
  - Bodenbedeckung
    - befestigt
      - Feature type instance 1
      - Feature type instance 2
      - Feature type instance 3
      - ...

4.3.9 Special configurations (feature type specific)

In this section some feature type specific configurations and concepts are described.

4.3.9.1 Building (building_feature_type)

The processing of buildings from CityGML is based on the EGID (a unique identification for all buildings in Switzerland). In the first step, all EGID located within the processing perimeter are identified. To do this, the EGID are queried using an SQL statement from a geodata source within the database (attribute `egid` in the SQL statement).
In a second step, the corresponding building objects are identified within the CityGML files. This is done via an attributive selection, i.e., the EGID must be included in the CityGML records. The parameter `egid_xpath` defines the XQuery expression through which the EGID number is defined within the CityGML objects.

The parameter `egid_xpath` defines an XQuery expression used to query a geometry definition within CityGML. This parameter can be used to define both the LOG to be processed and the BuildingParts to be processed by formulating the appropriate XQuery expressions.

4.3.9.2 Extrusion (extrusion_feature_type)

Extrusions are a bit more complicated because there are different cross-sections and extrusion types that require different columns. The columns `cross_section` [cross_section_type] and `extrusion_type` [extrusion_type] are always mandatory and used to identify the cross-section and extrusion type. See this table in this section that shows the valid values for these columns and the additional required columns for each valid combination.

For a list of all extrusion parameters see also this table.

5 Configuration

Root application configuration.

This class defines the complete configuration model for the application. It aggregates settings for logging, internationalization (i18n), Redis and PostGIS database connections, STAC data sources, TIN generation, and IFC export configuration. The configuration is typically loaded from a YAML file using the `load()` class method, which supports environment variable expansion.

5.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
log-ging_level	`string`		string		Logging level for the application (e.g., DEBUG, INFO, WARNING)
redis	`object`		RedisConfig		Redis configuration
db	`object`		DBConfig		Database configuration
ifc	`object`		IFCConfig		IFC (Industry Foundation Classes) export configuration
i18n	`object` or `null`		I18nConfig	`null`	Internationaliza-tion (i18n) configuration
stac	`object`		STACConfig		STAC configuration for external data sources

Property	Type	Required	Possible values	Default	Description
tin	`object`		TINConfig		TIN (Triangulated Irregular Network) generation configuration

6 Definitions

6.1 AttributeConfig

Attribute mapping configuration

6.1.0.1 Type: `object`

Property	Type	Required	Possible values	Description
attribute	`string`		string	Attribute name (Only applied if the attribute exists on the entity)
value	`string`		string	Attribute value

6.2 BuildingAttributeConfig

Attribute mapping configuration for building feature type

6.2.0.1 Type: `object`

Property	Type	Required	Possible values	Description
attribute	`string`		string	Attribute name (Only applied if the attribute exists on the entity)
source	`object`		BuildingSourceConfig	Source configuration for this attribute

6.3 BuildingEntityConfig

Entity mapping configuration for building feature type

6.3.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
attributes	`array`		BuildingAttributeConfig	`[]`	List of attribute mappings
properties	`array`		BuildingPropertyConfig	`[]`	List of property mappings
building_parts	`array`		BuildingPartConfig	`[]`	List of building parts belonging to this building entity

6.4 BuildingFeatureType

Feature type configuration for building feature type

6.4.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
name	`string`		string		Feature type name for the building
sql_path	`string`		string		Path to SQL definition for the building feature type. Must return at least a column named 'egid'.

