Contents

i. Preface ..................................................................................................................... ix
ii. Submitting organizations ....................................................................................... ix
iii. Submission contact points ..................................................................................... ix
iv. Participants in development .................................................................................. x
v. Revision history ....................................................................................................... x
vi. Changes to the OGC® Abstract Specification ...................................................... xi
vii. Acknowledgments .................................................................................................... xi

Foreword ....................................................................................................................... xiii

0 Introduction .............................................................................................................. xv
  0.1 Motivation ............................................................................................................. xv
  0.2 Historical background ......................................................................................... xv

1 Scope ......................................................................................................................... 1

2 Conformance .............................................................................................................. 2

3 Normative references ............................................................................................... 2

4 Conventions ................................................................................................................ 3
  4.1 Abbreviated terms ............................................................................................... 3
  4.2 UML Notation ..................................................................................................... 4
  4.3 XML namespaces and namespace prefixes ......................................................... 5
  4.4 XML-Schema ..................................................................................................... 6

5 Overview of CityGML ............................................................................................... 7

6 General characteristics of CityGML ......................................................................... 9
  6.1 Modularisation .................................................................................................... 9
  6.2 Multi-scale modelling (5 levels of detail, LOD) .................................................. 9
  6.3 Coherent semantical-geometrical modelling ..................................................... 10
  6.4 Closure surfaces ............................................................................................... 10
  6.5 Terrain Intersection Curve (TIC) ....................................................................... 11
  6.6 Dictionaries and external code lists for enumerative attributes ......................... 12
  6.7 External references ............................................................................................ 12
  6.8 City object groups ............................................................................................. 12
  6.9 Appearances ...................................................................................................... 13
  6.10 Prototypic objects / scene graph concepts ....................................................... 13
  6.11 Generic city objects and attributes .................................................................. 13
  6.12 Application Domain Extensions (ADE) ........................................................... 14

7 Modularisation ........................................................................................................... 15
8 Spatial model ................................................................. 23
  8.1 Geometric-topological model ........................................ 23
  8.2 Implicit geometries, prototypic objects, scene graph concepts .... 26
    8.2.1 External code lists ............................................. 27
    8.2.2 Example CityGML datasets .................................... 27
    8.2.3 Conformance requirements ..................................... 28

9 Appearance model ........................................................ 29
  9.1 Relation between appearances, features and geometry .......... 30
  9.2 Appearance and SurfaceData ......................................... 31
  9.3 Material .................................................................. 32
  9.4 Texture and texture mapping ......................................... 32
  9.5 Related concepts ...................................................... 38
  9.6 Conformance requirements ........................................... 39
  9.7 Material model of previous CityGML versions [deprecated] .... 40
    9.7.1 Textured surfaces ............................................... 41
    9.7.2 Conformance requirements ...................................... 42

10 Thematic model ............................................................. 43
  10.1 CityGML Core .......................................................... 44
    10.1.1 Base elements ................................................... 46
    10.1.2 Generalisation relation ........................................ 47
    10.1.3 External references ............................................. 47
    10.1.4 Address information ............................................. 48
    10.1.5 External code lists ............................................. 49
    10.1.6 Conformance requirements ..................................... 50
  10.2 Digital Terrain Model (DTM) ......................................... 51
    10.2.1 Relief feature and relief component ......................... 52
    10.2.2 TIN relief .......................................................... 53
    10.2.3 Raster relief ..................................................... 54
    10.2.4 Mass point relief ............................................... 54
    10.2.5 Breakline relief ............................................... 54
    10.2.6 Conformance requirements ..................................... 55
  10.3 Building model ........................................................ 56
    10.3.1 Building and building part ................................... 58
    10.3.2 Boundary surfaces ............................................. 62
    10.3.3 Outer building installations .................................... 65
    10.3.4 Openings .......................................................... 65
    10.3.5 Building interior ............................................... 66
    10.3.6 Modelling building storeys using CityObjectGroups ....... 68
    10.3.7 External code lists ............................................. 68

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
10.3.8 Conformance requirements ................................................................. 69

10.4 Water bodies ......................................................................................... 72
10.4.1 Water body ....................................................................................... 74
10.4.2 Boundary surfaces ........................................................................... 74
10.4.3 External code lists ............................................................................ 76
10.4.4 Conformance requirements ............................................................... 76

10.5 Transportation objects ........................................................................ 77
10.5.1 Transportation complex .................................................................. 80
10.5.2 Subclasses of transportation complexes ......................................... 81
10.5.3 Subdivisions of transportation complexes ....................................... 82
10.5.4 External code lists ........................................................................... 83
10.5.5 Conformance requirements ............................................................... 83

10.6 Vegetation objects ................................................................................ 85
10.6.1 Vegetation object ............................................................................. 87
10.6.2 Solitary vegetation objects ................................................................. 87
10.6.3 Plant cover objects .......................................................................... 88
10.6.4 External code lists ........................................................................... 88
10.6.5 Example CityGML dataset ............................................................... 88
10.6.6 Conformance requirements ............................................................... 89

10.7 City furniture ......................................................................................... 90
10.7.1 City furniture object ....................................................................... 91
10.7.2 External code lists ........................................................................... 92
10.7.3 Example CityGML dataset ............................................................... 92
10.7.4 Conformance requirements ............................................................... 93

10.8 Land use ............................................................................................... 94
10.8.1 Land use object ................................................................................ 95
10.8.2 External code lists ........................................................................... 95
10.8.3 Conformance requirements ............................................................... 95

10.9 City object groups ................................................................................ 96
10.9.1 City object group ............................................................................ 96
10.9.2 Conformance requirements ............................................................... 97

10.10 Generic city objects and attributes ..................................................... 98
10.10.1 Generic city object ......................................................................... 99
10.10.2 Generic attributes .......................................................................... 99
10.10.3 Conformance requirements ............................................................. 100

10.11 Application Domain Extensions (ADE) ............................................. 102
10.11.1 Technical principle of ADEs .......................................................... 102
10.11.2 Example ADE ................................................................................. 103

10.12 Definition of code lists ....................................................................... 107

Annex A (normative) XML Schema definition ........................................ 109
A.1 CityGML Core module ......................................................................... 109
A.2 Appearance module ............................................................................ 113
A.3 Building module ................................................................. 118
A.4 CityFurniture module .......................................................... 127
A.5 CityObjectGroup module ...................................................... 128
A.6 Generics module ................................................................. 129
A.7 LandUse module ............................................................... 131
A.8 Relief module ................................................................. 132
A.9 Transportation module ...................................................... 135
A.10 Vegetation module ............................................................ 139
A.11 WaterBody module .......................................................... 141
A.12 TexturedSurface module [deprecated] .................................. 144
A.13 Schematron rules on referential integrity ............................... 146

Annex B (normative) Abstract test suite for CityGML instance documents ........ 147
B.1 Test cases for mandatory conformance requirements .................. 147
  B.1.1 Valid CityGML instance document ................................... 147
  B.1.2 Valid CityGML profile .................................................... 147
  B.1.3 Conformance classes related to CityGML modules ................. 148
  B.1.4 Spatial geometry objects ............................................... 148
  B.1.5 Spatial topology relations ............................................. 148
  B.1.6 Address objects ........................................................ 148
B.2 Conformance classes related to CityGML modules ...................... 149
  B.2.1 CityGML Core module .................................................. 149
  B.2.2 Appearance module .................................................... 149
  B.2.3 Building module ........................................................ 150
  B.2.4 CityFurniture module ................................................... 150
  B.2.5 CityObjectGroup module .............................................. 151
  B.2.6 Generics module ......................................................... 151
  B.2.7 LandUse module ........................................................ 152
  B.2.8 Relief module ............................................................ 153
  B.2.9 Transportation module ................................................ 153
  B.2.10 Vegetation module ...................................................... 154
  B.2.11 WaterBody module ..................................................... 154
  B.2.12 TexturedSurface module [deprecated] ............................... 155

Annex C (informative) External code lists .................................... 157
C.1 Building ................................................................. 159
C.2 City furniture ............................................................. 166
C.3 Land use ................................................................. 167
C.4 MimeType ................................................................. 168
C.5 Vegetation ............................................................... 169
C.6 Transportation ........................................................... 171
C.7 Water body .............................................................. 174
Annex D (informative) Overview of employed GML3 geometry classes ............... 177
Annex E (informative) Overview of the assignment of features to LODs .......... 179
Annex F (informative) Example CityGML datasets ........................................ 187
  F.1 Example of a CityGML dataset for a building in LOD1 and LOD2 .......... 187
  F.2 Example of a CityGML dataset for a building in LOD3 ..................... 193
  F.3 Example of a CityGML dataset illustrating the appearance model ........ 196
  F.4 Example of a CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes ................................................................. 202
Annex G (informative) Example ADE for Noise Immission Simulation ........... 205
  G.1 CityGML Noise ADE ........................................................................... 208
  G.2 Example dataset ................................................................................ 213
Bibliography ................................................................................................. 227
i. Preface and Acknowledgements

This is the official CityGML logo. For current news on CityGML and information about ongoing projects and fields of research in the area of CityGML see http://www.citygml.org and http://www.citygmlwiki.org

CityGML is discussed and OGC work coordinated by the 3D Information Management (3DIM) Working Group of the OGC. It was implemented and evaluated within the OpenGIS Web Services Testbed, Phase 4 (OWS-4) in the CAD/GIS/BIM thread.

Version 1.0 of this standards document was prepared by the CityGML 1.0 Standards Working Group (SWG) of the OGC. Future discussion and development will be lead by the 3DIM Working Group.

For further information see http://www.opengeospatial.org/projects/groups/3dimwg

CityGML continues to be developed by the members of the Special Interest Group 3D of the initiative Geodata Infrastructure North-Rhine Westphalia (GDI NRW).

For further information see http://www.gdi-nrw.org/

The preparation of the English document version and the European discussion has been supported by the European Spatial Data Research Organization (EuroSDR; formerly known as OEEPE) in an EuroSDR Commission III project.

For further information see http://www.eurosdr.net

ii. Submitting organizations

This Implementation Standard was submitted to the Open Geospatial Consortium Inc. by the members of the CityGML 1.0 Standards Working Group of the OGC. Amongst others, this comprises the following organizations:

a) Autodesk, Inc. (primary submitter)
b) Bentley Systems, Inc. (primary submitter)
c) Technical University Berlin (submitter of technology)
d) Ordnance Survey, UK
e) University of Bonn, Germany
f) Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam

CityGML was originally developed by the Special Interest Group 3D (SIG 3D), 2002 – 2008 - www.citygml.org.

iii. Submission contact points

All questions regarding this document should be directed to the editors or the contributors (including participants in development, cf. clause iv):

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prof. Dr. Thomas H. Kolbe</td>
<td>Institute for Geodesy and Geoinformation Science,</td>
<td><a href="mailto:kolbe@igg.tu-berlin.de">kolbe@igg.tu-berlin.de</a></td>
</tr>
<tr>
<td>Claus Nagel</td>
<td>Technical University Berlin</td>
<td><a href="mailto:nagel@igg.tu-berlin.de">nagel@igg.tu-berlin.de</a></td>
</tr>
<tr>
<td>Alexandra Stadler</td>
<td></td>
<td><a href="mailto:stadler@igg.tu-berlin.de">stadler@igg.tu-berlin.de</a></td>
</tr>
</tbody>
</table>
iv. Participants in development

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ulrich Gruber, Birgit Joemann, Sandra Schlüter</td>
<td>District Administration Recklinghausen, Cadastre Department</td>
<td><a href="mailto:ulrich.gruber@kreis-recklinghausen.de">ulrich.gruber@kreis-recklinghausen.de</a>, <a href="mailto:birgit.joemann2@kreis-re.de">birgit.joemann2@kreis-re.de</a>, <a href="mailto:sandra.schlueter@kreis-re.de">sandra.schlueter@kreis-re.de</a></td>
</tr>
<tr>
<td>Dr. Joachim Benner, Karl-Heinz Häfele, Dr. Klaus Leinemann</td>
<td>Institute for Applied Computer Science, Helmholtz Research Center Karlsruhe</td>
<td><a href="mailto:Joachim.Benner@iai.fzk.de">Joachim.Benner@iai.fzk.de</a>, <a href="mailto:Karl-Heinz.Haefele@iai.fzk.de">Karl-Heinz.Haefele@iai.fzk.de</a></td>
</tr>
<tr>
<td>Frank Bildstein</td>
<td>Rheinmetall Defence Electronics</td>
<td><a href="mailto:bildstein.e@rheinmetall-de.com">bildstein.e@rheinmetall-de.com</a></td>
</tr>
<tr>
<td>Rüdiger Drees</td>
<td>T-Systems Enterprise Services GmbH, Bonn, Germany</td>
<td><a href="mailto:Ruediger.Drees@t-systems.com">Ruediger.Drees@t-systems.com</a></td>
</tr>
<tr>
<td>Andreas Kohlhaas</td>
<td>GISTec GmbH (formerly)</td>
<td><a href="mailto:AKohlhaas@t-online.de">AKohlhaas@t-online.de</a></td>
</tr>
<tr>
<td>Frank Thiemann</td>
<td>Institute for Cartography and Geoinformatics, University of Hannover</td>
<td><a href="mailto:Frank.Thiemann@ikg.uni-hannover.de">Frank.Thiemann@ikg.uni-hannover.de</a></td>
</tr>
<tr>
<td>Martin Degen</td>
<td>City of Dortmund Cadastre Department</td>
<td><a href="mailto:mdegen@stadttdo.de">mdegen@stadttdo.de</a></td>
</tr>
<tr>
<td>Heinrich Geerling</td>
<td>Architekturbüro Geerling</td>
<td><a href="mailto:Heinrich@geerling.de">Heinrich@geerling.de</a></td>
</tr>
<tr>
<td>Dr. Frank Knospe</td>
<td>City of Essen Cadastre and Mapping Department</td>
<td><a href="mailto:frank.knospe@amt62.essen.de">frank.knospe@amt62.essen.de</a></td>
</tr>
<tr>
<td>Martin Rechner</td>
<td>rechner logistik</td>
<td><a href="mailto:mail@rec-log.de">mail@rec-log.de</a></td>
</tr>
<tr>
<td>Jörg Haist, Daniel Holweg</td>
<td>Fraunhofer Institute for Computer Graphics (IGD), Darmstadt</td>
<td><a href="mailto:joerg.haist@igd.fraunhofer.de">joerg.haist@igd.fraunhofer.de</a>, <a href="mailto:daniel.holweg@igd.fraunhofer.de">daniel.holweg@igd.fraunhofer.de</a></td>
</tr>
<tr>
<td>Prof. Dr. Peter A. Henning</td>
<td>Faculty for Computer Science, University of Applied Sciences, Karlsruhe</td>
<td><a href="mailto:p.henning@fh-karlsruhe.de">p.henning@fh-karlsruhe.de</a></td>
</tr>
<tr>
<td>Rolf Wegener, Stephan Heitmann</td>
<td>State Cadastre and Mapping Agency of North-Rhine Westphalia</td>
<td><a href="mailto:wegener@lverma.nrw.de">wegener@lverma.nrw.de</a>, <a href="mailto:heitmann@lverma.nrw.de">heitmann@lverma.nrw.de</a></td>
</tr>
<tr>
<td>Prof. Dr. Marc-O. Löwner</td>
<td>Institute for Geodesy and Photogrammetry, Technical University of Braunschweig</td>
<td><a href="mailto:m-o.lowner@tu-bs.de">m-o.lowner@tu-bs.de</a></td>
</tr>
</tbody>
</table>

v. Revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Editor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-02-01</td>
<td>0.1.0</td>
<td>Czerwinski, Kolbe, Gröger</td>
<td>Initialisation of the document.</td>
</tr>
<tr>
<td>2006-02-22</td>
<td>0.1.0</td>
<td>Gröger, Gruber</td>
<td>Additions to UML diagrams.</td>
</tr>
<tr>
<td>2006-04-26</td>
<td>0.2.0</td>
<td>Kolbe, Gröger, Czerwinski</td>
<td>Release of CityGML draft standard to EuroSDR.</td>
</tr>
<tr>
<td>2006-03-06</td>
<td>0.3.0</td>
<td>Kolbe, Gröger</td>
<td>Changes on property names, some attributes added. Release of</td>
</tr>
</tbody>
</table>
vi. **Changes to the OGC® Abstract Specification**

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

vii. **Acknowledgments**

The SIG 3D wishes to thank the members of the CityGML 1.0 Standards Working Group and the 3D Information Management (3DIM) Working Group of the OGC: Tim Case, Paul Cote, Jeffrey Bell, Chris Body, Greg Buehler, David Burggraf, François Golay, John Herring, Jury Konga, Kai-Uwe Krause, Gavin Park, Richard Pearsall, George Percivall, Clemens Portele, Mauro Salvemini, Scott Simmons, Alessandro Triglia, David Wesloh, Tim Wilson, and Greg Yetman.

