Rules for the use of CityObjectRelation and XLink for linking geometries / features that are shared by multiple (top-level) features

1. Terms and Definitions

feature
abstraction of real world phenomena
NOTE Features are objects that have an identity that allows for distinguishing features from each other. Features can have spatial and non-spatial properties that describe the features in more detail. Spatial properties are properties that have a geometry from ISO 19107 as data type.
NOTE Features are denoted by the stereotype «FeatureType» in the CityGML 3.0 Conceptual Model.
[ISO 19101- 1:2014, 4.1.11]

geometry
An ordered set of n-dimensional points in a given coordinate reference system; can be used to model the spatial extent or shape of a feature [OGC 15-107, modified]
NOTE Geometries are objects that can have an identity that allows for distinguishing geometries from each other. Geometries define the data types of spatial properties of features.

top-level feature
a feature that represents one of the main components of 3D city models; can be further semantically and spatially decomposed and substructured into parts
NOTE Top-level features are denoted by the stereotype «TopLevelFeatureType» in the CityGML 3.0 Conceptual Model.
[CityGML Conceptual Model]

space
an entity of volumetric extent in the real world
NOTE Spaces can be spatially represented in different LODs and by different types of geometric objects.
NOTE Spaces are represented in the CityGML 3.0 Conceptual Model by those classes that are derived from the class AbstractSpace.
[CityGML Conceptual Model]

space boundary
an entity with areal extent in the real world. Space boundaries are objects that bound a Space. They also realize the contact between adjacent spaces.
NOTE Space boundaries can be spatially represented in different LODs and by different types of geometric objects.
NOTE Space boundaries are represented in the CityGML 3.0 Conceptual Model by those classes that are derived from the class AbstractSpaceBoundary.
[CityGML Conceptual Model]

city-object relation
a specific relation from the city object in which the relation is included to another city object
[CityGML Conceptual Model]

2. Object identifiers for features and geometries

In the GML encoding of the CityGML 3.0 Conceptual Model, different types of objects exist: *features*, *top-level features*, and *geometries*. All these objects have a unique identity, thus, they can be distinguished from each other and can also be referenced from other objects.

The CityGML 3.0 Conceptual Model introduces two attributes to provide all features and top-level features with a unique identity: 1) the mandatory *featureID* attribute to distinguish all (top-level) features and possible multiple versions of the same real-world object and 2) an optional *identifier* attribute to reference specific (top-level) features independent from their actual feature version. The featureID attribute value is unique within the same CityGML dataset and also for a specific version of a feature, whereas the identifier attribute has an identical value for all versions of the same real-world object. It is recommended to use globally unique identifiers like UUID values or identifiers maintained by an organization such as a mapping agency for both attributes: for the identifier attribute, as these identifiers should remain stable over the lifetime of the real-world object, and for the featureID attribute, as this ensures that 3D city models that are integrated from different sources into e.g. one GIS or database will not have colliding featureID values and also to be able to uniquely identify a specific version of a feature using these identifiers together with a timestamp and a version number.

The two attributes are defined in the UML class *AbstractFeature* as part of the *Core* module of the CityGML 3.0 Conceptual Model. The class *AbstractFeature* was introduced to the UML model to be able to define attributes at the conceptual level that are automatically inherited from the class *AbstractGML* in the GML encoding, but not in other encodings. In this way it will be ensured that these attributes are available in all possible CityGML 3.0 encodings. Since these two attributes originate from the GML specification, they are mapped onto the predefined GML concepts *gml:id* and *gml:identifier* in the GML encoding.

The table lists the two identity attributes together with their definition and GML encoding:

UML attributes to provide (top-level) features with identity	**Mapping to corresponding GML concepts**	**Definition**
featureID	XML attribute gml:id	Specifies the unique identifier of the feature that is valid in the instance document within which it occurs.
identifier	XML element gml:identifier	Specifies the unique identifier of the feature that is valid globally.

The GML concepts are used as illustrated below to represent the object identity in GML

```
<bldg:Building gml:id="B1020_t1">
  <gml:identifier codeSpace="www.test.org/xyz">B1020</gml:identifier>
  <core:creationDate>2012-08-02T00:00:00</core:creationDate>
  <core:terminationDate>2013-10-09T00:00:00</core:terminationDate>
  <bldg:function>Office</bldg:function>
</bldg:Building>
```

The GML encoding implements the geometry model defined in ISO 19107. This means that all geometries are derived from the class AbstractGML as well and, thus, also geometries exhibit the same unique identity as features do. The fact that geometries also have identifiers is of particular importance for CityGML, since CityGML requires defining references to geometries. One example is the application of textures to surfaces. The textures reference the gml:ids of those LinearRings that describe the boundaries of the surface geometries (e.g. Triangle, Polygon, or a MultiSurface consisting of Polygons) to which the textures are applied. Another example is that the CityGML GML encoding allows for realising topological relationships amongst others through referencing of shared geometries.

3. Referencing features and geometries

There are two XML concepts that are used to reference objects in GML instance documents: the XML Linking Language (XLink) and the XML Pointer Language (XPointer).

XLink allows for creating links to other documents by providing specific attributes that are added to the element that references the document. The most commonly used attribute is *xlink:href*, the value of the attribute is the URI of the referenced document.