Property	Type	Required	Possible values	Default	Description
egid_xpath	`string`		string		XPath expression to extract EGID identifier from city gml building entities
entity_mapping	`object`		BuildingEntityConfig		Entity mapping configuration for the building
spatial_structure_mapping	`object`		BuildingSpatialEntityConfig		Spatial structure mapping for the building
group_mapping	`array`		BuildingSourceConfig	`[]`	Group mappings for the building feature type

6.5 BuildingPartConfig

Building part configuration for building feature type

6.5.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
entity	`string`		string		Name of entity according to IFC schema, e.g. 'IfcWall'
geometry_mapping	`object` or `null`		GmlGeometryMapping	`null`	Geometry mapping for the building part
color	`object`		Color	`"white"`	Color assigned to the building part

6.6 BuildingPropertyConfig

Property mapping configuration for building feature type

6.6.0.1 Type: `object`

Property	Type	Required	Possible values	Description
property	`string`		string	Property name
property_set	`string`		string	Property set name
source	`object`		BuildingSourceConfig	Source configuration for this property

6.7 BuildingSource

Supported source types for properties and attributes in building feature types

6.7.0.1 Type: `string`

Possible Values: `STATIC` or `SQL` or `CITY_GML`

6.8 BuildingSourceConfig

Source configuration for building feature type

6.8.0.1 Type: `object`

Property	Type	Required	Possible values	Description
type	`string`		BuildingSource	Type of the data source
expression	`string`		string	Expression defining the source data

6.9 BuildingSpatialEntityConfig

Spatial structure mapping configuration for building feature type

6.9.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
attributes	`array`		BuildingAttributeConfig	`[]`	List of attribute mappings
properties	`array`		BuildingPropertyConfig	`[]`	List of property mappings

6.10 Color

Color configuration based on red, green, blue and alpha channels

6.10.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
r	`number`		`0.0 <= x <= 1.0`		Red channel
g	`number`		`0.0 <= x <= 1.0`		Green channel
b	`number`		`0.0 <= x <= 1.0`		Blue channel
a	`number`		`0.0 <= x <= 1.0`	`0.0`	Alpha channel

6.11 CoordinateReferenceSystem

Coordinate reference system for IFC export

6.11.0.1 Type: `object`

Property	Type	Required	Possible values	Description
epsg_code	`string`		string	EPSG code for the coordinate reference system
description	`string`		string	Description of the coordinate reference system
geodetic_datum	`string`		string	Geodetic datum for the coordinate reference system
vertical_datum	`string`		string	Vertical datum for the coordinate reference system

6.12 DBConfig

Postgis connection configuration

6.12.0.1 Type: `object`

Property	Type	Required	Possible values	Description
dbname	`string`		string	Database name
user	`string`		string	Database username
host	`string`		string	Database host address
port	`integer`		integer	Database port number
password	`string`		string	Database password

6.13 ExtrusionAttributeConfig

Attribute mapping configuration for extrusion feature type

6.13.0.1 Type: `object`

Property	Type	Required	Possible values	Description
attribute	`string`		string	Attribute name (Only applied if the attribute exists on the entity)
source	`object`		ExtrusionConfigSource	Source configuration for this attribute

6.14 ExtrusionConfigSource

Source configuration for extrusion feature type

6.14.0.1 Type: `object`

Property	Type	Required	Possible values	Description
type	`string`		ExtrusionSource	Type of the data source
expression	`string`		string	Expression defining the source data

6.15 ExtrusionEntityConfig

Entity mapping configuration for extrusion feature type

6.15.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
entity	`string`		string		Name of entity according to IFC schema, e.g. 'IfcWall'
attributes	`array`		ExtrusionAttribute-Config	`[]`	List of attribute mappings

Property	Type	Required	Possible values	Default	Description
properties	`array`		ExtrusionPropertyConfig	`[]`	List of property mappings

6.16 ExtrusionEntityTypeConfig

Entity type mapping configuration for extrusion feature type

6.16.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
attributes	`array`		ExtrusionAttributeConfig	`[]`	List of attribute mappings
properties	`array`		ExtrusionPropertyConfig	`[]`	List of property mappings

6.17 ExtrusionFeatureType

Feature type configuration for extrusion feature type

6.17.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
name	`string`		string		Feature type name for the extrusion
entity_mapping	`object`		ExtrusionEntityConfig		Entity mapping configuration for the extrusion