Further credits for careful reviewing and commenting of this document go to: Ludvig Emgard, Bettina Petzold, Dave Capstick, Mark Pendlington, Alain Lapierre, and Frank Steggink.
Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Significant changes from the previous CityGML version 0.4.0 (OGC document no. 07-062):

- Modularisation of the CityGML data model;
- Minor changes to the appearance model;
- Minor changes to city object groups;
- Minor changes to transportation objects; and
- Encoding of external code lists changed to GML 3.1.1 Simple Dictionary Profile.

With CityGML version 1.0.0, modularisation of the CityGML data model was introduced. The overall CityGML data model is thematically decomposed into a CityGML core module and extension modules. Each module is defined within its own globally unique XML namespace. Due to this modularisation approach, valid CityGML 0.4.0 instance documents are not valid CityGML 1.0.0 instance documents.
0 Introduction

0.1 Motivation

An increasing number of cities and companies are building virtual 3D city models for different application areas like urban planning, mobile telecommunication, disaster management, 3D cadastre, tourism, vehicle and pedestrian navigation, facility management and environmental simulations. Furthermore, in the implementation of the European Environmental Noise Directive (END, 2002/49/EC) 3D geoinformation and 3D city models play an important role.

In recent years, most virtual 3D city models have been defined as purely graphical or geometrical models, neglecting the semantic and topological aspects. Thus, these models could almost only be used for visualisation purposes but not for thematic queries, analysis tasks, or spatial data mining. Since the limited reusability of models inhibits the broader use of 3D city models, a more general modelling approach had to be taken in order to satisfy the information needs of the various application fields.

CityGML is a common semantic information model for the representation of 3D urban objects that can be shared over different applications. The latter capability is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the possibility of selling the same data to customers from different application fields. The targeted application areas explicitly include city planning, architectural design, tourist and leisure activities, environmental simulation, mobile telecommunication, disaster management, homeland security, real estate management, vehicle and pedestrian navigation, and training simulators.

CityGML is designed as an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is implemented as an application schema of the Geography Markup Language 3 (GML3), the extendible international standard for spatial data exchange and encoding issued by the Open Geospatial Consortium (OGC) and the ISO TC211. CityGML is based on a number of standards from the ISO 191xx family, the Open Geospatial Consortium, the W3C Consortium, the Web 3D Consortium, and OASIS.

CityGML defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and appearance properties. “City” is broadly defined to comprise not just built structures, but also elevation, vegetation, water bodies, “city furniture”, and more. Included are generalisation hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. CityGML is applicable for large areas and small regions and can represent the terrain and 3D objects in different levels of detail simultaneously. Since either simple, single scale models without topology and few semantics or very complex multi-scale models with full topology and fine-grained semantical differentiations can be represented, CityGML enables lossless information exchange between different GI systems and users.

0.2 Historical background

CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D) of the initiative Geodata Infrastructure North Rhine-Westphalia (GDI NRW) in Germany. The SIG 3D is an open group consisting of more than 70 companies, municipalities, and research institutions from Germany, Great Britain, Switzerland, and Austria working on the development and commercial exploitation of interoperable 3D models and geovisualisation. Another result of the work from the SIG 3D is the proposition of the Web 3D Service (W3DS), a 3D portrayal service that is also being discussed in the Open Geospatial Consortium (OGC Doc. No. 05-019).

A subset of CityGML has been successfully implemented and evaluated in the project “Pilot 3D” of the GDI NRW in 2005. Participants came from all over Germany and demonstrated city planning scenarios and tourist applications. Today, the official 3D city model of Berlin is based on the CityGML data model and employs CityGML as the exchange format between database, editor, and presentation systems. Also the 3D city models of Stuttgart, Bochum, Essen, Dortmund, Cologne, and Bonn are based on the CityGML model.

By the beginning of 2006, a CityGML project within EuroSDR (European Spatial Data Research) started focusing on the European harmonisation of 3D city modelling. From June to December 2006, CityGML was employed and evaluated in the CAD/GIS/BIM thread of the OpenGIS Web Services Testbed #4 (OWS-4).
OpenGIS® City Geography Markup Language (CityGML) Encoding Standard

1 Scope

This document is an OpenGIS® Encoding Standard for the representation, storage and exchange of virtual 3D city and landscape models. CityGML is implemented as an application schema of the Geography Markup Language version 3.1.1 (GML3).

CityGML models both complex and georeferenced 3D vector data along with the semantics associated with the data. In contrast to other 3D vector formats, CityGML is based on a rich, general purpose information model in addition to geometry and appearance information. For specific domain areas, CityGML also provides an extension mechanism to enrich the data with identifiable features under preservation of semantic interoperability.

Targeted application areas explicitly include urban and landscape planning; architectural design; tourist and leisure activities; 3D cadastres; environmental simulations; mobile telecommunications; disaster management; homeland security; vehicle and pedestrian navigation; training simulators and mobile robotics.

CityGML is considered a source format for 3D portraying. The semantic information contained in the model can be used in the styling process which generates computer graphics represented e.g. as KML/COLLADA or X3D files. The appropriate OGC Portrayal Web Service for this process is the OGC Web 3D Service (W3DS).

Features of CityGML:

- Geospatial information model (ontology) for urban landscapes based on the ISO 191xx family
- GML3 representation of 3D geometries, based on the ISO 19107 model
- Representation of object surface characteristics (e.g. textures, materials)
- Taxonomies and aggregations
  - Digital Terrain Models as a combination of (including nested) triangulated irregular networks (TINs), regular rasters, break and skeleton lines, mass points
  - Sites (currently buildings; bridges and tunnels in the future)
  - Vegetation (areas, volumes and solitary objects with vegetation classification)
  - Water bodies (volumes, surfaces)
  - Transportation facilities (both graph structures and 3D surface data)
  - Land use (representation of areas of the earth’s surface dedicated to a specific land use)
  - City furniture
  - Generic city objects and attributes
  - User-definable (recursive) grouping
- Multiscale model with 5 well-defined consecutive Levels of Detail (LOD):
  - LOD0 – regional, landscape
  - LOD1 – city, region
  - LOD2 – city districts, projects
  - LOD3 – architectural models (outside), landmarks
  - LOD4 – architectural models (interior)
- Multiple representations in different LODs simultaneously; generalisation relations between objects in different LODs
- Optional topological connections between feature (sub)geometries
- Application Domain Extensions (ADE): Specific “hooks” in the CityGML schema allow to define application specific extensions, for example for noise pollution simulation, or to augment CityGML by properties of the new National Building Information Model Standard (NBIMS) in the U.S.
2 Conformance

Conformance targets addressed by this International standard are CityGML instance documents only. Future revisions of this International Standard may also address consumers or producers as conformance targets.

Clauses 8 to 10 of this International standard specify separate CityGML XML Schema definitions and normative aspects, i.e. CityGML modules, which shall be used in CityGML instance documents in accordance with clause 7. Implementations are not required to support the full range of capabilities provided by the universe of all CityGML modules. Valid partial implementations are supported following the rules and guidelines for CityGML profiles in chapter 7.2.

CityGML instance documents claiming conformance to this International Standard shall:

a) conform to the rules and requirements specified in clauses 7 to 10;

b) pass all relevant test cases of the abstract test suite in annex B.1;

c) satisfy all relevant conformance classes of the abstract test suite related to CityGML modules in annex B.2.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of OGC 08-007r1. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 08-007r1 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

The following documents are indispensable for the application of the CityGML standard. The geometry model of GML 3.1.1 is used except for some added concepts like implicit geometries (see chapter 8.2). The appearance model (see chapter 9) draws concepts from both X3D and COLLADA. Addresses are represented using the OASIS extensible Address Language xAL.

ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times


ISO 19105:2000, Geographic information – Conformance and testing

ISO 19107:2003, Geographic Information – Spatial Schema

ISO 19109:2005, Geographic Information – Rules for Application Schemas

ISO 19111:2003, Geographic information – Spatial referencing by coordinates

ISO 19115:2003, Geographic Information – Metadata

ISO 19123:2005, Geographic Information – Coverages


OpenGIS® Abstract Specification Topic 0, Overview, OGC document 04-084


OpenGIS® Abstract Specification Topic 8, Relations between Features, OGC document 99-108r2


OpenGIS® Geography Markup Language Implementation Specification, Version 3.1.1, OGC document 03-105r1

OpenGIS® GML 3.1.1 Simple Dictionary Profile, Version 1.0.0, OGC document 05-099r2

IETF RFC 2045 & 2046, Multipurpose Internet Mail Extensions (MIME). (November 1996)


W3C XLink, XML Linking Language (XLink) Version 1.0. W3C Recommendation (27 June 2001)
W3C XML Name, Namespaces in XML. W3C Recommendation (14 January 1999)
W3C XML Base, XML Base, W3C Recommendation (27 June 2001)
W3C XML, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)
OASIS (Organization for the Advancement of Structured Information Standards): extensible Address Language (xAL v2.0).
Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.4.1
The Schematron Assertion Language 1.5. Rick Jelliffe 2002-10-01

4 Conventions

4.1 Abbreviated terms

The following abbreviated terms are used in this document:

2D Two Dimensional
3D Three Dimensional
AEC Architecture, Engineering, Construction
ALKIS German National Standard for Cadastral Information
ATKIS German National Standard for Topographic and Cartographic Information
B-Rep Boundary Representation
CAD Computer Aided Design
COLLADA Collaborative Design Activity
CSG Constructive Solid Geometry
DTM Digital Terrain Model
DXF Drawing Exchange Format
EuroSDR European Spatial Data Research Organisation
ESRI Environmental Systems Research Institute
FM Facility Management
GDF Geographic Data Files
GDI NRW Geodata Infrastructure North-Rhine Westphalia
GML Geography Markup Language
IAI International Alliance for Interoperability
IETF Internet Engineering Task Force
IFC Industry Foundation Classes
ISO International Organization for Standardisation
LOD Level of Detail
NBIMS National Building Information Model Standard
OASIS Organisation for the Advancement of Structured Information Standards
OGC Open Geospatial Consortium
OSCRE Open Standards Consortium for Real Estate
SIG 3D Special Interest Group 3D of the GDI NRW
TC211 ISO Technical Committee 211
The CityGML standard is presented in this document in diagrams using the Unified Modeling Language (UML) static structure diagram (see Booch et al. 1997). The UML notations used in this standard are described in the diagram below (Fig. 1).

According to GML3 all associations between model elements in CityGML are uni-directional. Thus, associations in CityGML are navigable in only one direction. The direction of navigation is depicted by an arrowhead. In general, the context an element takes within the association is indicated by its role. The role is displayed near the target of the association. If the graphical representation is ambiguous though, the position of the role has to be drawn to the element the association points to.

The following stereotypes are used:

`<<Geometry>>` represents the geometry of an object. The geometry is an identifiable and distinguishable object that is derived from the abstract GML type `AbstractGeometryType`. 
<<Feature>> represents a thematic feature according to the definition in ISO 19109. A feature is an identifiable and distinguishable object that is derived from the abstract GML type AbstractFeatureType.

<<Object>> represents an identifiable and distinguishable object that is derived from the abstract GML type AbstractGMLType.

<<CodeList>> enumerates the valid attribute values.

<<ExternalCodeList>> enumerates the valid attributes values. In contrast to CodeList, the values are not given inline the schema but are provided within an external dictionary file. External code lists are encoded using GML 3.1.1 Simple Dictionary Profile (see chapter 6.6).

<<Union>> is a list of attributes. The semantics are that only one of the attributes can be present at any time.

<<PrimitiveType>> is used for representations supported by a primitive type in the implementation.

<<DataType>> is used as a descriptor of a set of values that lack identity. Data types include primitive predefined types and user-definable types. A DataType is thus a class with few or no operations whose primary purpose is to hold the abstract state of another class for transmittal, storage, encoding or persistent storage.

<<Leaf>> is used within UML package diagrams to indicate model elements that can have no further subtypes.

<<XSDSchema>> is used within UML package diagrams to denote the root element of an XSD Schema containing all the definitions for a particular namespace. All the package contents or component classes are placed within the one schema.

<<ApplicationSchema>> is used within UML package diagrams to denote an XML Schema definition fundamentally dependent on the concepts of another independent Standard within the XML Schema meta-language. For example, ApplicationSchema indicates extensions of GML consistent with the GML “rules for application schemas”.

4.3 XML namespaces and namespace prefixes

The CityGML data model is thematically decomposed into a core module and thematic extension modules. All modules including the core are specified by their own XML schema file, each defining a globally unique XML namespace. The extension modules are based on the core module and, thus, contain (by reference) the CityGML core schema.

Within this document the module namespaces are associated with recommended prefixes. These prefixes are consistently used within the normative parts of this specification, for all UML diagrams and example CityGML instance documents. The CityGML core and extension modules along with their XML namespace identifiers and recommended namespace prefixes are listed in Tab. 1.

<table>
<thead>
<tr>
<th>CityGML module</th>
<th>Namespace identifier</th>
<th>Namespace prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityGML Core</td>
<td><a href="http://www.opengis.net/citygml/1.0">http://www.opengis.net/citygml/1.0</a></td>
<td>core</td>
</tr>
<tr>
<td>Appearance</td>
<td><a href="http://www.opengis.net/citygml/appearance/1.0">http://www.opengis.net/citygml/appearance/1.0</a></td>
<td>app</td>
</tr>
<tr>
<td>Building</td>
<td><a href="http://www.opengis.net/citygml/building/1.0">http://www.opengis.net/citygml/building/1.0</a></td>
<td>bldg</td>
</tr>
<tr>
<td>CityFurniture</td>
<td><a href="http://www.opengis.net/citygml/cityfurniture/1.0">http://www.opengis.net/citygml/cityfurniture/1.0</a></td>
<td>frn</td>
</tr>
<tr>
<td>CityObjectGroup</td>
<td><a href="http://www.opengis.net/citygml/cityobjectgroup/1.0">http://www.opengis.net/citygml/cityobjectgroup/1.0</a></td>
<td>grp</td>
</tr>
<tr>
<td>Generics</td>
<td><a href="http://www.opengis.net/citygml/generics/1.0">http://www.opengis.net/citygml/generics/1.0</a></td>
<td>gen</td>
</tr>
<tr>
<td>LandUse</td>
<td><a href="http://www.opengis.net/citygml/landuse/1.0">http://www.opengis.net/citygml/landuse/1.0</a></td>
<td>luse</td>
</tr>
<tr>
<td>Relief</td>
<td><a href="http://www.opengis.net/citygml/relief/1.0">http://www.opengis.net/citygml/relief/1.0</a></td>
<td>dem</td>
</tr>
<tr>
<td>Transportation</td>
<td><a href="http://www.opengis.net/citygml/transportation/1.0">http://www.opengis.net/citygml/transportation/1.0</a></td>
<td>tran</td>
</tr>
<tr>
<td>Vegetation</td>
<td><a href="http://www.opengis.net/citygml/vegetation/1.0">http://www.opengis.net/citygml/vegetation/1.0</a></td>
<td>veg</td>
</tr>
<tr>
<td>WaterBody</td>
<td><a href="http://www.opengis.net/citygml/waterbody/1.0">http://www.opengis.net/citygml/waterbody/1.0</a></td>
<td>wtr</td>
</tr>
<tr>
<td>TexturedSurface [deprecated]</td>
<td><a href="http://www.opengis.net/citygml/texturedsurface/1.0">http://www.opengis.net/citygml/texturedsurface/1.0</a></td>
<td>tex</td>
</tr>
</tbody>
</table>

Tab. 1: List of CityGML modules, their associated XML namespace identifiers, and example namespace prefixes.
Further XML Schema definitions relevant to this standard are shown in Tab. 2 along with the corresponding XML namespace identifiers and namespace prefixes consistently used within this document.