In CityGML, however, we are mainly interested in referencing specific objects (features and geometries) within the same or external GML instance document. For this, XPointer is used. To reference objects by their gml:id, a number sign (#) followed by the gml:id of the object to be referenced is added to the URI of the referenced document. In the example below, the element <versionMember> references a <Building> feature. Since the feature is part of the same GML instance document, a relative URI is specified by simply providing the number sign and the *gml:id* of the building as value of the attribute *xlink:href*. In the context of version management, referencing features by their gml:id means that a specific feature version of a real-world object is referenced.

```
<core:CityModel gml:id="CM_1">
  <core:versionMember>
    <vers:Version gml:id="V_1">
      <vers:versionMember xlink:href="#BU_234"/>
    </vers:Version>
  </core:versionMember>
  <core:cityObjectMember>
    <bldg:Building gml:id="BU_234"> ... </bldg:Building>
  </core:cityObjectMember>
<core:CityModel>
```

For referencing the actual real-world object itself, a reference to the <gml:identifier> element of the feature versions needs to be provided. However, this kind of reference can be ambiguous, since all feature versions of a specific real-world object hold the same <gml:identifier> value; thus, a reference to a <gml:identifier> value may refer to multiple features. A possible disambiguation can be performed when selecting only that feature version that is valid for a given time point (and optionally a specific workspace). In the example below, the XPointer references the real-world feature BuildingPart with the ID “BP_12” by referencing all versions of the BuildingPart feature with the corresponding <gml:identifier> value “BP_12”. For selecting a specific BuildingPart version only, the time points <core:creationDate> and <core:terminationDate> need to be evaluated in addition.

```xml
<core:CityModel gml:id="CM_2"
  <core:cityObjectMember>
    <bldg:Building gml:id="B1020_t1">
      <gml:identifier codeSpace="www.test.org/xyz">B1020</gml:identifier>
      <core:creationDate>2012-08-02T00:00:00</core:creationDate>
      <core:terminationDate>2013-10-09T00:00:00</core:terminationDate>
      <bldg:function>Office</bldg:function>
      <bldg:buildingPart>
        <bldg:BuildingPart gml:id="BP12_t1">
          <gml:identifier codeSpace="www.test.org/xyz">BP12</gml:identifier>
          <core:creationDate>2012-08-02T00:00:00</core:creationDate>
          <core:terminationDate>2014-06-03T00:00:00</core:terminationDate>
          <bldg:roofType>Flat</bldg:roofType>
        </bldg:BuildingPart>
      </bldg:buildingPart>
    </bldg:Building>
  </core:cityObjectMember>
  <core:cityObjectMember>
    <bldg:Building gml:id="B1020_t2">
      <gml:identifier codeSpace="www.test.org/xyz">B1020</gml:identifier>
      <core:creationDate>2013-10-09T00:00:00</core:creationDate>
      <bldg:function>Living</bldg:function>
      <bldg:buildingPart
          xlink:href="#xmlns(bldg=http://www.opengis.net/citygml/building/3.0)
                      xmlns(gml=http://www.opengis.net/gml/3.2)
                      xpointer(//bldg:BuildingPart[gml:identifier[text()='BP12']])"/>
    </bldg:Building>
  </core:cityObjectMember>
</core:CityModel>
```

When using references, it is important to keep in mind that many (top-level) features in CityGML have a complex structure. In general, features can have spatial and non-spatial properties that describe the features in more detail. Spatial properties are properties that have a geometry from ISO 19107 as data type (<lod2Solid> in the example). Non-spatial properties can either have a simple data type (<function> with Integer value or <creationDate> with Date value), but they can also be further specified by additional attributes, such as the height of the building that is described by further properties (<height> with the additional properties <highReference>, <lowReference>, <status>, and <value>). In addition, they can also be composed of subfeatures that contain spatial and non-spatial properties themselves (<WallSurface> with <lod2MultiSurface> geometry).

The subfeatures can be provided *inline* or *by reference*. Inline means that the subfeatures are provided directly as content of the (top-level) feature as is shown below, where the subfeature <BuildingRoom> is provided as content of the <Building> feature. In contrast, by reference means that the subfeatures are provided elsewhere in the CityGML document and are referenced from the (top-level) feature using an XLink as is described above and illustrated in Figure x.

```
<bldg:Building gml:id="BU_234">
  <core:creationDate>2019-09-24T00:00:00</core:creationDate>
  <core:boundary>
    <con:WallSurface gml:id="WS_21">
      <core:lod2MultiSurface>
        <gml:MultiSurface gml:id="MS_21_1">
          ...
        </gml:MultiSurface>
      </core:lod2MultiSurface>
    </con:WallSurface>
  </core:boundary>
  <core:lod2Solid>
    <gml:Solid gml:id="S_1"> ... </gml:Solid>
  </core:lod2Solid>
  <con:height>
    <con:Height>
      <con:highReference>topOfConstruction</con:highReference>
      <con:lowReference>lowestGroundPoint</con:lowReference>
      <con:status>measured</con:status>
      <con:value uom="urn:adv:uom:m">24.709</con:value>
    </con:Height>
  </con:height>
  <bldg:function>1000</bldg:function>
  <bldg:buildingRoom>
    <bldg:BuildingRoom gml:id="BR_3">
      ...
    </bldg:BuildingRoom>
  <bldg:buildingRoom>
</bldg:Building>
```

4. Rules for linking features and geometries that are shared by multiple (top-level) features

When modelling cities, geometries and features can be integral parts of multiple city objects. To avoid redundant modelling of these geometries and features, CityGML offers the possibility to represent geometries and features only once and to reference them from any other city object to which they belong as well.

This non-redundant representation guarantees that no integrity problems occur, i.e. several differing instances of the same geometry or feature will not exist.

Three different cases for non-redundant representation need to be differentiated:

1. Geometries are represented in different parts within the same top-level feature. An example is the roof surface of a building. The polygon representing the geometry of the RoofSurface feature is at the same time part of the RoofSurface feature and of the Solid geometry of the Building feature.

2. One geometry can be part of the representation of different features. An example is a road across a bridge, the road surface sharing the geometry with the roof surface of the bridge.

3. One and the same feature can belong to different aggregations. Examples are an intersection that belongs to two roads, the intersection being one and the same feature for both roads, or features that belong to a CityObjectGroup and that are already integral part of the city model.

For these cases different recommendations are provided for how to encode the references in CityGML.