Property	Type	Required	Possible values	Default	Description
sql_path	`string` or `null`		string	`null`	Path to SQL definition for the extrusion feature type. Exact specification can be found in the configuration documentation.
entity_type_mapping	`object` or `null`		ExtrusionEntityTypeConfig	`null`	Entity type mapping configuration for the extrusion. (Only supported for entities with TypeObject)
spatial_structure_mapping	`object`		ExtrusionSpatialEntityConfig		Spatial structure mapping for the projection
group_mapping	`array`		ExtrusionConfigSource	`[]`	Group mappings for the projection feature type
color	`object`		Color	`"white"`	Color assigned to the extrusion feature type

6.18 ExtrusionPropertyConfig

Property mapping configuration for extrusion feature type

6.18.0.1 Type: `object`

Property	Type	Required	Possible values	Description
property	`string`		string	Property name
property_set	`string`		string	Property set name

Property	Type	Required	Possible values	Description
source	`object`		ExtrusionConfigSource	Source configuration for this property

6.19 ExtrusionSource

Supported source types for properties and attributes in extrusion feature types

6.19.0.1 Type: `string`

Possible Values: `STATIC` or `SQL`

6.20 ExtrusionSpatialEntityConfig

Spatial structure mapping configuration for extrusion feature type

6.20.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
attributes	`array`		ExtrusionAttribute-Config	`[]`	List of attribute mappings
properties	`array`		ExtrusionPropertyCon-fig	`[]`	List of property mappings

6.21 GeoReferencing

Supported geo referencing methods

6.21.0.1 Type: `string`

Possible Values: `LO_GEO_REF_30` or `LO_GEO_REF_40` or `LO_GEO_REF_50`

6.22 GmlGeometry

Supported gml geometry types

6.22.0.1 Type: `string`

Possible Values: `MULTI_SURFACE` or `SOLID` or `COMPOSITE_SOLID`

6.23 GmlGeometryMapping

Geometry mapping for building part

6.23.0.1 Type: `object`

Property	Type	Required	Possible values	Description
xpath	`string`		string	XPath expression to locate the building part geometry in source data
geometry	`string`		GmlGeometry	Referenced geometry type of the building part

6.24 GridSize

Available grid sizes for projections

6.24.0.1 Type: `number`

Possible Values: `0.5` or `2.0`

6.25 GroupConfig

Group configuration for IFC export

6.25.0.1 Type: `object`

Property	Type	Required	Possible values	Description
path	`string`		string	Path identifier for the group
entity_mapping	`object`		GroupEntityConfig	Entity mapping configuration for the group

6.26 GroupEntityConfig

Entity mapping configuration for group

6.26.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
entity	`string`		string		Name of entity according to IFC schema, e.g. 'IfcWall'
attributes	`array`		AttributeConfig	`[]`	List of attribute mappings
properties	`array`		PropertyConfig	`[]`	List of property mappings

6.27 I18nConfig

Internationalization (i18n) configuration

6.27.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
de	`string` or `null`		string	`null`	Path to the german translation file
fr	`string` or `null`		string	`null`	Path to the french translation string
it	`string` or `null`		string	`null`	Path to the italian translation string

6.28 IFCConfig

IFC export configuration

6.28.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
author	`string`		string		Author of the IFC model
version	`string`		string		IFC schema version
application_name	`string`		string		Name of the application generating IFC
project_name	`string`		string		Project name in IFC
geo_referencing	`string`		GeoReferencing		Georeferencing configuration for IFC
coordinate_reference_system	`object`		CoordinateReferenceSystem		Coordinate reference system for IFC

Property	Type	Required	Possible values	Default	Description
projection_feature_types	`array`		ProjectionFeatureType	`[]`	List of projection feature type definitions
building_feature_types	`array`		BuildingFeatureType	`[]`	List of building feature type definitions
extrusion_feature_types	`array`		ExtrusionFeatureType	`[]`	List of extrusion feature type definitions
groups	`array`		GroupConfig	`[]`	List of group configurations for IFC