<table>
<thead>
<tr>
<th>XML Schema definition</th>
<th>Namespace identifier</th>
<th>Namespace prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geography Markup Language version 3.1.1</td>
<td><a href="http://www.opengis.net/gml">http://www.opengis.net/gml</a></td>
<td>gml</td>
</tr>
<tr>
<td>(from OGC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensible Address Language version 2.0</td>
<td>urn:oasis:names:tc:ciq:xsd:address</td>
<td>xAL</td>
</tr>
<tr>
<td>(from OASIS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schematron Assertion Language version 1.5</td>
<td><a href="http://www.ascc.net/xml/schematron">http://www.ascc.net/xml/schematron</a></td>
<td>sch</td>
</tr>
</tbody>
</table>

Tab. 2: List of XML Schema definitions, their associated XML namespace identifiers, and example namespace prefixes used within this document.

### 4.4 XML-Schema

The normative parts of the standard use the W3C XML schema language to describe the grammar of conformant CityGML data instances. XML schema is a rich language with many capabilities. While a reader who is unfamiliar with an XML schema may be able to follow the description in a general fashion, this standard is not intended to serve as an introduction to XML schema. In order to have a full understanding of this candidate standard, it is necessary for the reader to have a reasonable knowledge of XML schema.
5 Overview of CityGML

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

CityGML not only represents the graphical appearance of city models but specifically addresses the representation of the semantic and thematic properties, taxonomies and aggregations. CityGML includes a geometry model and a thematic model. The geometry model allows for the consistent and homogeneous definition of geometrical and topological properties of spatial objects within 3D city models (chapter 8). The base class of all objects is CityObject which is a subclass of the GML class Feature. All objects inherit the properties from CityObject.

The thematic model of CityGML employs the geometry model for different thematic fields like Digital Terrain Models, sites (i.e. buildings; future extensions of CityGML will also include explicit models for bridges and tunnels), vegetation (solitary objects and also areal and volumetric biotopes), water bodies, transportation facilities, and city furniture (chapter 10). Further objects, which are not explicitly modelled yet, can be represented using the concept of generic objects and attributes (chapter 6.11). In addition, extensions to the CityGML data model applying to specific application fields can be realised using the Application Domain Extensions (ADE) (chapter 6.12). Spatial objects of equal shape which appear many times at different positions like e.g. trees, can also be modelled as prototypes and used multiple times in the city model (chapter 8.2). A grouping concept allows the combination of single 3D objects, e.g. buildings to a building complex (chapter 6.8). Objects which are not geometrically modelled by closed solids can be virtually sealed in order to compute their volume (e.g. pedestrian underpasses, tunnels, or airplane hangars). They can be closed using ClosureSurfaces (chapter 6.4). The concept of the TerrainIntersectionCurve is introduced to integrate 3D objects with the Digital Terrain Model at their correct positions in order to prevent e.g. buildings from floating over or sinking into the terrain (chapter 6.5).

CityGML differentiates five consecutive Levels of Detail (LOD), where objects become more detailed with increasing LOD regarding both their geometry and thematic differentiation (chapter 6.2). CityGML files can - but do not have to - contain multiple representations (and geometries) for each object in different LOD simultaneously. Generalisation relations allow the explicit representation of aggregated objects over different scales.

In addition to spatial properties, CityGML features can be assigned appearances. Appearances are not limited to visual data but represent arbitrary observable properties of the feature’s surface such as infrared radiation, noise pollution, or earthquake-induced structural stress (chapter 9).

Furthermore, objects can have external references to corresponding objects in external datasets (chapter 6.7). Enumerative object attributes are restricted to external code lists and values defined in external, definable dictionaries (chapter 6.6).
6 General characteristics of CityGML

6.1 Modularisation

The CityGML data model consists of class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. However, implementations are not required to support the overall CityGML data model in order to be conformant to the standard, but may employ a subset of constructs according to their specific information needs. For this purpose, modularisation is applied to the CityGML data model (cf. chapter 7).

The CityGML data model is thematically decomposed into a core module and thematic extension modules. The core module comprises the basic concepts and components of the CityGML data model and, thus, must be implemented by any conformant system. Based on the core module, each extension covers a specific thematic field of virtual 3D city models. CityGML introduces the following eleven thematic extension modules: Appearance, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Vegetation, WaterBody, and TexturedSurface [deprecated].

CityGML compliant implementations may support any combination of extension modules in conjunction with the core module. Such combinations of modules are called CityGML profiles. Therefore, CityGML profiles allow for valid partial implementations of the overall CityGML data model.

6.2 Multi-scale modelling (5 levels of detail, LOD)

CityGML supports different Levels of Detail (LOD). LODs are required to reflect independent data collection processes with differing application requirements. Further, LODs facilitate efficient visualisation and data analysis (see Fig. 2). In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualisation of the same object with regard to different degrees of resolution. Furthermore, two CityGML data sets containing the same object in different LOD may be combined and integrated. However, it will be within the responsibility of the user or application to make sure objects in different LOD refer to the same real-world object.

The coarsest level LOD0 is essentially a two and a half dimensional Digital Terrain Model, over which an aerial image or a map may be draped. LOD1 is the well-known blocks model comprising prismatic buildings with flat roofs. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated surfaces. Vegetation objects may also be represented. LOD3 denotes architectural models with detailed wall and roof structures, balconies, bays and projections. High-resolution textures can be mapped onto these structures. In addition, detailed vegetation and transportation objects are components of a LOD3 model. LOD4 completes a LOD3 model by adding interior structures for 3D objects. For example, buildings are composed of rooms, interior doors, stairs, and furniture.

![Fig. 2: The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn)](image)
LODs are also characterised by differing accuracies and minimal dimensions of objects (Tab. 3). The accuracy requirements given in this standard are debatable and should be considered as discussion proposals. Accuracy is described as standard deviation \( \sigma \) of the absolute 3D point coordinates. Relative 3D point accuracy will be added in a future version of CityGML and it is typically much higher than the absolute accuracy. In LOD1, the positional and height accuracy of points must be 5m or less, while all objects with a footprint of at least 6m by 6m have to be considered. The positional and height accuracy of LOD2 must be 2m or better. In this LOD, all objects with a footprint of at least 4m × 4m have to be considered. Both types of accuracies in LOD3 are 0.5m, and the minimal footprint is 2m × 2m. Finally, the positional and height accuracy of LOD4 must be 0.2m or less. By means of these figures, the classification in five LOD may be used to assess the quality of 3D city model datasets. The LOD categorisation makes datasets comparable and provides support for their integration.

<table>
<thead>
<tr>
<th>Model scale description</th>
<th>LOD0</th>
<th>LOD1</th>
<th>LOD2</th>
<th>LOD3</th>
<th>LOD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class of accuracy</td>
<td>regional, landscape</td>
<td>city, region</td>
<td>city districts, projects</td>
<td>architectural models (outside), landmark</td>
<td>architectural models (interior)</td>
</tr>
<tr>
<td>Absolute 3D point accuracy (position / height)</td>
<td>lower than LOD1</td>
<td>5/5m</td>
<td>2/2m</td>
<td>0.5/0.5m</td>
<td>0.2/0.2m</td>
</tr>
<tr>
<td>Generalisation</td>
<td>maximal generalisation (classification of land use)</td>
<td>object blocks as generalised features; &gt; 6*6m/3m</td>
<td>objects as generalised features; &gt; 4*4m/2m</td>
<td>object as real features; &gt; 2*2m/1m</td>
<td>constructive elements and openings are represented</td>
</tr>
<tr>
<td>Building installations</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>representative exterior effects</td>
<td>real object form</td>
</tr>
<tr>
<td>Roof form/structure</td>
<td>no</td>
<td>flat</td>
<td>roof type and orientation</td>
<td>real object form</td>
<td>real object form</td>
</tr>
<tr>
<td>Roof overhanging parts</td>
<td>-</td>
<td>-</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>CityFurniture</td>
<td>-</td>
<td>important objects</td>
<td>prototypes</td>
<td>real object form</td>
<td>real object form</td>
</tr>
<tr>
<td>SolitaryVegetationObject</td>
<td>-</td>
<td>important objects</td>
<td>prototypes, higher 6m</td>
<td>prototypes, higher 2m</td>
<td>prototypes, real object form</td>
</tr>
<tr>
<td>PlantCover</td>
<td>-</td>
<td>&gt;50*50m</td>
<td>&gt;5*5m</td>
<td>&lt; LOD2</td>
<td>&lt; LOD2</td>
</tr>
</tbody>
</table>

Tab. 3: LOD 0–4 of CityGML with its accuracy requirements (source: Albert et al. 2003).

Whereas in CityGML each object can have a different representation for every LOD, often different objects from the same LOD will be generalised to be represented by an aggregate object in a lower LOD. CityGML supports the aggregation / decomposition by providing an explicit generalisation association between any CityObjects (further details see UML diagram in chapter 10.1).

### 6.3 Coherent semantical-geometrical modelling

One of the most important design principles for CityGML is the coherent modelling of semantics and geometrical/topological properties. At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms. The description also includes attributes, relations and aggregation hierarchies (part-whole-relations) between features. Thus the part-of-relationship between features can be derived at the semantic level only, without considering geometry. However, at the spatial level, geometry objects are assigned to features representing their spatial location and extent. So the model consists of two hierarchies: the semantic and the geometrical in which the corresponding objects are linked by relationships (cf. Stadler & Kolbe 2007). The advantage of this approach is that it can be navigated in both hierarchies and between both hierarchies arbitrarily, for answering thematic and/or geometrical queries or performing analyses.

If both hierarchies exist for a specific object, they must be coherent (i.e. it must be ensured that they match and fit together). For example, if a wall of a building has two windows and a door on the semantic level, then the geometry representing the wall must contain also the geometry parts of both windows and the door.

### 6.4 Closure surfaces

Objects, which are not modelled by a volumetric geometry, must be virtually closed in order to compute their volume (e.g. pedestrian underpasses or airplane hangars). They can be sealed using ClosureSurfaces. Closure-
Surfaces are special surfaces, which are taken into account, when needed to compute volumes and are neglected, when they are irrelevant or not appropriate, for example in visualisations.

The concept of ClosureSurfaces is also employed to model the entrances of subsurface objects. Those objects like tunnels or pedestrian underpasses have to be modelled as closed solids in order to compute their volume, for example in flood simulations. The entrances to subsurface objects also have to be sealed to avoid holes in the digital terrain model (see Fig. 3). However, in close-range visualisations the entrance must be treated as open. Thus, ClosureSurfaces are an adequate way to model those entrances.

6.5 Terrain Intersection Curve (TIC)

A crucial issue in city modelling is the integration of 3D objects and the terrain. Problems arise if 3D objects float over or sink into the terrain. This is particularly the case if terrains and 3D objects in different LOD are combined, or if they come from different providers (Kolbe and Gröger 2003). To overcome this problem, the TerrainIntersectionCurve (TIC) of a 3D object is introduced. These curves denote the exact position, where the terrain touches the 3D object (see Fig. 4). TICs can be applied to buildings and building parts (cf. chapter 10.3), city furniture objects (cf. chapter 10.7), and generic city objects (cf. chapter 10.10). If, for example, a building has a courtyard, the TIC consists of two closed rings: one ring representing the courtyard boundary, and one which describes the building’s outer boundary. This information can be used to integrate the building and a terrain by ‘pulling up’ or ‘pulling down’ the surrounding terrain to fit the TerrainIntersectionCurve. The DTM may be locally warped to fit the TIC. By this means, the TIC also ensures the correct positioning of textures or the matching of object textures with the DTM. Since the intersection with the terrain may differ depending on the LOD, a 3D object may have different TerrainIntersectionCurves for all LOD.
6.6 **Dictionaries and external code lists for enumerative attributes**

Attributes, which are used to classify objects, often have values that are restricted to a number of discrete values. An example is the attribute *roof type*, whose attribute values typically are saddle back roof, hip roof, semi-hip roof, flat roof, pent roof, or tent roof. If such an attribute is typed as string, misspellings or different names for the same notion obstruct interoperability. In CityGML such classifying of attributes is specified as *External-CodeLists* and implemented by *simple dictionaries* defined in the GML 3.1.1 Simple Dictionary Profile (cf. Whiteside 2005). Such a structure enumerates all possible values of the attribute in an external file, ensuring that the same name is used for the same notion. In addition, the translation of attribute values into other languages is facilitated.

Simple dictionaries and external code lists may be extended or redefined by users. They can have references to existing models. For example, room codes defined by the Open Standards Consortium for Real Estate (OSCRE) can be referenced instead of CityGML’s predefined values. Likewise, classifications of buildings and building parts introduced by the National Building Information Model Standard (NBIMS) can be used alternatively.

6.7 **External references**

3D objects are often derived from or have relations to objects in other databases or data sets. For example, a 3D building model may have been constructed from a two-dimensional footprint in a cadastre data set, or may be derived from an architectural model (Fig. 5). The reference of a 3D object to its corresponding object in an external data set is essential, if an update must be propagated or if additional data is required, for example the name and address of a building’s owner in a cadastral information system or information on antennas and doors in a facility management system. In order to supply such information, each *CityObject* may have *External References* to corresponding objects in external data sets (for the UML diagram see Fig. 20; and for XML schema definition see annex A.1). Such a reference denotes the external information system and the unique identifier of the object in this system. Both are specified as a *Uniform Resource Identifier (URI)*, which is a generic format for references to any kind of resources on the internet. The generic concept of external references allows for any *CityObject* an arbitrary number of links to corresponding objects in external information systems (e.g. ALKIS, ATKIS, OS MasterMap®, GDF, etc.).

![Fig. 5: External references (graphic: IGG Uni Bonn).](image)

6.8 **City object groups**

The grouping concept of CityGML allows for the aggregation of arbitrary city objects according to user-defined criteria, and to represent and transfer these aggregations as part of a city model (for the UML diagram see chapter 10.9; XML schema definition see annex A.5). A group may be assigned one or more names and may be further classified by specific attributes, for example, "escape route from room no. 43 in house no. 1212 in a fire scenario" as a name and "escape route" as type. Each member of the group can optionally be assigned a role name, which specifies the role this particular member plays in the group. This role name may, for example, describe the sequence number of this object in an escape route, or in the case of a building complex, denote the main building.

A group may contain other groups as members, allowing nested grouping of arbitrary depth. The grouping concept is delivered by the thematic extension module *CityObjectGroup* of CityGML (cf. chapter 10.9).
6.9 **Appearances**

Information about a surface’s appearance, i.e. observable properties of the surface, is considered an integral part of virtual 3D city models in addition to semantics and geometry. Appearance relates to any surface-based theme, e.g. infrared radiation or noise pollution, not just visual properties. Consequently, data provided by appearances can be used as input for both presentation of and analysis in virtual 3D city models.