Although these recommendations impose restrictions, they facilitate at the same time reading, storing, processing, and generating of CityGML documents, because they reduce the multiple possibilities of how to represent and link features and geometries in CityGML documents to the most appropriate ones. Furthermore, top-level features can now completely be loaded in the main memory, because links to shared geometries that are part of different top-level features represented further down in the GML document do not need to be resolved any more. This also facilitates querying features and geometries using web services, as up to now queries cannot address specific parts of a geometry.

4.1. Referencing geometries using XLinks within the same and from different top-level features

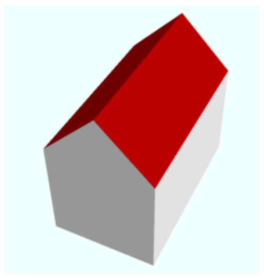
1. XLinks may be used to reference geometries within the same top-level feature in accordance with rules 4.2.1-4.2.5.
2. XLinks shall not be used to reference geometries from another top-level feature.

4.2. Referencing geometries of spaces and space boundaries

1. [The geometry describing a space shall be stored with the space or its space boundaries.]
2. Geometries stored inline a space boundary must be referenced from the geometry of the space using XLinks.
3. Space boundaries shall not reference geometries of the space using XLinks.
4. The geometry of a space may contain the geometries of nested spaces.
5. LoDs must be self-contained: Geometries shall not be shared between different LoDs using XLinks.

Here, XLink represents a link at the geometry level (“geometry link”), i.e. a reference to the ID of the geometry to be reused. The link direction is always from the geometry of the space to the geometries of the space boundaries (example 1).
If the space is not bounded by space boundaries (e.g. WallSurface or RoofSurface), then the geometry is stored as a geometry property (e.g. lod2MultiSurface) of the space. No XLinks are required in this case.

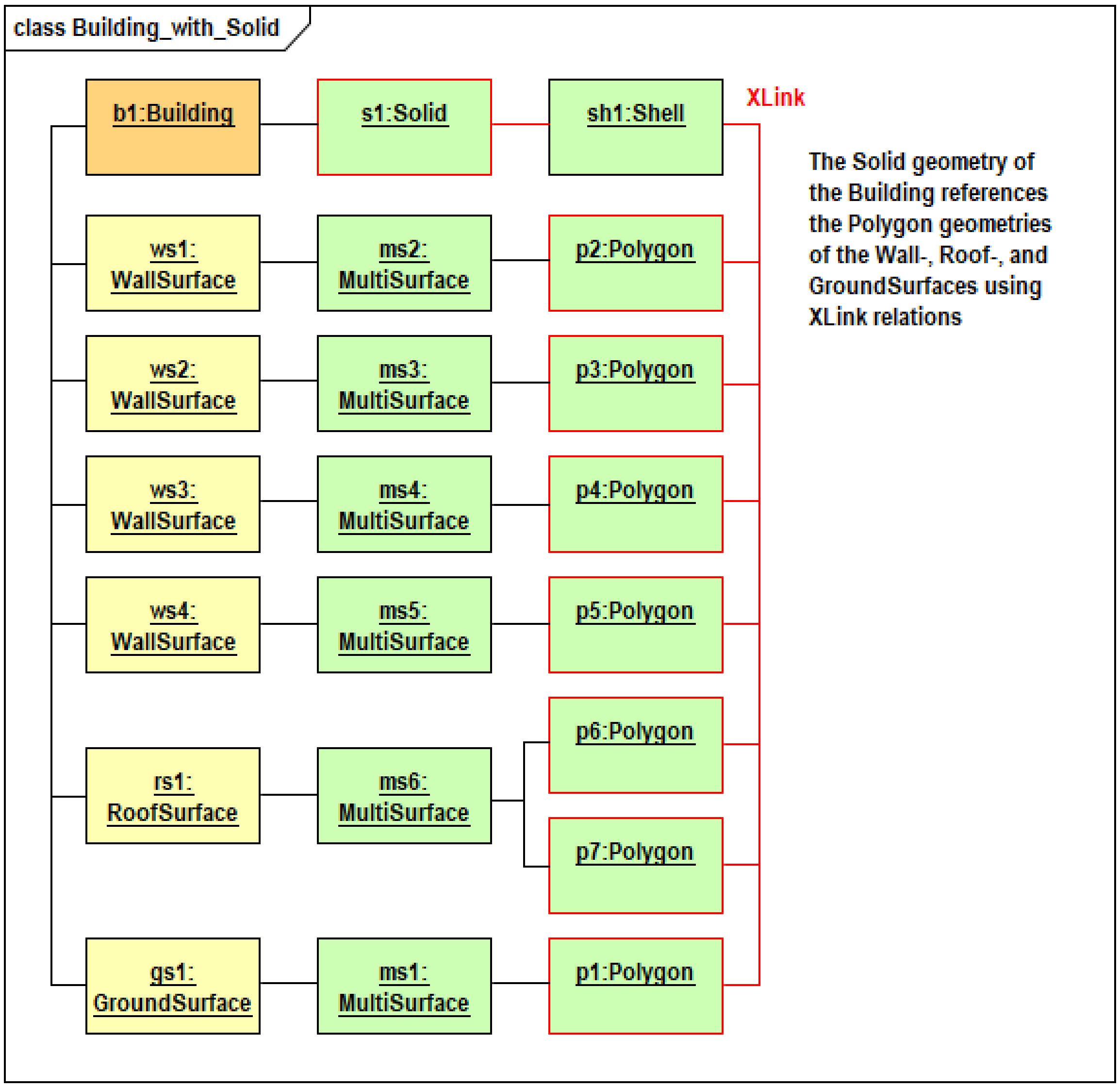
Example 1: Building with Solid geometry and boundary surfaces



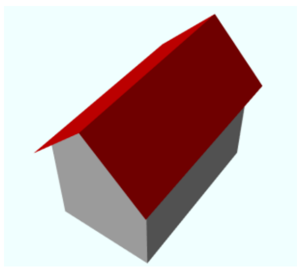
The building (=space) is modelled in LOD2 as Solid geometry and is bounded by four WallSurfaces, one RoofSurface, and one GroundSurface (=space boundaries). All space boundaries are modelled as Polygon geometries. The Solid geometry of the building references the Polygon geometries using XLink.