6.29 ProjectionAttributeConfig

Attribute mapping configuration for projection feature type

6.29.0.1 Type: `object`

Property	Type	Required	Possible values	Description
attribute	`string`		string	Attribute name (Only applied if the attribute exists on the entity)
source	`object`		ProjectionConfigSource	Source configuration for this attribute

6.30 ProjectionConfigSource

Source configuration for projection feature type

6.30.0.1 Type: `object`

Property	Type	Required	Possible values	Description
type	`string`		ProjectionSource	Type of the data source
expression	`string`		string	Expression defining the source data

6.31 ProjectionEntityConfig

Entity mapping configuration for projection feature type

6.31.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
entity	`string`		string		Name of entity according to IFC schema, e.g. 'IfcWall'
attributes	`array`		ProjectionAttribute-Config	`[]`	List of attribute mappings
properties	`array`		ProjectionProperty-Config	`[]`	List of property mappings

6.32 ProjectionEntityTypeConfig

Entity type mapping configuration for projection feature type

6.32.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
attributes	`array`		ProjectionAttribute-Config	`[]`	List of attribute mappings
properties	`array`		ProjectionProperty-Config	`[]`	List of property mappings

6.33 ProjectionFeatureType

Feature type configuration for projection feature type

6.33.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
name	`string`		string		Feature type name for the projection
sql_path	`string`		string		Path to SQL definition for the projection feature type. Must return at least a column named 'wkt'.
entity_mapping	`object`		ProjectionEntityConfig		Entity mapping configuration for the projection
entity_type_mapping	`object` or `null`		ProjectionEntityTypeConfig	`null`	Entity type mapping configuration for the projection. (Only supported for entities with TypeObject)
spatial_structure_mapping	`object`		ProjectionSpatialEntityConfig		Spatial structure mapping for the projection
group_mapping	`array`		ProjectionConfigSource	`[]`	Group mappings for the projection feature type
color	`object`		Color	`"white"`	Color assigned to the projection feature type

6.34 ProjectionPropertyConfig

Property mapping configuration for projection feature type

6.34.0.1 Type: `object`

Property	Type	Required	Possible values	Description
property	`string`		string	Property name
property_set	`string`		string	Property set name
source	`object`		ProjectionConfigSource	Source configuration for this property

6.35 ProjectionSource

Supported source types for properties and attributes in projection feature types

6.35.0.1 Type: `string`

Possible Values: `STATIC` or `SQL`

6.36 ProjectionSpatialEntityConfig

Spatial structure mapping configuration for projection feature type

6.36.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
attributes	`array`		ProjectionAttribute-Config	`[]`	List of attribute mappings
properties	`array`		ProjectionProperty-Config	`[]`	List of property mappings

6.37 PropertyConfig

Property mapping configuration

6.37.0.1 Type: `object`

Property	Type	Required	Possible values	Description
property	`string`		string	Property name
property_set	`string`		string	Property set name
value	`string`		string	Property value

6.38 RedisConfig

Redis connection configuration

6.38.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
host	`string`		string		Redis host address
port	`integer`		integer		Redis port number
db	`object`		RedisDBConfig		Nested Redis database configuration
global_keyprefix	`string` or `null`		string	`null`	Optional global keyprefix for the Redis result backend
queue	`string` or `null`		string	`null`	Optional queue name for Celery tasks

6.39 RedisDBConfig

Redis databases configuration

6.39.0.1 Type: `object`

Property	Type	Required	Possible values	Description
celery_broker	`integer`		`0 <= x`	DB index for Celery broker
celery_backend	`integer`		`0 <= x`	DB index for Celery result backend
file_cache	`integer`		`0 <= x`	DB index for file cache

6.40 STACConfig

STAC URLs configuration for external data sources

6.40.0.1 Type: `object`

Property	Type	Required	Possible values	Default	Description
dtm_items_url	`string` or `null`		string	`null`	URL to STAC items for DTM data
building_items_url	`string` or `null`		string	`null`	URL to STAC items for building data

6.41 TINConfig

TIN generation configuration

6.41.0.1 Type: `object`

Property	Type	Required	Possible values	Description
grid_size	`number`		GridSize	TIN grid size
max_height_error	`number`		`0.0 <= x <= 0.05`	Maximum allowed height error for TIN generation

---

Markdown generated with jsonschema-markdown.