CityGML supports feature appearances for an arbitrary number of themes per city model. Each LOD of a feature can have an individual appearance. Appearances can represent – among others – textures and georeferenced textures. CityGML’s appearance model is packaged within its own extension module Appearance (cf. chapter 9).

6.10 **Prototypic objects / scene graph concepts**

In CityGML objects of equal shape like trees and other vegetation objects, traffic lights and traffic signs can be represented as prototypes which are instantiated multiple times at different locations (Fig. 6). The geometry of prototypes is defined in local coordinate systems. Every instance is represented by a reference to the prototype, a base point in the world coordinate reference system and a transformation matrix that facilitates scaling, rotation, and translation of the prototype. The principle is adopted from the concept of scene graphs used in computer graphics standards like VRML and X3D. As the GML3 geometry model does not provide support for scene graph concepts, it is implemented as an extension to the GML3 geometry model (for further description see chapter 8.2).

![Fig. 6: Examples of prototypic shapes (source: Rheinmetall Defence Electronics).](image)

6.11 **Generic city objects and attributes**

CityGML is being designed as a universal topographic information model that defines object types and attributes which are useful for a broad range of applications. In practical applications the objects within specific 3D city models will most likely contain attributes which are not explicitly modelled in CityGML. Moreover, there might be 3D objects which are not covered by the thematic classes of CityGML. CityGML provides two different concepts to support the exchange of such data: 1) generic objects and attributes, and 2) Application Domain Extensions (see chapter 6.12).

The concept of generic objects and attributes allows for the extension of CityGML applications during runtime, i.e. any CityObject may be augmented by additional attributes, whose names, data types, and values can be provided by a running application without any change of the CityGML XML schema. Similarly, features not represented by the predefined thematic classes of the CityGML data model may be modelled and exchanged using generic objects. The generic extensions of CityGML are provided by the thematic extension module Generics (cf. chapter 10.10).
The current version of CityGML does not include explicit thematic models for bridges, tunnels, and walls. They will be added in a future version. In the meantime, these objects may be stored or exchanged using generic objects and attributes.

6.12 Application Domain Extensions (ADE)

Application Domain Extensions (ADE) specify additions to the CityGML data model. Such additions comprise the introduction of new properties to existing CityGML classes like e.g. the number of habitants of a building or the definition of new object types. The difference between ADEs and generic objects and attributes is, that an ADE has to be defined in an extra XML schema definition file with its own namespace. This file has to explicitly import the XML Schema definition of the extended CityGML modules.

The advantage of this approach is that the extension is formally specified. Extended CityGML instance documents can be validated against the CityGML and the respective ADE schema. ADEs can be defined (and even standardised) by information communities which are interested in specific application fields. More than one ADE can be actively used in the same dataset (further description cf. chapter 10.11).

ADEs may be defined for one or even several CityGML modules providing a high flexibility in adding additional information to the CityGML data model. Thus, the ADE mechanism is orthogonally aligned with the modularisation approach of CityGML. Consequently, there is no separate extension module for ADEs.

Recently, a first ADE for noise pollution simulation has been developed, which is employed in the simulation of environmental noise dispersion according to the Environmental Noise Directive of the European Commission (2002/49/EC). Annex G shows and explains the CityGML Noise ADE as an example.
7 Modularisation

CityGML is a rich standard both on the thematic and geometric-topological level of its data model. On its thematic level CityGML defines classes and relations for the most relevant topographic objects in cities and regional models comprising built structures, elevation, vegetation, water bodies, “city furniture”, and more. In addition to geometry and appearance content these thematic components allow to employ virtual 3D city models for sophisticated analysis tasks in different application domains like simulations, urban data mining, facility management, and thematic inquiries.

CityGML is to be seen as a framework giving geospatial 3D data enough space to grow in geometrical, topological and semantically aspects over its lifetime. Thus, geometry and semantics of city objects may be flexibly structured covering purely geometric datasets up to complex geometric-topologically sound and spatio-semantically coherent data. By this means, CityGML defines a single object model and data exchange format applicable to consecutive process steps of 3D city modelling from geometry acquisition, data qualification and refinement to preparation of data for specific end-user applications, allowing for iterative data enrichment and lossless information exchange.

According to this idea of a framework, applications are not required to support all thematic fields of CityGML in order to be compliant to the standard, but may employ a subset of constructs corresponding to specific relevant requirements of an application domain or process step. The use of logical subsets of CityGML limits the complexity of the overall data model and explicitly allows for valid partial implementations. As for version 1.0 of the CityGML standard, possible subsets of the data model are defined and embraced by so called CityGML modules. A CityGML module is an aggregate of normative aspects that must all be implemented as a whole by a conformant system. CityGML consists of a core module and thematic extension modules.

The CityGML core module defines the basic concepts and components of the CityGML data model. It is to be seen as the universal lower bound of the overall CityGML data model and a dependency of all thematic extension modules. Thus, the core module is unique and must be implemented by any conformant system. Based on the CityGML core module, each extension module contains a logically separate thematic component of the CityGML data model. The extensions to the core are derived by vertically slicing the overall CityGML data model. Since the core module is contained (by reference) in each extension module, its general concepts and components are universal to all extension modules. The following eleven thematic extension modules are introduced by version 1.0 of the CityGML standard. They are directly related to clauses of this document each covering the corresponding thematic field of CityGML:

- Appearance (cf. clause 9),
- Building (cf. clause 10.3),
- CityFurniture (cf. clause 10.7),
- CityObjectGroup (cf. clause 10.9),
- Generics (cf. clause 10.10),
- LandUse (cf. clause 10.8),
- Relief (cf. clause 10.2),
- Transportation (cf. clause 10.5),
- Vegetation (cf. clause 10.6),
- WaterBody (cf. clause 10.4), and
- TexturedSurface [deprecated] (cf. clause 9.7).

The thematic decomposition of the CityGML data model allows for implementations to support any combination of extension modules in conjunction with the core module in order to be CityGML conformant. Thus, the extension modules may be arbitrarily combined according to the information needs of an application or application domain. A combination of modules is called a CityGML profile. The union of all modules is defined as the CityGML base profile. The base profile is unique at any given time and forms the upper bound of the overall CityGML data model. Any other CityGML profile must be a valid subset of the base profile. By following the
concept of CityGML modules and profiles, valid partial implementations of the CityGML data model may be realised in a well-defined way.

As for future development, each CityGML module may be further developed independently from other modules by expert groups and information communities. Resulting proposals and changes to modules may be introduced into future revisions of the CityGML standard without affecting the validity of other modules. Furthermore, thematic components not covered by the current CityGML data model may be added to future revisions of the standard by additional thematic extension modules. These additional extensions may establish dependency relations to any other existing CityGML module but shall at least be dependent on the CityGML core module. Consequently, the CityGML base profile may vary over time as new extensions are added. However, if a specific application has information needs to be modelled and exchanged which are beyond the scope of the CityGML data model, this application data can also be incorporated within the existing modules using CityGML’s Application Domain Extension mechanism (cf. clause 10.11) or by employing the concepts of generic city objects and attributes (cf. chapter 10.10).

The introduced modularisation approach supports CityGML’s versatility as a data modelling framework and exchange format addressing various application domains and different steps of 3D city modelling. For sake of clarity, applications should announce the level of conformance to the CityGML standard by declaring the employed CityGML profile. Since the core module is part of all profiles, this should be realised by enumerating the implemented thematic extension modules. For example, if an implementation supports the Building module, the Relief module, and the Vegetation module in addition to the core, this should be announced by “CityGML [Building, Relief, Vegetation]”. In case the base profile is supported, this should be indicated by “CityGML [full]”.

7.1 CityGML core and extension modules

Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However, a single namespace shall not be directly included in two modules. Thus, all elements belonging to one module are associated to the module’s namespace only. By this means, module elements are guaranteed to be properly separated and distinguishable in CityGML instance documents.

Compared to previous CityGML versions, the aforementioned namespace conventions introduce an extra level of complexity to data files as there is no single CityGML namespace any more. In contrast, components of different CityGML modules and, thus, of different namespaces may be arbitrarily mixed within the same CityGML instance document. Furthermore, an application might have to parse instance documents containing elements of modules which are not employed by the application itself. These parsing problems though can easily be overcome by non-“schema-aware” applications, i.e. applications that do not parse and interpret GML application schemas in a generic way. Elements from different namespaces than those declared by the application’s employed CityGML profile could be skipped. Comparable observations have to be made when using CityGML’s Application Domain Extension mechanism (cf. clause 10.11).

As for version 1.0 of the CityGML standard, there are no two thematic extension modules related by dependency. Thus, all extension modules are truly independent from each other and may be separately supported by implementations. However, the CityGML core module is a dependency for any extension module. This means that the XML schema file of the core module is imported by each XML schema file defining an extension.

The dependency relations between CityGML’s modules are illustrated in Fig. 7 using an UML package diagram. Each module is represented by a package. The package names correspond to the module names. A dashed arrow in the figure indicates that the schema at the tail of the arrow depends upon the schema at the head of the arrow. For CityGML modules, a dependency occurs where one schema <import>s another schema and accordingly the corresponding XML namespace. For example, the extension module Building imports the schema of the CityGML Core module. A short description of each module is given in Tab. 4.
Fig. 7: 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.

<table>
<thead>
<tr>
<th>Module name</th>
<th>CityGML Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/1.0">http://www.opengis.net/citygml/1.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>cityGMLBase.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>core</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>CityGML Core</em> module defines the basic components of the CityGML data model. Primarily, this comprises abstract base classes from which all thematic classes are (transitively) derived. But also non-abstract content common to more than one extension module, for example basic data types, is defined within the core module. The core module itself imports the XML schema definition files of GML version 3.1.1 and the OASIS extensible Address Language xAL.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/appearance/1.0">http://www.opengis.net/citygml/appearance/1.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>appearance.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>app</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Appearance</em> module provides the means to model appearances of CityGML features, i.e. observable properties of the feature’s surface. Appearance data may be stored for each city object. Therefore, the abstract base class <em>CityObject</em> defined within the core module is augmented by an additional property using CityGML’s <em>Application Domain Extension</em> mechanism. Thus, the <em>Appearance</em> module has a deliberate impact on all thematic extension modules.</td>
</tr>
<tr>
<td>Module name</td>
<td>Building</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/building/1.0">http://www.opengis.net/citygml/building/1.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>building.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>bldg</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Building</em> module allows for the representation of thematic and spatial aspects of buildings, building parts, building installations, and interior building structures in four levels of detail (LOD 1 – 4).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>CityFurniture</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/cityfurniture/1.0">http://www.opengis.net/citygml/cityfurniture/1.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>cityFurniture.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>frn</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>CityFurniture</em> module is used to represent city furniture objects in cities. City furniture objects are immovable objects like lanterns, traffic signs, advertising columns, benches, or bus stops that can be found in traffic areas, residential areas, on squares, or in built-up areas.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>CityObjectGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/cityobjectgroup/1.0">http://www.opengis.net/citygml/cityobjectgroup/1.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>cityObjectGroup.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>grp</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>CityObjectGroup</em> module provides a grouping concept for CityGML. Arbitrary city objects may be aggregated in groups according to user-defined criteria to represent and transfer these aggregations as part of the city model. A group may be further classified by specific attributes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Generics</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/generics/1.0">http://www.opengis.net/citygml/generics/1.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>generics.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>gen</td>
</tr>
</tbody>
</table>
| Module description| The *Generics* module provides generic extensions to the CityGML data model that may be used to model and exchange additional attributes and features not covered by the predefined thematic classes of CityGML. However, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.  
In order to represent generic attributes, the *Generics* module augments the abstract base class *CityObject* defined within the core module by an additional property using CityGML’s *Application Domain Extension* mechanism. Thus, the *Generics* module has a deliberate impact on all thematic extension modules. |
<table>
<thead>
<tr>
<th>Module name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LandUse</td>
<td>The <em>LandUse</em> module allows for the representation of areas of the earth’s</td>
</tr>
<tr>
<td></td>
<td>surface dedicated to a specific land use.</td>
</tr>
<tr>
<td>Relief</td>
<td>The <em>Relief</em> module allows for the representation of the terrain in a city</td>
</tr>
<tr>
<td></td>
<td>model. CityGML supports terrain representations in different levels of</td>
</tr>
<tr>
<td></td>
<td>detail, reflecting different accuracies or resolutions. The terrain may be</td>
</tr>
<tr>
<td></td>
<td>specified as a regular raster or grid, as a TIN, by break lines, and by</td>
</tr>
<tr>
<td></td>
<td>mass points.</td>
</tr>
<tr>
<td>Transportation</td>
<td>The <em>Transportation</em> module is used to represent the transportation features</td>
</tr>
<tr>
<td></td>
<td>within a city, for example roads, tracks, railways, or squares. Transportation</td>
</tr>
<tr>
<td></td>
<td>features may be represented as a linear network or by geometrically</td>
</tr>
<tr>
<td></td>
<td>describing their 3D surfaces.</td>
</tr>
<tr>
<td>Vegetation</td>
<td>The <em>Vegetation</em> module provides thematic classes to represent vegetation</td>
</tr>
<tr>
<td></td>
<td>objects. CityGML’s vegetation model distinguishes between solitary vegetation</td>
</tr>
<tr>
<td></td>
<td>objects like trees, and vegetation areas which represent biotopes like forests</td>
</tr>
<tr>
<td></td>
<td>or other plant communities.</td>
</tr>
</tbody>
</table>
### Module name: WaterBody

<table>
<thead>
<tr>
<th>XML namespace identifier</th>
<th><a href="http://www.opengis.net/citygml/waterbody/1.0">http://www.opengis.net/citygml/waterbody/1.0</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Schema file</td>
<td>waterBody.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>wtr</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>WaterBody</em> module represents the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. It does, however, not inherit any hydrological or other dynamic aspects so far.</td>
</tr>
</tbody>
</table>

### Module name: TexturedSurface [deprecated]

<table>
<thead>
<tr>
<th>XML namespace identifier</th>
<th><a href="http://www.opengis.net/citygml/texturedsurface/1.0">http://www.opengis.net/citygml/texturedsurface/1.0</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>XML Schema file</td>
<td>texturedSurface.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>tex</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>TexturedSurface</em> module allows for assigning visual appearance properties (color, shininess, transparency) and textures to 3D surfaces. Due to inherent limitations of its modelling approach this module has been marked deprecated and is expected to be removed in future CityGML versions. Appearance information provided by this module can be converted to CityGML’s <em>Appearance</em> module without information loss. Thus, the use of the <em>TexturedSurface</em> module is strongly discouraged.</td>
</tr>
</tbody>
</table>

Tab. 4: Overview of CityGML’s core and thematic extensions modules.

#### 7.2 CityGML profiles

A CityGML profile is a combination of thematic extension modules in conjunction with the core module of CityGML. Each CityGML instance document shall employ the CityGML profile appropriate to the provided data. In general, two approaches to employ a CityGML profile within an instance document can be differentiated:

1. **CityGML profile definition embedded inline the CityGML instance document**
   A CityGML profile can be bound to an instance document using the `schemaLocation` attribute defined in the XML Schema instance namespace, http://www.w3.org/2001/XMLSchema-instance (commonly associated with the prefix `xsi`). The `xsi:schemaLocation` attribute provides a way to locate the XML Schema definition for namespaces defined in an XML instance document. Its value is a whitespace-delimited list of pairs of Uniform Resource Identifiers (URIs) where each pair consists of a namespace followed by the location of that namespace’s XML Schema definition, which is typically a .xsd file.