The GML file is available here:
https://github.com/opengeospatial/CityGML-3.0Encodings/tree/xlinks-discussion/CityGML/Examples/Building/XLink_examples/1_SimpleBuilding

The Building from the GML file is illustrated in the object diagram below. The XLink relations between the Solid geometry and the Polygon geometries are highlighted in red.



Example 2: Building with roof overhangs

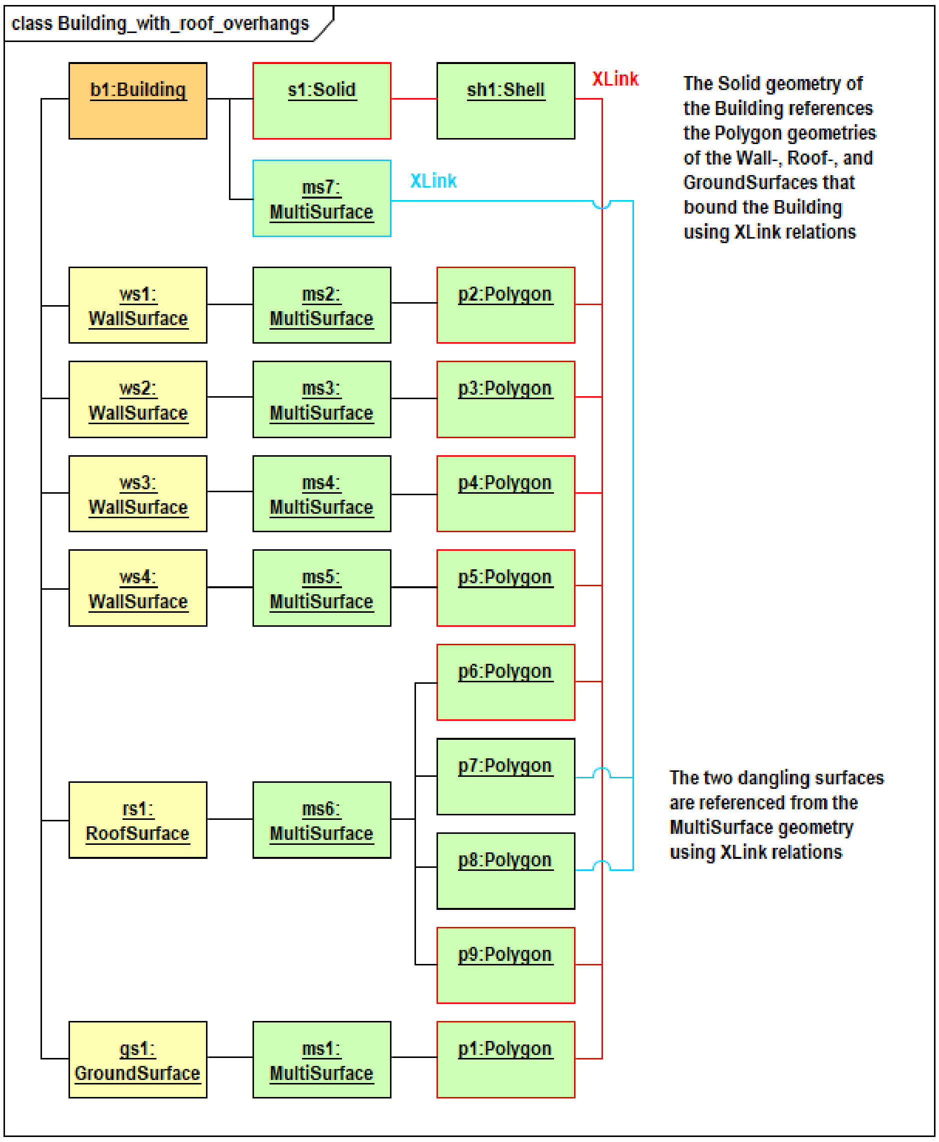


The building (=space) is modelled in LOD2 as Solid geometry and is bounded by four WallSurfaces, one RoofSurface, and one GroundSurface (=space boundaries). All space boundaries are modelled as Polygon geometries. The Solid geometry of the building references the Polygon geometries using XLink.

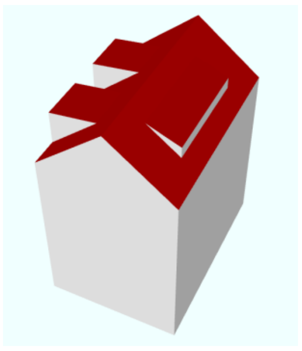
The RoofSurface contains four Polygon geometries. Two of these Polygons are roof overhangs (i.e. dangling surfaces), and, thus, are not referenced by the Solid geometry of the building, as they would render the solid invalid if referenced. For this reason, an additional MultiSurface geometry is added to the building that references the dangling surfaces.

The GML file is available here:
https://github.com/opengeospatial/CityGML-3.0Encodings/tree/xlinks-discussion/CityGML/Examples/Building/XLink_examples/2_SimpleBuilding_Roof_Overhangs

The Building from the GML file is illustrated in the object diagram below. The XLink relations between the Solid geometry and the Polygon geometries are highlighted in red, the XLink relations between the MultiSurface geometry and the dangling surfaces in blue.