7 API

This API provides endpoints to generate IFC models based on polygon input, check the generation state, and retrieve the generated file. There is also a swagger documentation site documenting all endpoints: http://0.0.0.0:8000/docs

7.1 Endpoints

7.1.1 `POST /generate-model/`

Description: Starts the generation of a new IFC model.

Request Body (JSON):

- `IFC_VERSION` *(string, required)*: The IFC version (`IFC4`, `IFC4X3_ADD2`).
- `NAME` *(string, required)*: The name of the model.
- `POLYGON` *(string, required)*: A closed polygon in WKT (Well-Known Text) format.
- `PROJECT_ORIGIN` *(string, optional)*: Origin point as a comma-separated string `[x,y,z]`.
- `LANGUAGE` *(string, optional)*: The language of the model (`DE`, `FR`, `IT`)

Responses:

- `200`: Model generation started successfully. Returns task ID.
- `422`: Validation error in the input data.
- `500`: Error.

7.1.2 `GET /generation-state/{task_id}`

Description: Retrieves the current state of a model generation task.

Path Parameter:

- `task_id` *(string, required)*: The ID of the generation task.

Responses:

- `200`: Returns the state of the task.
- `500`: Error.

7.1.3 `GET /generated-file/{task_id}`

Description: Fetches the generated IFC file once the task is completed.

Path Parameter:

- `task_id` *(string, required)*: The ID of the generation task.

Responses:

- `200`: Returns the generated file.
- `202`: Task is still ongoing.
- `400`: Model generation failed.
- `410`: File not found.
- `500`: Error.

References

buildingSmart International, 2023. IFC4.3.2.0 Documentation (official 4.3.2.0) [WWW Document]. URL https://standards.buildingsmart.org/IFC/RELEASE/IFC4_3/index.html (accessed 11.28.2023).

eCH-0031 iliRefMan, 2024. *eCH-0031 INTERLIS 2 – Referenzhandbuch, Version 2.1.0*, eCH Government Standards.

Gröger, G., Kolbe, T.H., Nagel, C., Häfele, K.-H., 2012. *OGC City Geography Markup Language (CityGML) Encoding Standard*. OGC Version 2.0.0, 344.

ISO 16739-1, 2024. *ISO 16739-1:2024 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema.*

ISO 19125-1, 2006. *ISO 19125-1:2006 Geographic information - Simple feature access - Part 1: Common architecture.*

CityGML CM, 2021. *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard.*

Schildknecht, L., 2023. *Leitungskataster nach SIA405 - Analyse zur Nutzung von IFC.* Phase0 - Journal für integriertes Planen, Bauen und Betreiben. https://doi.org/10.21428/71cd88bc.016ca100

Schildknecht, L., Schneider, O., Meyer, J., Gamma, C., Gschwind, J., 2025b. *Integration von Geodaten in BIM/IFC - ein Open-Source-Ansatz für die Daten der amtlichen Vermessung.* Phase0 - Journal für integriertes Planen, Bauen und Betreiben. https://doi.org/10.21428/71cd88bc.1463dde3

Schildknecht, L., Schneider, O., Meyer, J., Gamma, C., Gschwind, J., 2025a. *Integration of land administration data into BIM/IFC - an open source approach for Swiss cadastral survey data*, Dreiländertagung der DGPF, der OVG und der SGPF; Band 33.

SIA 4008, 2025. *SIA 4008:2025 Leitungskataster - Wegleitung zur Norm SIA 405.*

SIA 405, 2025. *SIA 405:2025 Geodaten zu Ver- und Entsorgungsleitungen.*