   By this means, the namespaces of the respective CityGML modules shall be defined within a CityGML instance document. The `xsi:schemaLocation` attribute then shall be used to provide the location to the respective XML Schema definition of each module. An example instance document following this first approach can be found in annex F.1.

2. **CityGML profile definition provided by a separate XML Schema definition file**
   The CityGML profile may also be specified by its own XML Schema file. This schema file shall combine the appropriate CityGML modules by importing the corresponding XML Schema definitions. For this purpose, the `import` element defined in the XML Schema namespace shall be used, http://www.w3.org/2001/XMLSchema (commonly associated with the prefix `xs`). For the `xs:import` element, the namespace of the imported CityGML module along with the location of the namespace’s XML Schema definition have to be declared. In order to apply a CityGML profile to an instance document, the profile’s schema has to be bound to the instance document using the `xsi:schemaLocation` attribute. The XML Schema file of the CityGML profile shall not contain any further content.
The targetNamespace of the profile’s schema shall differ from the namespaces of the imported CityGML modules. The namespace associated with the profile should be in control of the originator of the instance document and must be given as a previously unused and globally unique URI. The profile’s XML Schema file must be available (or accessible on the internet) to everybody parsing the associated CityGML instance document.

The second approach is illustrated by the following example XML Schema definition for the base profile of CityGML. Since the base profile is the union of all CityGML modules, the corresponding XML Schema definition imports each and every CityGML module. By this means, all components of the CityGML data model are available in and may be exchanged by instance documents referencing this example base profile. The schema definition file is shipped with the CityGML schema package and is accessible by the name CityGML.xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.citygml.org/citygml/profiles/base/1.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.citygml.org/citygml/profiles/base/1.0"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/citygml/landuse/1.0"
      schemaLocation="http://www.citygml.org/citygml/landuse/1.0/landUse.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/relief/1.0"
      schemaLocation="http://www.citygml.org/citygml/relief/1.0/relief.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/1.0"
      schemaLocation="http://www.citygml.org/citygml/transportation/1.0/transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/waterbody/1.0"
      schemaLocation="http://www.citygml.org/citygml/waterbody/1.0/waterBody.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/texturedsurface/1.0"
      schemaLocation="http://www.citygml.org/citygml/texturedsurface/1.0/texturedSurface.xsd"/>
</xs:schema>
```

The following excerpt of a CityGML dataset exemplifies how to apply the base profile schema CityGML.xsd to a CityGML instance document. The dataset contains two building objects and a city object group. The base profile defined by CityGML.xsd is referenced using the xsi:schemaLocation attribute of the root element. Thus, all CityGML modules are employed by the instance document and no further references to CityGML modules are necessary.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns="http://www.citygml.org/citygml/profiles/base/1.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:core="http://www.opengis.net/citygml/1.0"
    xmlns:buildg="http://www.opengis.net/citygml/building/1.0"
    xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/1.0"
    xmlns:gm="http://www.opengis.net/gml"
    xmlns:xAL="urn:oasis:names:tc:ciq:xsd:schema:xAL:2.0"
    xsi:schemaLocation="http://www.citygml.org/citygml/profiles/base/1.0 http://www.citygml.org/citygml/profiles/base/1.0/CityGML.xsd">
  <core:cityObjectMember>
    <buildg:Building gml:id="Build0815">
      <core:externalReference>
        <core:informationSystem>http://www.adv-online.de</core:informationSystem>
      </core:externalReference>
      <core:externalObject>
        <core:uri>urn:adv:oid:DEHE123400007001</core:uri>
      </core:externalObject>
      <core:externalReference>
        <buildg:identifier>1000</buildg:identifier>
        <buildg:yearOfConstruction>1985</buildg:yearOfConstruction>
        <buildg:roofType>1030</buildg:roofType>
        <buildg:measuredHeight uom="#m" n="8.0" buildg:measuredHeight>
          <buildg:storeysAboveGround n="2" buildg:storeysAboveGround>
```

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
<bldg:storeyHeightsAboveGround uom="m">2.5 2.5</bldg:storeyHeightsAboveGround>

<core:cityObjectMember>
  <bldg:Building gml:id="Build0817">
    ...
  </bldg:Building>
</core:cityObjectMember>

<core:cityObjectMember>
  <bldg:Building gml:id="Build0815">
    ...
  </bldg:Building>
</core:cityObjectMember>

<grp:CityObjectGroup gml:id="Complex113">
  <gml:name>Hotel complex 'Scenic View'</gml:name>
  <grp:function>building group</grp:function>
  <grp:groupMember xlink:href="#Build0817"/>
  <grp:groupMember xlink:href="#Build0815"/>
</grp:CityObjectGroup>
</core:cityObjectMember>
</core:CityModel>
8 Spatial model

Spatial properties of CityGML features are represented by objects of GML3’s geometry model. This model is based on the standard ISO 19107 ‘Spatial Schema’ (Herring 2001), representing 3D geometry according to the well-known Boundary Representation (B-Rep, cf. Foley et al. 1995). CityGML actually uses only a subset of the GML3 geometry package, defining a profile of GML3. This subset is depicted in Fig. 8 and Fig. 9. Furthermore, GML3’s explicit Boundary Representation is extended by scene graph concepts, which allow the representation of the geometry of features with the same shape implicitly and thus more space efficiently (chapter 8.2).

8.1 Geometric-topological model

The geometry model of GML 3 consists of primitives, which may be combined to form complexes, composite geometries or aggregates. For each dimension, there is a geometrical primitive: a zero-dimensional object is a Point, a one-dimensional a Curve, a two-dimensional a Surface, and a three-dimensional a Solid (Fig. 8). Each geometry can have its own coordinate reference system. A solid is bounded by surfaces and a surface by curves. In CityGML, a curve is restricted to be a straight line, thus only the GML3 class LineString is used. Surfaces in CityGML are represented by Polygons, which define a planar geometry, i.e. the boundary and all interior points are required to be located in one single plane.

Fig. 8: UML diagram of CityGML’s geometry model (subset and profile of GML3): Primitives and Composites.

Combined geometries can be aggregates, complexes or composites of primitives (see illustration in Fig. 10). In an Aggregate, the spatial relationship between components is not restricted. They may be disjoint, overlapping, touching, or disconnected. GML3 provides a special aggregate for each dimension, a MultiPoint, a MultiCurve, a MultiSurface or a MultiSolid (see Fig. 9). In contrast to aggregates, a Complex is topologically structured: its parts must be disjoint, must not overlap and are allowed to touch, at most, at their boundaries or share parts of their boundaries. A Composite is a special complex provided by GML3. It can only contain elements of the same dimension. Its elements must be disjoint as well, but they must be topologically connected along their boundaries. A Composite can be a CompositeSolid, a CompositeSurface, or CompositeCurve. (cf. Fig. 8).
An OrientableSurface is a surface with an explicit orientation, i.e. two sides, front and back, can be distinguished. This may be used to assign textures to specific sides of a surface, or to distinguish the exterior and the interior side of a surface when bounding a solid. Please note, that Curves and Surfaces have a default orientation in GML which results from the order of the defining points. Thus, an OrientableSurface only has to be used, if the orientation of a given GML geometry has to be reversed.

TriangulatedSurfaces are special surfaces, which specify triangulated irregular networks often used to represent the terrain. While a TriangulatedSurface is a composition of explicit Triangles, the subclass TIN is used to represent a triangulation in an implicit way by a set of control points, defining the nodes of the triangles. The triangulation may be reconstructed using standard triangulation methods (Delaunay triangulation). In addition, break lines and stop lines define contour characteristics of the terrain.

The GML3 composite model realises a recursive aggregation schema for every primitive type of the corresponding dimension. This aggregation schema allows the definition of nested aggregations (hierarchy of components). For example, a building geometry (CompositeSolid) can be composed of the house geometry (CompositeSolid) and the garage geometry (Solid), while the house’s geometry is further decomposed into the roof geometry (Solid) and the geometry of the house body (Solid).

CityGML provides the explicit modelling of topology, for example the sharing of geometry objects between features or other geometries. One part of space is represented only once by a geometry object and is referenced by all features or more complex geometries which are defined or bounded by this geometry object. Thus redundancy is avoided and explicit topological relations between parts are maintained. Basically, there are three cases. First, two features may be defined spatially by the same geometry. For example, if a path is both a transportation feature and a vegetation feature, the surface geometry defining the path is referenced both by the transportation object and by the vegetation object. Second, geometry may be shared between a feature and another geometry. A geometry defining a wall of a building may be referenced twice: by the solid geometry defining the geometry of the building, and by the wall feature. Third, two geometries may reference the same geometry, which is in the boundary of both. For example, a building and an adjacent garage may be represented by two solids. The surface describing the area where both solids touch may be represented only once and it is referenced by both solids. As
it can be seen from Fig. 11, this requires partitioning of the respective surfaces. In general, Boundary Representation only considers visible surfaces. However, to make topological adjacency explicit and to allow the possibility of deletion of one part of a composed object without leaving holes in the remaining aggregate touching elements are included. Whereas touching is allowed, permeation of objects is not in order to avoid the multiple representation of the same space. However, the use of topology in CityGML is optional.

In order to implement topology, CityGML uses the XML concept of XLinks provided by GML. Each geometry object that should be shared by different geometric aggregates or different thematic features is assigned an unique identifier, which may be referenced by a GML geometry property using a href attribute. CityGML does not deploy the built-in topology package of GML3, which provides separate topology objects accompanying the geometry. This kind of topology is very complex and elaborate. Nevertheless, it lacks flexibility when data sets, which might include or neglect topology, should be covered by the same data model. The XLink topology is simple and flexible and nearly as powerful as the explicit GML3 topology model. However, a disadvantage of the XLink topology is that navigation between topologically connected objects can only be performed in one direction (from an aggregate to its components), not (immediately) bidirectional as it is the case for GML’s built-in topology. An example for CityGML’s topology representation is given in the dataset listed in annex F.1.

![Recursive aggregation with arbitrary depth](image)

**Fig. 11:** Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

The following excerpt of a CityGML example file defines a gml:Polygon with an id *wallSurface4711*, which is part of the geometry property *lod2Solid* of a building. Another building being adjacent to the first building references this polygon in its geometry property.

```xml
<bldg:Building>
  ...
  <bldg:lod2Solid>
    ...
    <gml:surfaceMember>
      <gml:Polygon gml:id="wallSurface4711">
        <gml:exterior>
          <gml:LinearRing>
            <gml:pos srsDimension="3">32.0 31.0 2.5</gml:pos>
            ...
          </gml:LinearRing>
          <gml:exterior>
            ...
          </gml:Polygon>
          <gml:Polygone>
            <gml:PolygoneMember>
              <bldg:lod2Solid>
                ...
              </bldg:lod2Solid>
              ...
            </gml:PolygoneMember>
            <bldg:lod2Solid>
              ...
            </bldg:lod2Solid>
            ...
          </gml:PolygoneMember>
          <bldg:lod2Solid>
            ...
          </bldg:lod2Solid>
          ...
        </gml:exterior>
      </gml:Polygon>
    </gml:surfaceMember>
    ...
  </bldg:lod2Solid>
  ...
</bldg:Building>
```
8.2 Implicit geometries, prototypic objects, scene graph concepts

The concept of implicit geometries is an enhancement of the geometry model of GML3. It is, for example, used in CityGML’s vegetation model, for city furniture and generic objects (see chapters 10.6, 10.7 and 10.10). Implicit geometries may be applied to features from different thematic fields of CityGML in order to geometrically represent the features within a specific level of detail (LOD). Thus, each extension module may define spatial properties providing implicit geometries for its thematic classes. For this reason, the concept of implicit geometries is defined within the CityGML core module (cf. chapter 10.1). However, its description is drawn here since implicit geometries are part of CityGML’s spatial model. The UML diagram is depicted in Fig. 12. The corresponding XML schema definition is provided in annex A.1.

An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, for example a tree or other vegetation object, a traffic light or a traffic sign. This prototypic geometry object is reused or referenced many times, wherever the corresponding feature occurs in the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cartesian coordinate system), by a transformation matrix that is multiplied with each 3D coordinate of the prototype, and by an anchor point denoting the base point of the object in the world coordinate reference system. This reference point also defines the CRS to which the world coordinates belong after the application of the transformation. In order to determine the absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way even a projection might be modelled by the transformation matrix.

The reason for using the concept of implicit geometries in CityGML is space efficiency. Since the shape of, for example, trees of the same species can be treated as identical, it would be inefficient to model the detailed geometry of each of the large number of trees explicitly. The concept of implicit geometries is similar to the well known concept of primitive instancing used for the representation of scene graphs in the field of computer graphics (Foley et al. 1995).

The term implicit geometry refers to the principle that a geometry object with a complex shape can be simply represented by a base point and a transformation, implicitly unfolding the object’s shape at a specific location in the world coordinate system.

The shape of an ImplicitGeometry can be represented in an external file with a proprietary format, e.g. a VRML file, a DXF file, or a 3D Studio MAX file. The reference to the implicit geometry can be specified by an URI pointing to a local or remote file, or even to an appropriate web service. Alternatively, the shape can be defined by a GML3 geometry object. This has the advantage that it can be stored or exchanged inline within the CityGML dataset. Typically, the shape of the geometry is defined in a local coordinate system where the origin lies within or near to the object’s extent. If the shape is referenced by an URI, also the MIME type of the denoted object has to be specified (e.g. “model/vrml” for VRML models or “model/x3d+xml” for X3D models).

The implicit representation of 3D object geometry has some advantages compared to the explicit modelling, which represents the objects using absolute world coordinates. It is more space-efficient, and thus more extensive scenes can be stored or handled by a system. The visualisation is accelerated since 3D graphics cards support the scene graph concept. Furthermore, the usage of different shape versions of objects is facilitated, e.g. different seasons, since only the library objects have to be exchanged (see example in Fig. 40).
XML namespace

The XML namespace of the CityGML Core module defining the concept of implicit geometries is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/1.0. Within the XML Schema definition of the core module, this URI is also used to identify the default namespace.