Example 3: Building with BuildingInstallation

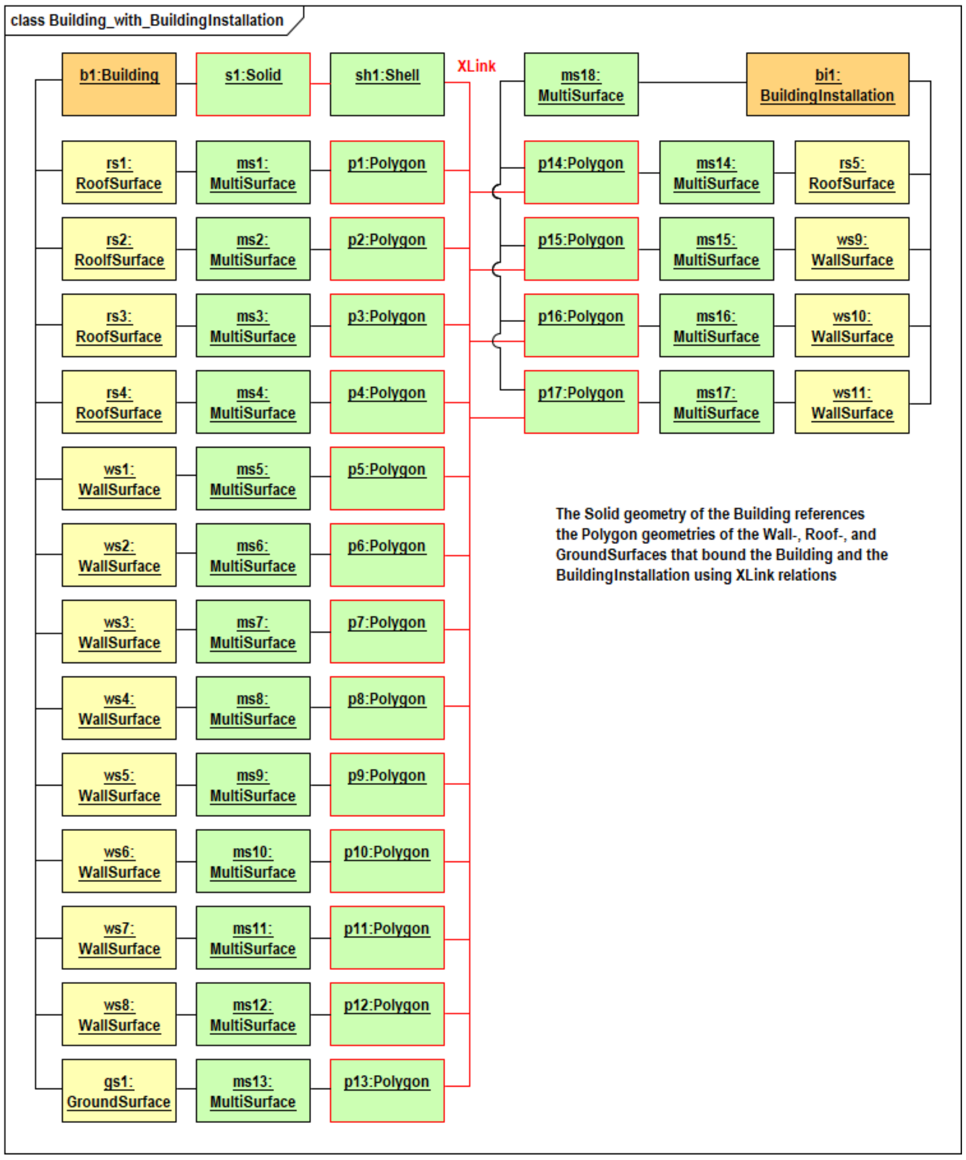


The building (=space) is modelled in LOD2 as Solid geometry and is bounded by eight WallSurfaces, four RoofSurfaces, and one GroundSurface (=space boundaries). In addition, the building has a dormer that is modelled as BuildingInstallation (=space). The building installation is modelled as MultiSurface geometry and is bounded by one RoofSurface and three WallSurfaces (=space boundaries).

The space boundaries of the building and of the building installation are all modelled as Polygon geometries. The Solid geometry of the building references those Polygon geometries that represent the space boundaries of the building space using XLink. The MultiSurface geometry of the building installation references those Polygon geometries that represent the space boundaries of the building installation using XLink. In addition, the Solid geometry may also reference the Polygon geometries that represent the space boundaries of the building installation using XLink.

The GML file is available here:
https://github.com/opengeospatial/CityGML-3.0Encodings/tree/xlinks-discussion/CityGML/Examples/Building/XLink_examples/3_Building_With_Nested_Features

The Building from the GML file is illustrated in the object diagram below. The XLink relations are highlighted in red.



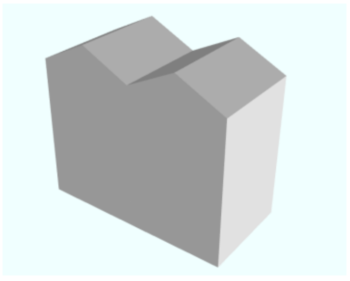
class Building_with_BuildingInstallation
b1:Building
s1:Solid
sh1:Shell
XLink
ms18: MultiSurface
bi1: BuildingInstallation
rs1: RoofSurface
ms1: MultiSurface
p1:Polygon
rs2: RoolfSurface
ms2: MultiSurface
p2:Polygon
rs3: RoofSurface
ms3: MultiSurface
p3:Polygon
rs4: RoofSurface
ms4: MultiSurface
p4:Polygon
ws1: WallSurface
ms5: MultiSurface
p5:Polygon
ws2: WallSurface
ms6: MultiSurface
p6:Polygon
ws3: WallSurface
ms7: MultiSurface
p7:Polygon
ws4: WallSurface
ms8: MultiSurface
p8:Polygon
ws5: WallSurface
ms9: MultiSurface
p9:Polygon
ws6: WallSurface
ms10: MultiSurface
p10:Polygon
ws7: WallSurface
ms11: MultiSurface
p11:Polygon
ws8: WallSurface
ms12: MultiSurface
p12:Polygon
gs1: GroundSurface
ms13: MultiSurface
p13:Polygon
p14:Polygon
ms14: MultiSurface
rs5: RoofSurface
p15:Polygon
ms15: MultiSurface
ws9: WallSurface
p16:Polygon
ms16: MultiSurface
ws10: WallSurface
p17:Polygon
ms17: MultiSurface
ws11: WallSurface
The Solid geometry of the Building references the Polygon geometries of the Wall-, Roof-, and GroundSurfaces that bound the Building and the BuildingInstallation using XLink relations

4.3. Expressing shared geometries between top-level features using CityObjectRelations

1. If two top-level features share a common geometry, the shared geometry must be stored for each top-level feature separately (follows from 4.1.2).
2. A CityObjectRelation may be modelled for the features where the shared geometries are stored (might be the top-level feature itself or one of its nested features).
3. Each CityObjectRelation must be assigned the relation type “shared”.
4. Each CityObjectRelation must reference the other feature using an XLink. Thus, the reference shall be bi-directional.