ImplicitGeometryType, ImplicitRepresentationPropertyType

```xml
<xsd:complexType name=" ImplicitGeometryType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractGMLType">
      <xsd:sequence>
        <xsd:element name="mimeType" type="MimeTypeType" minOccurs="0"/>
        <xsd:element name="transformationMatrix" type="TransformationMatrix4x4Type" minOccurs="0"/>
        <xsd:element name="libraryObject" type="xs:anyURI" minOccurs="0"/>
        <xsd:element name="relativeGMLGeometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xsd:element name="referencePoint" type="gml:PointPropertyType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

8.2.1 External code lists

The ImplicitGeometry model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.4):

- MimeTypeType

8.2.2 Example CityGML datasets

An example for an implicit geometry is given by the following city furniture object (cf. chapter 10.7), which is represented by a geometry in LOD2:

```xml
<frn:CityFurniture>
  <frn:lod2ImplicitRepresentation>
    <core:ImplicitGeometry>
      <core:mimeType>model/vrml</core:mimeType>
      <core:libraryObject>
        http://www.some-3d-library.com/3D/furnitures/TrafficLight434.wrl
      </core:libraryObject>
      <core:referencePoint>
          <gml:pos srsDimension="3">5793898.77 3603845.54 44.8</gml:pos>
        </gml:Point>
      </core:referencePoint>
    </core:ImplicitGeometry>
    </frn:lod2ImplicitRepresentation>
  </frn:CityFurniture>
```

The shape of the geometry of the traffic light (city furniture with class “1000” and function “1080” according to the external code lists proposed in annex C.2) is defined by a VRML file which is specified by a URL. This
library object, which is defined in a local coordinate system, is transformed to its actual location by adding the coordinates of the reference point.

The following clip of a CityGML file provides a more complex example for an implicit geometry:

```
<frn:CityFurniture>
  <frn:class>1000</frn:class>
  <!-- “traffic”; declared in external code list (CityFurnitureClassType) in annex C.2 -->
  <frn:function>1080</frn:function>
  <!-- “traffic light”; declared in external code list (CityFurnitureFunctionType) in annex C.2 -->
  <frn:lod2ImplicitRepresentation>
    <core:ImplicitGeometry>
      <core:mimeType>model/vrml</core:mimeType>
      <core:transformationMatrix>
        0.866025 -0.5 0 0.7
        0.5 0.866025 0 0.8
        0 0 1 0
        0 0 0 1
      </core:transformationMatrix>
      <core:libraryObject>
        http://www.some-3d-library.com/3D/furnitures/TrafficLight434.wrl
      </core:libraryObject>
      <core:referencePoint>
          <gml:pos srsDimension="3">5793898.77 3603845.54 44.8</gml:pos>
        </gml:Point>
      </core:referencePoint>
    </core:ImplicitGeometry>
  </frn:lod2ImplicitRepresentation>
</frn:CityFurniture>
```

In addition to the first example, a transformation matrix is specified. It is a homogeneous matrix, serialized in a row major fashion, i.e. the first four entries in the list denote the first row of the matrix, etc. The matrix combines a translation by the vector (0.7, 0.8, 0) – the origin of the local reference system is not the center of the object – and a rotation around the z-axis by 30 degrees ($\cos(30) = 0.866025$ and $\sin(30) = 0.5$). This rotation is necessary to align the traffic light with respect to a road. The actual position of the traffic light is computed as follows:

1. each point of the VRML file (with homogeneous coordinates) is multiplied by the transformation matrix;
2. for each resulting point, the reference point $(5793898.77, 3603845.54, 44.8, 1)^T$ is added, yielding the actual geometry of the city furniture.

### 8.2.3 Conformance requirements

#### Base requirements

1. In order to geometrically represent a feature using the concept of implicit geometries, the corresponding thematic class of the feature shall define a spatial property of the type `ImplicitRepresentationPropertyType`. Thus, for all CityGML extension modules only the type `ImplicitRepresentationPropertyType` shall be used for spatial properties providing implicit geometries.

2. If the shape of an implicit geometry is referenced by an URI using the `libraryObject` property (type: `xs:anyURI`) of the element `ImplicitGeometry`, also the MIME type of the denoted object must be specified using the `mimeType` property (type: `MimeTypeType`).

#### Referential integrity

3. The type `ImplicitRepresentationPropertyType` may contain an `ImplicitGeometry` element inline or an XLink reference to a remote `ImplicitGeometry` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the corresponding property of type `ImplicitRepresentationPropertyType` may only point to a remote `ImplicitGeometry` element (where remote `ImplicitGeometry` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
9 Appearance model

In addition to spatial properties, CityGML features have *appearances*—observable properties of the feature’s surface. Appearances are not limited to visual data but represent arbitrary categories called *themes* such as infrared radiation, noise pollution, or earthquake-induced structural stress. Each LOD can have an individual appearance for a specific theme. An appearance is composed of data for each surface geometry object, i.e. *surface data*. A single surface geometry object may have surface data for multiple themes. Similarly, surface data can be shared by multiple surface geometry objects (e.g. road paving). Finally, surface data values can either be constant across a surface or depend on the exact location within the surface.

CityGML’s appearance model is defined within the extension module *Appearance* (cf. chapter 7). The UML diagram of the appearance model is illustrated in Fig. 13, for XML Schema definition see annex A.2.

Fig. 13: UML diagram of CityGML’s appearance model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML *Appearance* module.
In CityGML’s appearance model, themes are represented by an identifier only. The appearance of a city model for a given theme is defined by a set of Appearance objects referencing this theme. Thus, the Appearance objects belonging to the same theme compose a virtual group. They may be included in different places within a CityGML dataset. Furthermore a single CityGML dataset may contain several themes. An Appearance object collects surface data relevant for a specific theme either for individual features or the whole city model in any LOD. Surface data is represented by objects of class _SurfaceData and its descendants with each covering the whole area of a surface geometry object. The relation between surface data and surface geometry objects is expressed by an URI (Uniform Resource Identifier) link from a _SurfaceData object to an object of type gml:AbstractSurfaceType or type gml:MultiSurface.

A constant surface property is modelled as material. A surface property, which depends on the location within the surface, is modelled as texture. Each surface geometry object can have both a material and a texture per theme and side. This allows for providing both a constant approximation and a complex measurement of a surface’s property simultaneously. An application is responsible for choosing the appropriate property representation for its task (e.g. analysis or rendering). A specific mixing is not defined since this is beyond the scope of CityGML. If a surface geometry object is to receive multiple textures or materials, each texture or material requires a separate theme. The mixing of themes or their usage is not defined within CityGML and left to the application.

**XML namespace**

The XML namespace of the CityGML Appearance module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/appearance/1.0. Within the XML Schema definition of the Appearance module, this URI is also used to identify the default namespace.

### 9.1 Relation between appearances, features and geometry

Despite the close relation between surface data and surface, surface data is stored separately in the feature to preserve the original GML geometry model. Instead of surface data being an attribute of the respective target surface geometry object, each surface data object maintains a set of URIs specifying the gml:ids of the target surface geometry objects (of type gml:AbstractSurfaceType or gml:MultiSurface). In case of a composite or aggregate target surface, the surface data object is assigned to all contained surfaces. Other target types such as features, solids, or gml:AbstractSurfacePatchType (which includes gml:Triangle) are invalid, even though the XML schema language cannot formally express constrains on URI target types. For the exact mapping function of surface data values to a surface patch refer to the respective surface data type description.

The limitation of valid target types to gml:AbstractSurfaceType and gml:MultiSurface excluding gml:AbstractSurfacePatchType is based on the GML geometry model and its use in CityGML. In general, GML surfaces are represented using subclasses of gml:AbstractSurfaceType. Such surfaces are required to be continuous. A gml:MultiSurface does not need to fulfill this requirement and consequently is no gml:AbstractSurfaceType (cf. 8.1). Since captured real-world surfaces often cannot be guaranteed to be continuous, CityGML allows for gml:MultiSurface to represent a feature’s boundary in various places as an alternative to a continuous surface. To treat such surfaces similarly to a gml:CompositeSurface, surface data objects are allowed to link to gml:MultiSurface objects. The exclusion of gml:AbstractSurfacePatchType as valid target type results from its standard as a root class without gml:AbstractGMLType being its parent class. Thus, a gml:AbstractSurfacePatchType (which includes gml:Triangle and gml:Rectangle) cannot receive a gml:id and cannot be referenced.

Each surface geometry object can have per theme at most one active front-facing material, one active back-facing material, one active front-facing texture, and one active back-facing texture. If multiple surface data objects of the same category and theme are assigned to a surface geometry object, one is chosen to become active. Multiple indirect assignments due to nested surface definitions are resolved by overwriting, e.g. the front-facing material of a gml:Polygon becomes active by overwriting the front-facing material of the parental gml:CompositeSurface. Multiple direct assignments, i.e. a surface geometry object’s gml:id is referenced multiple times within a theme, are not allowed and are resolved implementation-dependently by choosing exactly one of the conflicting surface data objects. Thus, multiple direct assignments within a theme need to be avoided.

Each CityObject feature can store surface data. Thus, surface data is arranged in the feature hierarchy of a CityGML dataset. Surface data then links to its target surface using URIs. Even though the linking mechanism permits arbitrary links across the feature hierarchy to another feature’s surface, it is recommended to follow the
principle of locality: Surface data should be stored such that the linked surfaces only belong to the containing CityObject feature and its children. “Global” surface data should be stored with the city model. Adhering to the locality principle also ensures that CityObjects retrieved from a WFS will contain the respective appearance information.

9.2 Appearance and SurfaceData

The feature class Appearance defines a container for surface data objects. It provides the theme that all contained surface data objects are related to. All appearance objects with the same theme in a CityGML file are considered a group. Surface data objects are stored in the surfaceDataMember property. They can be used in multiple themes simultaneously as remote properties.

The feature class _SurfaceData is the base class for materials and textures. Its only element is the boolean flag isFront, which determines the side a surface data object applies to. Please note, that all classes of the appearance model support CityGML’s ADE mechanism (cf. chapters 6.12 and 10.11). The hooks for application specific extensions are realized by the elements “_GenericApplicationPropertyOf…”.

**AppearanceType, AppearancePropertyType**

```xml
<xs:complexType name="AppearanceType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="theme" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAppearance" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

**appearanceMember, appearance**

```xml
<xs:element name="appearanceMember" type="AppearancePropertyType" substitutionGroup="gml:featureMember"/>
<xs:element name="appearance" type="AppearancePropertyType" substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>
```

The definition of appearanceMember allows for an arbitrary or even mixed sequence of CityObject features and Appearance features within a CityModel feature collection (cf. chapter 10.1).

In order to store appearance information within a single CityObject feature, the corresponding abstract class _CityObject of the core module is augmented by the property element appearance. The additional property appearance is injected into _CityObject using CityGML’s Application Domain Extension mechanism (cf. chapter 10.11). By this means, each thematic subclass of _CityObject inherits this property. Thus, the Appearance module has a deliberate impact on each extension module defining thematic subclasses of _CityObject.

**AbstractSurfaceDataType, _SurfaceData, SurfaceDataPropertyType**

```xml
<xs:complexType name="AbstractSurfaceDataType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType"/>
  </xs:complexContent>
</xs:complexType>
```
9.3 Material

Materials define light reflection properties being constant for a whole surface geometry object. The definition of the class X3DMaterial is adopted from the X3D and COLLADA specification (cf. X3D, COLLADA specification). diffuseColor defines the color of diffusely reflected light. specularColor defines the color of a directed reflection. emissiveColor is the color of light generated by the surface. All colors use RGB values with red, green, and blue between 0 and 1. Transparency is defined separately using the transparency element where 0 stands for fully opaque and 1 for fully transparent. ambientIntensity defines the minimum percentage of diffuseColor that is visible regardless of light sources. shininess controls the sharpness of the specular highlight. 0 produces a soft glow while 1 results in a sharp highlight. isSmooth gives a hint for normal interpolation. If this boolean flag is set to true, vertex normals should be used for shading (Gouraud shading). Otherwise, normals should be constant for a surface patch (flat shading).

Target surfaces are specified using target elements. Each element contains the URI of one target surface geometry object (of type gml:AbstractSurfaceType or gml:MultiSurface).

X3DMaterialType, X3DMaterial

9.4 Texture and texture mapping

The abstract base class for textures is Texture. Textures in CityGML are always raster-based 2D textures. The raster image is specified by imageURI using a URI and can be an arbitrary image data resource, even a preformatted request for a web service. The image data format can be defined using standard MIME types in the mimeType element.
Textures can be qualified by the attribute `textureType`. The `textureType` differentiates between textures, which are specific for a certain object (`specific`) and prototypic textures being typical for that object surface (`typical`). Textures may also be classified as `unknown`.

The specification of texture wrapping is adopted from the COLLADA standard. Texture wrapping is required when accessing a texture outside the underlying image raster. `wrapMode` can have one of five values (Fig. 14 illustrates the effect of these wrap modes):

1. `none` – the resulting color is fully transparent
2. `wrap` – the texture is repeated
3. `mirror` – the texture is repeated and mirrored
4. `clamp` – the texture is clamped to its edges
5. `border` – the resulting color is specified by the `borderColor` element (RGBA)

In wrap mode `mirror`, the texture image is repeated both in horizontal and in vertical direction to fill the texture space similar to wrap mode `wrap`. Unlike `wrap`, each repetition results from flipping the previous texture part along the repetition direction. This behaviour removes the edge correspondence constraint for wrapped textures and always results in a seamless texture.

![Fig. 14: A texture (a) applied to a facade using different wrap modes: (b) none, (c) wrap, (d) mirror, (e) clamp and (f) border. The border color is red. The numbers denote texture coordinates (image: Hasso-Plattner-Institute).](image)

AbstractTextureType, _Texture, WrapModeType, TextureTypeType

```xml
<xs:complexType name="AbstractTextureType">
  <xs:complexContent>
    <xs:extension base="AbstractSurfaceDataType">
      <xs:sequence>
        <xs:element name="imageURI" type="xs:anyURI" minOccurs="0"/>
        <xs:element name="mimeType" type="core:MimeTypeType" minOccurs="0"/>
        <xs:element name="textureType" type="TextureTypeType" minOccurs="0"/>
        <xs:element name="wrapMode" type="WrapModeType" minOccurs="0"/>
        <xs:element name="borderColor" type="ColorPlusOpacity" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfTexture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
_Texture is further specialised according to the texture parameterisation, i.e. the mapping function from a location on the surface to a location in the texture image. CityGML uses the notion of texture space, where the texture image always occupies the region \([0,1]^2\) regardless of the actual image size or aspect ratio. The lower left image corner is located at the origin. The mapping function must be known for each surface geometry object to receive texture.

Fig. 15: A georeferenced texture applied to ground and roof surfaces (source: Senate of Berlin, Hasso-Plattner-Institute).

The class GeoreferencedTexture describes a texture that uses a planimetric projection. Consequently, it does not make sense to texture vertical surfaces using a GeoreferencedTexture. Such a texture has a unique mapping function which is usually provided with the image file (e.g. georeferenced TIFF) or as a separate ESRI world file. The search order for an external georeference is determined by the boolean flag preferWorldFile. If this flag is set to true (its default value), a world file is looked for first and only if it is not found the georeference from the image data is used. If preferWorldFile is false, the world file is used only if no georeference from the image data is available.

Alternatively, CityGML allows for inline specification of a georeference similar to a world file. This internal georeference specification always takes precedence over any external georeference. referencePoint defines the location of the center of the upper left image pixel in world space and corresponds to values 5 and 6 in an ESRI world file. Since GeoreferencedTexture uses a planimetric projection, referencePoint is two-dimensional. orientation defines the rotation and scaling of the image in form of a 2x2 matrix (a list of 4 doubles in row-major order corresponding to values 1, 3, 2, and 4 in an ESRI world file). The CRS of this transformation is identical to the referencePoint’s CRS. A planimetric point \((x, y)\) in that CRS is transformed to a point \((s, t)\) in texture space using the formula:

\[
\begin{bmatrix}
  s \\
  t
\end{bmatrix} = \begin{bmatrix}
  1/w & 0 \\
  0 & -1/h
\end{bmatrix} \cdot M^{-1} \cdot \begin{bmatrix}
  x \\
  y
\end{bmatrix} - P_R + \begin{bmatrix}
  0 \\
  1
\end{bmatrix}
\]

with \(M\) denoting orientation, \(P_R\) denoting referencePoint., \(w\) the image’s width in pixels, and \(h\) the image’s height in pixels.

If neither an internal nor an external georeference is given the GeoreferencedTexture is invalid. Each target surface geometry object is specified by an URI in a target element. All target surface geometry objects share the mapping function defined by the georeference. No other mapping function is allowed. Please note, that the gml:boundedBy property inherited from gml:AbstractFeatureType could be set to the bounding box of valid image data to allow for spatial queries. Fig. 15 shows a georeferenced texture applied to the ground and all roof surfaces.
GeoreferencedTextureType, GeoreferencedTexture

The class ParameterizedTexture describes a texture with target-dependent mapping function. The mapping is defined by subclasses of class _TextureParameterization as a property of the link to the target surface geometry object. Each target surface geometry object is specified as URI in the uri attribute of a separate target element. Since target implements gml:AssociationAttributeGroup, it allows referencing to a remote _TextureParameterization object (using the xlink:href attribute), e.g. for sharing a mapping function between targets or textures in different themes. The mapping function can either use the concept of texture coordinates (through class TexCoordList) or a transformation matrix from world space to texture space (through class TexCoordGen).

Texture coordinates are applicable only to polygonal surfaces, whose boundaries are described by gml:LinearRing (e.g., gml:Triangle, gml:Polygon, or a gml:MultiSurface consisting of gml:Polygons). They define an explicit mapping of a surface’s vertices to points in texture space, i.e. each vertex including interior ring vertices must receive a corresponding coordinate pair in texture space (for the notion of coordinates, refer to ISO 19111). These coordinates are not restricted to the [0,1] interval. Texture coordinates for interior surface points are planarly interpolated from the vertices’ texture coordinates. Fig. 16 shows an example.

Texture coordinates for a target surface geometry object are specified using class TexCoordList as a texture parameterization object in the texture’s target property. Each exterior and interior gml:LinearRing composing the boundary of the target surface geometry object (which also might be a gml:CompositeSurface, gml:MultiSurface, or gml:TriangulatedSurface) requires its own set of texture coordinates. A set of texture coordinates is specified using the textureCoordinates element of class TexCoordList. Thus, a TexCoordList contains as many textureCoordinate elements as the target surface geometry object contains gml:LinearRings. textureCoordinate’s mandatory attribute ring provides the gml:id of the respective ring. The content is an ordered list of double values where each two values define a \((s, t)\) texture coordinate pair with \(s\) denoting the horizontal and \(t\) the vertical texture axis. The list contains one pair per ring point with the pairs’ order corresponding to the ring points’ order. If any ring point of a target surface geometry object has no texture coordinates...
assigned, the mapping is incomplete and the respective surface cannot be textured. In case of aggregated target geometry objects, mapping completeness is determined only for leaf geometry objects.

Fig. 17: Projecting a photograph (a) onto multiple facades (b) using the `worldToTexture` transformation. The photograph does not cover the left facade completely. Thus, the texture appears to be clipped. Texture wrapping is set to "none" (source: Senate of Berlin, Hasso-Plattner-Institute).

Alternatively, the mapping function can comprise a 3x4 transformation matrix specified by class `TexCoordGen`. The transformation matrix, specified by the `worldToTexture` element, defines a linear transformation from a spatial location in homogeneous coordinates to texture space. The use of homogeneous coordinates facilitates perspective projections as transformation, e.g. for projecting a photograph into a city model (cf. Fig. 17). Texture coordinates \((s, t)\) are calculated from a space location \((x, y, z)\) as \((s, t) = (s', t', q')\) with \((s', t', q') = M \cdot (x, y, z, 1)^\top\). \(M\) denotes the 3x4 transformation matrix. Compared to a general 4x4 transformation, the resulting \(z\) component is ignored. Thus, the respective matrix row is omitted. Additionally, the `worldToTexture` element uses the `gml:SRSReferenceGroup` attributes to define its CRS. A location in world space has to be first transformed into this CRS before the transformation matrix can be applied.

The following construction results in a `worldToTexture` transformation that mimics the process of taking a photograph by projecting a location in world space (in the city model) to a location in texture space:

\[
M = \begin{bmatrix} 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2f/w & 0 & 0 & 0 \\ 0 & 2f/h & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ d_x & d_y & d_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

In this formula, \(f\) denotes the focal length; \(w\) and \(h\) represent the image sensor’s physical dimensions; \(\vec{r}, \vec{u}, \vec{d}\) define the camera’s frame of reference as right, up and directional unit vectors expressed in world coordinates; and \(P\) stands for the camera’s location in world space. Fig. 18 sketches this setting.

Fig. 18: Projective texture mapping. All points on a ray \(R\) starting from the projection center \(c\) are mapped to the same point \(T\) in texture space (image: Hasso-Plattner-Institute, IGG TU Berlin).
Alternatively, if the 3x4 camera matrix $M_f$ is known (e.g. through a calibration and registration process), it can easily be adopted for use in worldToTexture. $M_f$ is derived from intrinsic and extrinsic camera parameters (interior and exterior orientation) and transforms a location in world space to a pixel location in the image. Assuming the upper left image corner has pixel coordinates (0,0), the complete transformation to texture space coordinates can be written as (width_image and height_image denote the image size in pixels):

$$M = \begin{bmatrix}
\frac{1}{\text{width}_\text{image}} & 0 & 0 \\
0 & -\frac{1}{\text{height}_\text{image}} & 1 \\
0 & 0 & 1
\end{bmatrix} \cdot M_f$$

Please note, that worldToTexture cannot compensate for radial or other non-linear distortions introduced by a real camera lens.

Another use of worldToTexture is texturing a facade with complex geometry without specifying texture coordinates for each gml:LinearRing. Instead, only the facade’s aggregated surface becomes the texture target using a TexCoordGen as parameterization. Then, worldToTexture effectively encodes an orthographic projection of world space into texture space. For the special case of a vertical facade this transformation is given by:

$$M = \begin{bmatrix}
\frac{1}{\text{width}_f} & 0 & 0 & 0 \\
0 & \frac{1}{\text{height}_f} & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
-n_x & n_y & 0 & 0 \\
0 & 0 & 1 & 0 \\
n_x & n_y & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 0 & -F_z \\
0 & 1 & 0 & -F_y \\
0 & 0 & 1 & -F_z \\
0 & 0 & 0 & 1
\end{bmatrix}$$

This equation assumes $\vec{n}$ denoting the facade’s overall normal vector (normalized, pointing outward, and being parallel to the ground), $F$ denoting the facade’s lower left point, and width_f and height_f specifying the facade’s dimensions in world units. For the general case of an arbitrary normal vector the facade orientation matrix assumes a form similar to the camera orientation matrix:

$$M = \begin{bmatrix}
\frac{1}{\text{width}_f} & 0 & 0 & 0 \\
0 & \frac{1}{\text{height}_f} & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
\vec{r}_x & \vec{r}_y & \vec{r}_z & 0 \\
\vec{u}_x & \vec{u}_y & \vec{u}_z & 0 \\
\vec{n}_x & \vec{n}_y & \vec{n}_z & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 0 & -F_z \\
0 & 1 & 0 & -F_y \\
0 & 0 & 1 & -F_z \\
0 & 0 & 0 & 1
\end{bmatrix}$$

$$\vec{f} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \vec{n}$$

$$\vec{u} = \vec{n} \times \vec{f}$$

---

ParameterizedTextureType, ParameterizedTexture, TextureAssociationType
AbstractTextureParameterizationType, TexCoordListType, TexCoordGenType

9.5 Related concepts

The notion of appearance clearly relates to the generic coverage approach (cf. ISO 19123 and OGC Abstract specification, Topic 6). Surface data can be described as discrete or continuous coverage over a surface as two-dimensional domain with a specific mapping function. Such an implementation requires the extension of GML coverages (as of version 3.1) by suitable mapping functions and specialization for valid domain and range sets. For reasons of simplicity and comprehensibility both in implementation and usage, CityGML does not follow this approach, but relies on textures and materials as well-known surface property descriptions from the field of
computer graphics (cf. X3D, COLLADA specification, Foley et al.). Textures and materials store data as color using an appropriate mapping. If such a mapping is impractical, data storage can be customised using ADEs. A review of coverages for appearance modelling is considered for CityGML beyond version 1.0.0.

Appearance is also related to portrayal. Portrayal describes the composition and symbolisation of a digital model’s image, i.e. presentation, while appearance encodes observations of the real object’s surface, i.e. data. Even though being based on graphical terms such as textures and materials, surface data is not limited to being input for portrayal, but similarly serves as input or output for analyses on a feature’s surface. Consequently, CityGML does not define mixing or composition of themes for portrayal purposes. Portrayal is left to viewer applications or styling specification languages such as OGC Styled Layer Descriptors (SLD) or OGC Symbol Encoding (SE).

9.6 Conformance requirements

Base requirements

1. A surface geometry object may be the target of at most two textures and two materials (one for front and back respectively) per theme.

2. The referencePoint property (type: gml:PointPropertyType) of the element GeoreferencedTexture may only contain or reference a point geometry object with 2D coordinate values.

3. Texture coordinates given by the textureCoordinates property of the element TexCoordList define an explicit mapping of a surface’s boundary points to points in texture space. Each boundary point of the surface must receive a corresponding coordinate pair in texture space. The coordinate pair in texture space shall be given as two doubles per boundary point. The order of the coordinate pairs must follow the order of the boundary points. Each gml:LinearRing composing the boundary of the target surface geometry object requires its own set of texture coordinates.

4. A GeoreferencedTexture element must provide either internal or external georeference, otherwise it is invalid. Internal georeference shall be declared by the referencePoint property (type: gml:PointPropertyType) and the orientation property (type: core:TransformationMatrix2x2Type) of the element GeoreferencedTexture. External georeference may be provided by the texture image file itself (e.g. GeoTIFF) or by an accompanying world file.

Referential integrity

5. The appearanceMember element (type: AppearancePropertyType) may contain an Appearance element inline or an XLink reference to a remote Appearance element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the appearanceMember element may only point to a remote Appearance element (where remote Appearance elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

6. The appearance property (type: AppearancePropertyType) of the element core:_CityObject may contain an Appearance element inline or an XLink reference to a remote Appearance element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the appearance property may only point to a remote Appearance element (where remote Appearance elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

7. The surfaceDataMember property (type: SurfaceDataPropertyType) of the element Appearance may contain a _SurfaceData element inline or an XLink reference to a remote _SurfaceData element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the surfaceDataMember property may only point to a remote _SurfaceData element (where remote _SurfaceData elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

8. The target property (type: TextureAssociationType) of the element ParameterizedTexture may contain a _TextureParameterization element inline or an XLink reference to a remote _TextureParameterization element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the target
property may only point to a remote \_TextureParameterization element (where remote \_TextureParameterization elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

9. The target property (type xs:anyURI) of the element GeoreferencedTexture shall specify the gml:id of the target surface geometry object which may only be of type gml:AbstractSurfaceType or gml:MultiSurface.

10. The uri attribute of the complex type TextureAssociationType shall specify the gml:id of the target surface geometry object which may only be of type gml:AbstractSurfaceType or gml:MultiSurface.

11. The ring attribute of the textureCoordinates property of the element TexCoordList shall specify the gml:id of the target surface geometry object which may only be of type gml:LinearRing.

12. The target property (type xs:anyURI) of the element X3DMaterial shall specify the gml:id of the target surface geometry object which may only be of type gml:AbstractSurfaceType or gml:MultiSurface.

9.7 Material model of previous CityGML versions [deprecated]

Since GML3 has no built-in concept for the representation of surface materials, previous versions of CityGML extend the GML3 geometry model by the class TexturedSurface, which allows for assigning appearance properties (colors, shininess, transparency) and textures to 3D surfaces. The definition of the appearance properties is adopted from the X3D specification. This approach for appearance modelling has been deprecated due to inherent limitations. However, in order to provide a certain degree of backwards compatibility for already existing CityGML implementations, the approach has been incorporated into CityGML version 1.0 as a separate extension module called TexturedSurface. By this means, implementations may employ the old material model by supporting this module. Please note, that appearance information modelled according to the TexturedSurface module can be converted without information loss to the concepts provided by CityGML’s Appearance module that has been introduced in the previous clauses of this chapter. Thus, the use of the TexturedSurface module is strongly discouraged and implementations should only stick to the Appearance module instead. Moreover, the TexturedSurface module is expected to be removed in future versions of CityGML.

For the TexturedSurface module, each surface or composite surface can be specialized to a TexturedSurface, which can be assigned Materials (colors, shininess, transparency) or SimpleTextures. Fig. 19 depicts the UML diagram, for XML schema definition see annex A.12.

---

Fig. 19: UML diagram of CityGML’s material model. Please note, that this approach for appearance modelling has been marked as deprecated and is expected to be removed in future CityGML versions. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML TexturedSurface module.
The concept of positioning textures on surfaces complies with the 3D computer graphics standard X3D (web 3D 2004), a successor of VRML97. CityGML adds the class TexturedSurface to the geometry model of GML3 because there has been no appropriate texturing concept in ISO 19107 and in GML3.

A texture is specified as a raster image referenced by an URI (Uniform Resource Identifier) and can be an arbitrary resource, even on the internet. Textures are positioned by employing the concept of texture coordinates, i.e. each texture coordinate matches with exactly one 3D coordinate of the TexturedSurface (Fig. 16). The use of texture coordinates allows an exact positioning and trimming of the texture on the surface geometry.

The color of a surface is defined by RGB values. These have to be in the range of 0 to 1. The frontOpacity and the backOpacity define the level of transparency of each surface. Their values have also to be in the range of 0 to 1, where 1 means completely covering and 0 denotes a completely transparent surface. The colors can be differentiated in diffuseColor (color when illuminated by a source of light), emissiveColor (color when self-illuminating) and specularColor/shininess (color for shiny surfaces).

Textures can be qualified by the attribute textureType. The textureType differentiates between textures which are specific for a certain object (specific) and prototypic textures being typical for that object surface (typical). Textures may also be classified as unknown.

.Appearance is derived from gml:AbstractGMLType to be referenced in an appearance property. The attribute gml:id is inherited, whose value may be referenced by a XLink. _Appearance is the upper class of Material and SimpleTexture.

XML namespace

The XML namespace of the CityGML TexturedSurface module is identified by the Uniform Resource Identifier (URI) http://www.opengeospatial.org/citygml/texturedsurface/1.0. Within the XML Schema definition of the TexturedSurface module, this URI is also used to identify the default namespace.

9.7.1 Textured surfaces

TexturedSurfaceType, TexturedSurface, AppearancePropertyType

```
<xs:complexType name="TexturedSurfaceType">
  <xs:complexContent>
    <xs:extension base="gml:OrientableSurfaceType">
      <xs:complexContent>
        <xs:element name="appearance" ref="gml:AppearancePropertyType" maxOccurs="unbounded" />
        <xs:extensionGroup name="AppearancePropertyType" substitutionGroup="gml:AbstractGMLType" />
      </xs:complexContent>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

TexturedSurface may have one or more appearance properties, which can either be a Material (Color,...) or a Texture. The _Appearance element can either be represented inline as an element of this type or by an XLink reference to a remote _Appearance element. Either the reference or the contained element must be given, but neither both nor none. The side of the surface the _Appearance refers to is given by the orientation attribute of the appearance property element, which refers to the corresponding sign attribute of the orientable surface: + means the side with positive orientation and - the side with negative orientation.
AbstractAppearanceType, _Appearance

```xml
<x:simpleType name="AbstractAppearanceType" abstract="true">
  <xs:extension base="gml:AbstractGMLType"/>
</xs:simpleType>
```

MaterialType, Material

```xml
<x:simpleType name="MaterialType">
  <xs:extension base="AbstractAppearanceType"/>
</xs:simpleType>
```

SimpleTextureType, SimpleTexture, TextureTypeType

```xml
<x:simpleType name="SimpleTextureType">
  <xs:extension base="AbstractAppearanceType"/>
</xs:simpleType>
```

9.7.2 Conformance requirements

Referential integrity

- The `appearance` property (type: `AppearancePropertyType`) of the element `TexturedSurface` may contain an `_Appearance` element inline or an XLink reference to a remote `_Appearance` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `appearance` property may only point to a remote `_Appearance` element (where remote `_Appearance` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10 Thematic model

The thematic model of CityGML consists of the class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. Most thematic classes are (transitively) derived from the basic classes Feature and FeatureCollection, the basic notions defined in ISO 19109 and GML3 for the representation of spatial objects and their aggregations. Features contain spatial as well as non-spatial attributes which are mapped to GML3 feature properties with corresponding data types. Geometric properties are represented as associations to the geometry classes described in chapter 8. The thematic model also comprises different types of interrelationships between Feature classes like aggregations, generalisations and associations.