CityObjectRelation represents a link at the feature level (“feature link”) referencing the ID of another feature that contains a shared geometry.

Example 1: Two buildings with shared boundary surface

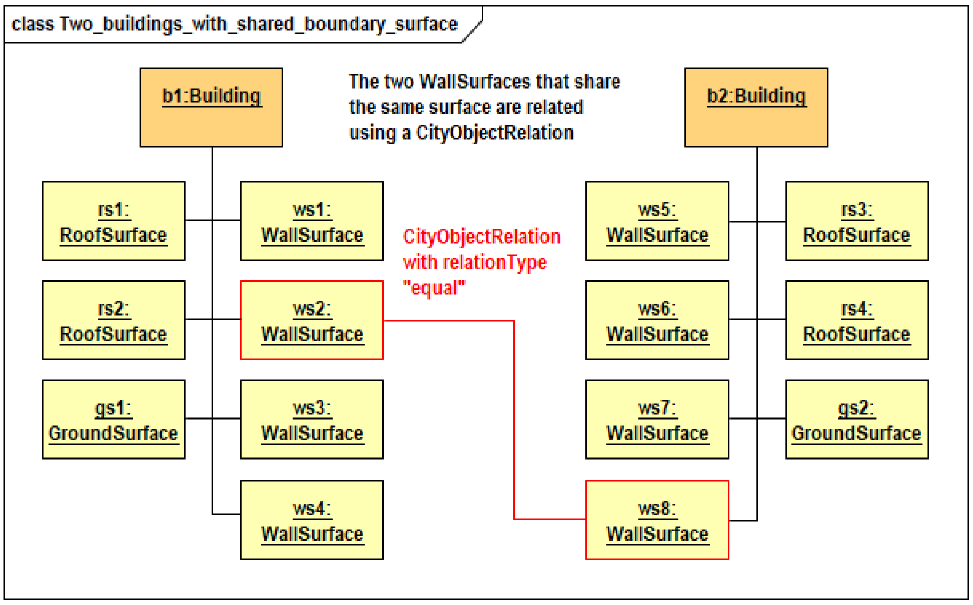


Two buildings (=top-level features) are modelled in LOD2 as Solid geometry and are bounded by Wall-, Roof-, and GroundSurfaces that are modelled as Polygon geometries. One of the WallSurfaces of the first Building shares the Polygon geometry with one of the WallSurfaces of the second Building. Both WallSurfaces might appear identical, however, the surface normals of the Polygon geometries of the WallSurfaces are pointing in opposite directions.

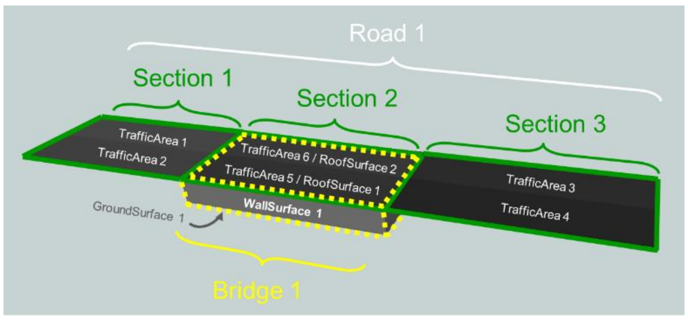
To express that the WallSurfaces of the two buildings share the Polygon geometry, the WallSurfaces reference each other using a CityObjectRelation with the relation type "shared". Both WallSurfaces contain the Polygon geometry themselves, the second WallSurface, however, in reverse order.

The GML file is available here:
https://github.com/opengeospatial/CityGML-3.0Encodings/tree/xlinks-discussion/CityGML/Examples/Building/XLink_examples/4_Cross-Top-Level-XLink

The Buildings from the GML file are illustrated in the object diagram below. The CityObjectRelation is highlighted in red.



Example 2: Road crossing a Bridge

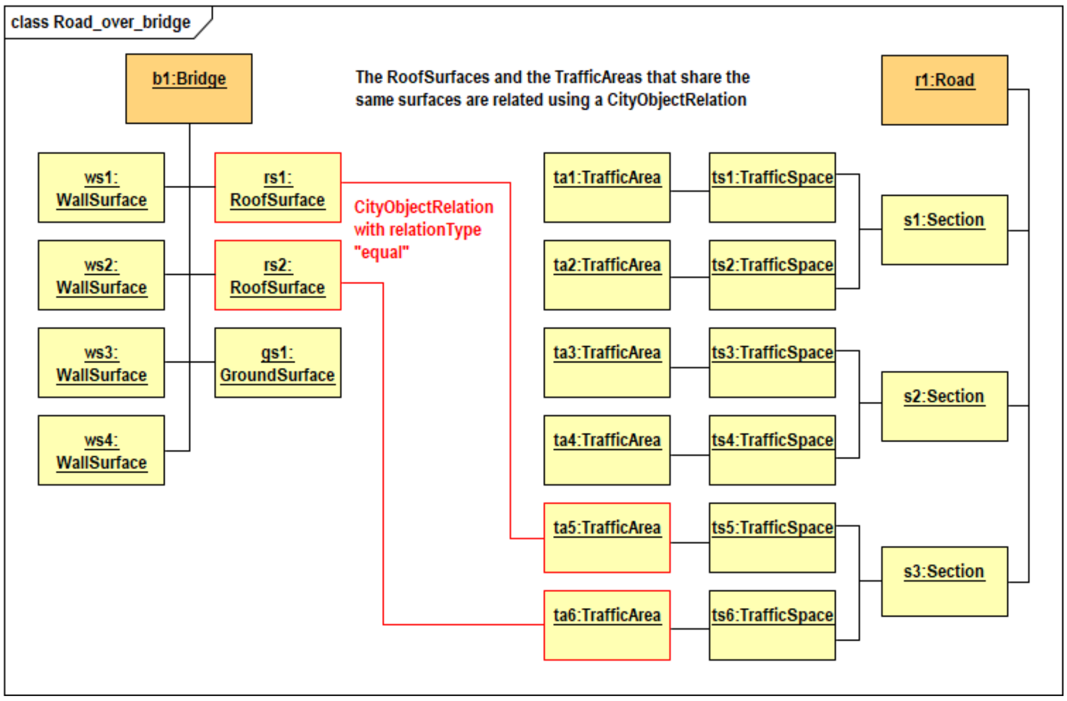


A Road and a Bridge (=top-level features) are modelled in LOD2. The Bridge is bounded by Ground-, Roof-, and WallSurfaces that are modelled as MultiSurface geometries. The Road consists of three sections; each section is bounded by two TrafficAreas that are modelled as MultiSurface geometries as well. The RoofSurfaces of the Bridge share MultiSurface geometries with two TrafficAreas of the Road. The RoofSurfaces and the TrafficAreas are geometrically identical, but they differ semantically.

To express that the RoofSurfaces share MultiSurface geometries with two TrafficAreas, they reference each other using CityObjectRelations with the relation type "shared".

The GML file is available here: https://github.com/opengeospatial/CityGML-3.0Encodings/blob/master/CityGML/Examples/Transportation/Basic%20examples/Road_over_Bridge_CityGML3.0_LOD2_with_CityObjectRelations.gml

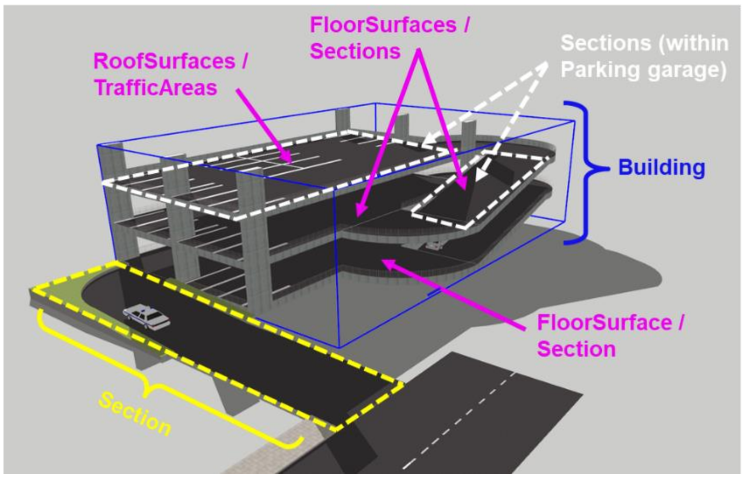
The Road and Bridge from the GML file are illustrated in the object diagram below. The CityObjectRelations are highlighted in red.



Example 3: Parking garage

A parking garage is modelled in LOD2 as a building (=top-level feature) with Floor-, Roof-, and WallSurfaces that are modelled as MultiSurface geometries. The parking garage contains a Road with Sections and TrafficAreas that are modelled as MultiSurface geometries as well. The Floor- and RoofSurface of the Building share MultiSurface geometries with the Sections and TrafficAreas of the Road.

To express the sharing of MultiSurface geometries between the Roof-/WallSurfaces and the Sections/TrafficAreas, they reference each other using CityObjectRelations with the relation type "shared".



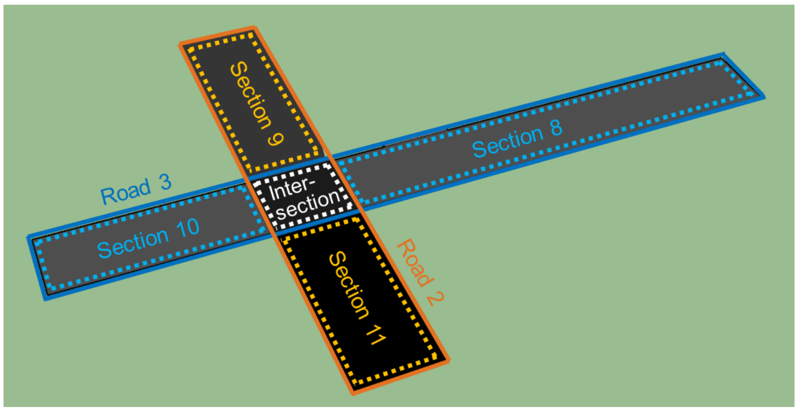
The GML file is available here: https://github.com/opengeospatial/CityGML-3.0Encodings/blob/master/CityGML/Examples/Transportation/Basic%20examples/Road_over_Bridge_CityGML3.0_LOD2_with_CityObjectRelations.gml

4.4. Referencing features from alternative aggregations

1. Each feature belongs to a natural aggregation hierarchy and shall be stored inline in this hierarchy.
2. Alternative aggregations may not contain the feature inline but must use an XLink to reference the feature.

Here, XLink represents a link at the feature level (“feature link”), i.e. a reference to the ID of the feature being part of the natural aggregation. All features are part of a natural aggregation, i.e. features are typically represented in a data set once in physical form, either directly as part of the city model when they are top-level features (e.g. a Building), or inline as part of other (top-level) features (e.g. a BuildingRoom represented inline as part of the top-level feature Building). At the same time, the features can also occur in alternative aggregations.

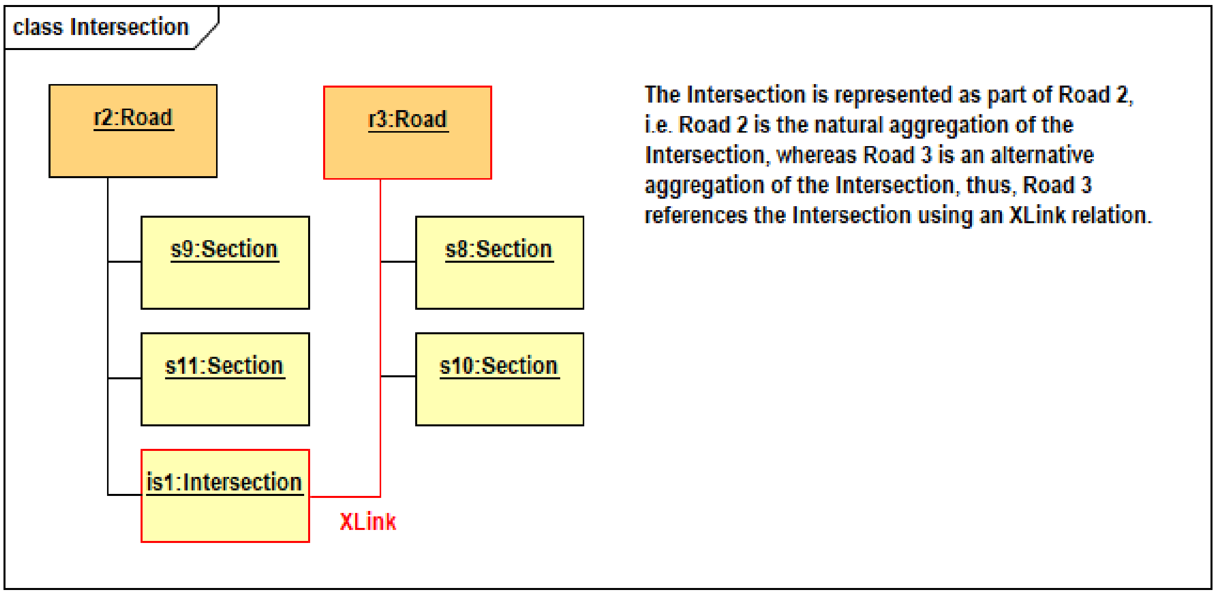
Example 1: Intersection as part of two Roads



Given are two roads that each have two Sections and one Intersection. The two roads cross each other at the Intersection. Although the Intersection is shared by the Roads, it exists in reality only once. This is reflected by specifying that the natural aggregation of the Intersection feature is Road 2, whereas Road 3 represents an alternative aggregation of the Intersection. Thus, the Intersection feature is represented inline as part of Road 2, whereas it is referenced by Road 3 using an XLink that references the ID of the Intersection feature.

The GML file is available here: https://github.com/opengeospatial/CityGML-3.0Encodings/blob/master/CityGML/Examples/Transportation/Basic%20examples/ParkingGarage_CityGML3.0_LOD2_with_CityObjectRelations_and_XLinks.gml

The two Roads and the Intersection from the GML file are illustrated in the object diagram below. The XLink relation is highlighted in red.


class Intersection
r2:Road
r3:Road
s9:Section
s8:Section
s11:Section
s10:Section
is1:Intersection
XLink
The Intersection is represented as part of Road 2, i.e. Road 2 is the natural aggregation of the Intersection, whereas Road 3 is an alternative aggregation of the Intersection, thus, Road 3 references the Intersection using an XLink relation.

Example 2: A specific version of a city model

A Version features groups, for instance, versions of city objects that are valid within a specific time period. The city model represents the natural aggregation of these versioned city objects, whereas the Version feature represents the alternative aggregation. Thus, the versioned city objects are represented inline as part of the city model, whereas they are referenced by the Version feature using XLink relations.

Example 3: Building rooms belonging to a Storey

BuildingRooms are usually represented inline as part of the Building they belong to, thus, the Building represents the natural aggregation. In addition, Storeys can group BuildingRooms to indicate which BuildingRoom belongs to which Storey. This grouping represents an alternative aggregation, thus, the Storeys must reference the BuildingRooms using XLink relations.

Example 4: A Building installation spanning across several Building Parts

This example was discussed in issue https://github.com/opengeospatial/CityGML-3.0CM/issues/8: “Installations that are spanning across several building parts are to be physically modelled as part of one building part, all other building parts reference the installation using XLinks, expressing in this way, that the installation does not exclusively belong to one building part only.”

This means, that one of the BuildingParts represents the natural aggregation of the BuildingInstallation (i.e. inline representation), whereas the other BuildingParts represent alternative aggregations (i.e. XLink reference).

Example 5: CityObjectGroups

A CityObjectGroup groups existing city objects that are usually represented inline somewhere else in the data set. Thus, CityObjectGroups represent alternative aggregations and have to use XLink to reference the city objects they are grouping.

Please note:

- In case 4.2) XLink represents a link at the geometry level (“geometry link”), i.e. a reference to the ID of the geometry to be reused.
- In case 4.3) and 4.4), CityObjectRelation and XLink represent links at the feature level (“feature links”), i.e. a reference to the ID of the feature.
- Please note: “top-level” is put in brackets, as it is not always top-level features that share a surface / object.

Advantages:

- The explicit representation of the relation between city objects facilitates spatial analyses.
- For visualisation it would be enough to render only one of the two surfaces (if the viewer is capable of doing so).
- The features can be rendered independently without having to take care of geometry links that might not easily be resolvable.
- Maintenance becomes easier. For instance, when one of the features changes and also the geometry of that feature changes, or when the feature does not exist anymore, it's difficult to maintain/update XLinks between the feature geometries.