The aim of the explicit modelling is to reach a high degree of semantic interoperability between different applications. By specifying the thematic concepts and their semantics along with their mapping to UML and GML3 different applications can rely on a well-defined set of Feature types, attributes and data types with a standardised meaning or interpretation. In order to also allow for the exchange of objects and/or attributes that are not explicitly modelled in CityGML, the concepts of generic city objects and attributes as well as CityGML’s Application Domain Extension mechanism have been introduced (cf. chapter 10.10 and chapter 10.11).

Each field of CityGML’s thematic model is covered by a separate CityGML extension module. Thus, the extension modules are derived by vertically slicing the overall thematic data model of CityGML. All extension modules are based on and dependent from the CityGML core module. The core comprises the basic concepts and components of the CityGML data model. Implementations may choose to combine CityGML extension modules in conjunction with the core according to their specific information needs or application domain. As for version 1.0 of CityGML, the following eleven thematic extension modules are defined: Appearance, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Vegetation, WaterBody, and TexturedSurface [deprecated]. Valid combinations of CityGML modules are called CityGML profiles. By this means, CityGML profiles explicitly allow for partial implementations of the overall CityGML data model (cf. chapter 7).

The thematic fields covered by the CityGML data model are introduced within the sub clauses of this chapter. Each sub clause is related to a specific CityGML module.
10.1 CityGML Core

The CityGML Core module defines the basic concepts and components of the overall CityGML data model. It forms the universal lower bound of the CityGML data model and, thus, is a dependency of all extension modules. Consequently, the core module has to be implemented by any conformant system. Primarily, the core module provides the abstract base classes from which thematic classes within extension modules are (transitively) derived. Besides abstract type definitions, the core also contains non-abstract content, for example basic data types and thematic classes that may be used by more than one extension module. The UML diagram in Fig. 20 illustrates CityGML’s core module, for XML Schema definition see below and annex A.1.

The base class of all thematic classes within CityGML’s data model is the abstract class _CityObject. _CityObject provides a creation and a termination date for the management of histories of features as well as the possibility to model external references to the same object in other data sets. _CityObject is a subclass of the GML class Feature, thus it inherits the metadata property (which can be e.g. information about the lineage, quality aspects, accuracy) and name property from the superclass _GML. A CityObject may have multiple names, which are optionally qualified by a codeSpace. This enables the differentiation between, for example, an official name and a popular name or of names in different languages (cf. the name property of GML objects, Cox et al. 2004). The generalisation property generalizesTo of _CityObject may be used to relate features, which represent the same real-world object in different Levels-of-Detail, i.e. a feature and its generalised counterpart(s). The direction of this relation is from the feature to the corresponding generalised feature.

Thematic classes may have further subclasses with relations, attributes and geometry. Features of the specialized subclasses of _CityObject may be aggregated to a single CityModel, which is a feature collection with optional metadata. Generally, each feature has the attributes class, function, and usage, unless it is stated otherwise. The class attribute can occur only once, while the attributes usage and function can be used multiple times. The class attribute allows for the classification of features beyond the thematic class hierarchy of _CityObject. For example, a building feature is represented by the thematic subclass bldg:Building of _CityObject in the first place (this subclass is defined within CityGML’s Building module, cf. chapter 10.3). A further classification, e.g. as residential or administration building, may then be modelled using the class attribute of the class bldg:Building. The
attribute function denotes the intended purpose or usage of the object, such as hotel or shopping centre for a building, while the attribute usage defines its real or actual usage. Possible values for the attributes class, function, and usage are specified in external code lists following the Simple Dictionary Profile of GML 3.1.1 (cf. chapter 6.6). CityGML provides predefined external code lists containing feasible attribute values (cf. Annex C). However, these code lists are proposals and may be extended or redefined by users.

In addition to thematic content, the core module also provides the concept of implicit geometries as an enhancement of the geometry model of GML3. Since this concept is strongly related to the spatial model of CityGML it has already been introduced in chapter 8.2.

The top level class hierarchy of the thematic model in CityGML is presented in Fig. 21. The subclasses of _CityObject comprise the different thematic fields of a city model covered by separate CityGML extension modules: the terrain, buildings, the coverage by land use objects, water bodies, vegetation, generic city objects, city furniture objects, city object groups, and transportation. To indicate the extension module defining a respective subclass of _CityObject, the class names in Fig. 21 are preceded with prefixes. Each prefix is associated with one CityGML extension module (see chapter 4.3 and chapter 7 for a list of CityGML’s extension modules and the corresponding prefixes).

The classes GenericCityObject and _GenericAttribute defined within CityGML’s Generics module (cf. chapters 6.11 and 10.10) allow for modelling and exchanging of 3D objects which are not covered by any other thematic class or which require attributes not represented in CityGML. For example, in the future, sites derived from the abstract class _Site of the core module will be completed by further subclasses like tunnel, bridge, excavation, wall or embankment. At present, the class GenericCityObject may be used in order to represent and exchange these features. However, the concept of generic city objects and attributes may only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

If the Generics module is employed, each CityObject may be assigned an arbitrary number of generic attributes in order to represent additional properties of features. For this purpose, the Generics module augments the abstract base class _CityObject by the property element _genericAttribute. The additional property _genericAttribute is injected into _CityObject using CityGML’s Application Domain Extension mechanism (cf. chapter 10.11). By this means, each thematic subclass of _CityObject inherits this property and, thus, the possibility to contain generic attributes. Therefore, the Generics module has a deliberate impact on all CityGML extension modules defining thematic subclasses of _CityObject.
Appearance information about a feature’s surfaces can be represented by the class Appearance provided by CityGML’s Appearance module (cf. chapter 9). In contrast to the other thematic extensions to the core, Appearance is not derived from CityObject but from the GML class Feature. CityObject features and Appearance features may be embraced within a single CityModel feature collection in an arbitrary or even mixed sequence using the cityObjectMember and appearanceMember elements, both being members of the substitution group gml:featureMember (cf. chapter 9 and chapter 10.1.1). Furthermore, feature appearances may be stored inline the CityObject itself. In order to enable CityObjects to store appearance information, the Appearance module augments the abstract base class _CityObject by the property element appearance using CityGML’s Application Domain Extension mechanism (cf. chapter 10.11). Consequently, the appearance property is only available for _CityObject and its thematic subclasses if the Appearance module is supported. Therefore, like the Generics module, the Appearance module has a deliberate impact on any other extension module.

For sake of completeness, the class TexturedSurface is also illustrated in Fig. 21. This approach of appearance modelling of previous versions of CityGML has been deprecated and is expected to be removed in future CityGML versions. Since the information covered by TexturedSurface can be losslessly converted to the Appearance module, the use of TexturedSurface is strongly discouraged.

XML namespace

The XML namespace of the CityGML Core module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/1.0. Within the XML Schema definition of the core module, this URI is also used to identify the default namespace.

10.1.1 Base elements

AbstractCityObjectType, _CityObject

<xs:complexType name="AbstractCityObjectType" abstract="true">
  <xs:complexContent>
    <xs:restriction base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="creationDate" type="xs:date" minOccurs="0"/>
        <xs:element name="terminationDate" type="xs:date" minOccurs="0"/>
        <xs:element name="externalReference" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfCityObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

CityModelType, CityModel

<xs:complexType name="CityModelType">
  <xs:complexContent base="gml:AbstractFeatureCollectionType">
    <xs:sequence>
      <xs:element ref="_GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>
The abstract class _Site is intended to be the superclass for buildings, facilities, etc. Future extension of CityGML (e.g. bridges or tunnels) would be modelled as subclasses of _Site. As subclass of _CityObject, a _Site inherits all attributes and relations, in particular the id, names, external references, and generalisation relations.

### 10.1.2 Generalisation relation

GeneralizationRelationType

### 10.1.3 External references

An ExternalReference defines a hyperlink from a CityObject to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS MasterMap®. The reference consists of the name of the external information system, represented by an URI, and the reference of the external object, given either by a string or by an URI. If the informationSystem element is missing in the ExternalReference, the ExternalObjectReference must be an URI.

ExternalReferenceType, ExternalObjectReferenceType
10.1.4 Address information

The CityGML core module provides the means to represent address information of real-world features within virtual city models. Since not every real-world feature is assigned an address, a correspondent address property is not defined for the base class _CityObject, but has to explicitly be modelled for a thematic subclass. For example, the building model declares address properties for its classes _AbstractBuilding and Door. Both classes are referencing the corresponding data types of the core module to represent address information (cf. chapter 10.3).

Addresses are modelled as GML features having one address property and an optional multiPoint property. For example, for a building feature the multiPoint property allows for the specification of the exact positions of the building entrances that are associated with the corresponding address. The point coordinates can be 2D or 3D. Modelling addresses as features has the advantage that GML3’s method of representing features by reference (using XLinks) can be applied. This means, that addresses might be bundled as an address FeatureCollection that is stored within an external file or that can be served by an external Web Feature Service. The address property elements within the CityGML file then would not contain the address information inline but only references to the corresponding external features.

The address information is specified using the xAL address standard issued by the OASIS consortium (OASIS 2003), which provides a generic schema for all kinds of international addresses. Therefore, child elements of the xalAddress property of Address have to be structured according to the OASIS xAL schema.

AddressPropertyType, AddressType, Address

The following two excerpts of a CityGML dataset contain examples for the representation of German and British addresses in xAL. The address information is attached to building objects (bldg:Building) according to the CityGML Building module (cf. chapter 10.3). Generally, if a CityGML instance document contains address information, the namespace prefix “xAL” should be declared in the root element and must refer to “urn:oasis:names:tc:ciq:xsd:schema:xAL:2.0”. An example showing a complete CityGML dataset including a building with an address element is provided in annex F.1.
10.1.5 External code lists

The ImplicitGeometry model of CityGML’s core module introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.4):

- MimeTypeId
10.1.6 Conformance requirements

Base requirements

1. The CityModel element (type: CityModelType, substitutionGroup: gml:_FeatureCollection) shall only contain cityObjectMember elements (type: gml:FeaturePropertyType), app:appearanceMember elements (type: app:AppearancePropertyType), and gml:featureMember elements (type: gml:FeaturePropertyType) as feature members.

2. The type ExternalObjectReference introduces the two elements name (type: xs:string) and uri (type: xs:anyURI). The external reference may be specified by either of them. However, if the informationSystem property element (type: xs:anyURI) of the type ExternalReferenceType is not provided, the uri element of ExternalObjectReference must be given.

3. In order to represent address information about a feature, the corresponding thematic class of the feature shall define a property of the type AddressPropertyType. Thus, for all CityGML extension modules only the type AddressPropertyType shall be used for elements providing address information.

4. Since the concept of implicit geometries (cf. chapter 8.2) is part of the CityGML Core module, the conformance requirements introduced for implicit geometries (cf. chapter 8.2.3) are part of the conformance requirements of the core.

Referential integrity

5. The cityObjectMember element (type: gml:FeaturePropertyType) may contain a _CityObject element, which typically is an object from a derived subclass like bldg:Building, inline or an XLink reference to a remote _CityObject element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the cityObjectMember element may only point to a remote _CityObject element (where remote _CityObject elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

6. The type AddressPropertyType may contain an Address element inline or an XLink reference to a remote Address element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the corresponding element of type AddressPropertyType may only point to a remote Address element (where remote Address elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.2 Digital Terrain Model (DTM)

An essential part of a city model is the terrain. The Digital Terrain Model (DTM) of CityGML is provided by the thematic extension module Relief (cf. chapter 7). In CityGML, the terrain is represented by the class ReliefFeature in LOD 0-4 (Fig. 22 depicts the UML diagram, for XML schema definition see annex A.8). A ReliefFeature consists of one or more entities of the class ReliefComponent. Its validity may be restricted to a certain area defined by an optional validity extent polygon. As ReliefFeature and ReliefComponent are derivatives of CityObject, the corresponding attributes and relations are inherited. The class ReliefFeature is associated with different concepts of terrain representations which can coexist. The terrain may be specified as a regular raster or grid (RasterRelief), as a TIN (Triangulated Irregular Network, TINRelief), by break lines (BreaklineRelief), or by mass points (MasspointRelief). The four types are implemented by the corresponding GML3 classes: grids by RectifiedGridCoverages, break lines by Curves, mass points by Points and TINs either by TriangulatedSurfaces or by GML3 TINs. In case of TriangulatedSurfaces, the triangles are given explicitly while in case of GML3 TINs only 3D points are represented, where the triangulation can be reconstructed by standard methods (Delaunay triangulation, cf. Okabe et al. 1992). Break lines are represented by 3D curves. Mass points are simply a set of 3D points.

In a CityGML dataset the four terrain types may be combined in different ways, yielding a high flexibility. First, each type may be represented in different levels of detail, reflecting different accuracies or resolutions. Second, a part of the terrain can be described by the combination of multiple types, for example by a raster and break lines, or by a TIN and break lines. In this case, the break lines must share the geometry with the triangles. Third, neighboring regions may be represented by different types of terrain models. To facilitate this combination, each terrain object is provided with a spatial attribute denoting its extent of validity (Fig. 23). In most cases, the extent of validity of a regular raster dataset corresponds to its bounding box. This validity extent is represented by a 2D footprint polygon, which may have holes. This concept enables, for example, the modelling of a terrain by a coarse grid, where some distinguished regions are represented by a detailed, high-accuracy TIN. The boundaries between both types are given by the extent attributes of the corresponding terrain objects.

Fig. 22: UML diagram of the Digital Terrain Model in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Relief module.
Accuracy and resolution of the DTM are not necessarily dependent on those of the building model. Hence, there is the possibility to integrate building models with higher LOD to a DTM with lower accuracy or resolution.

This approach interacts with the concept of TerrainIntersecitonCurves TIC (see chapter 6.5). The TIC can be used like break lines to adjust the DTM to different building models and hence to ensure a consistent representation of the DTM. If necessary, a retriangulation may have to be processed. A TIC can also be derived by the individual intersection of the DTM and the building model. 

ReliefFeature and its ReliefComponents both have an lod attribute denoting the corresponding level of detail. In most cases, the LOD of a ReliefFeature matches the LOD of its ReliefComponents. However, it is also allowed to specify a ReliefFeature with a high LOD which consists of ReliefComponents where some of them can have a LOD lower than that of the aggregating ReliefFeature. The idea is that, for example, for a LOD3 scene it might be sufficient to use a regular grid in LOD2 with certain higher precision areas defined by ReliefComponents in LOD3. The LOD2 grid and the LOD3 components can easily be integrated using the concept of the validity extent polygon. Therefore, although some of the ReliefComponents would have been classified to a lower LOD, the whole ReliefFeature would be appropriate to use with other LOD3 models which is indicated by setting its lod value to 3.

XML namespace

The XML namespace of the CityGML Relief module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/relief/1.0. Within the XML Schema definition of the Relief module, this URI is also used to identify the default namespace.

10.2.1 Relief feature and relief component

ReliefFeatureType, ReliefFeature
The geometry of a TINRelief is defined by the GML geometry class gml:TriangulatedSurface. This allows either the explicit provision of a set of triangles (gml:TriangulatedSurface) or specifying of only the control points, break and stop lines using the subclass gml:Tin of gml:TriangulatedSurface. In the latter case, an application that processes an instance document containing a gml:Tin has to reconstruct the triangulated surface by the application of a constrained Delaunay triangulation algorithm (cf. Okabe et al. 1992).
10.2.3 Raster relief

**RasterReliefType, RasterRelief, Elevation**

```xml
<xs:complexType name="RasterReliefType">
  <xs:complexContent>
    <xs:restriction base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="grid" type="gml:gridPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfRasterRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

10.2.4 Mass point relief

**MassPointReliefType, MassPointRelief**

```xml
<xs:complexType name="MassPointReliefType">
  <xs:complexContent>
    <xs:restriction base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="reliefPoints" type="gml:MultiPointPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfMassPointRelief" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

10.2.5 Breakline relief

**BreaklineReliefType, BreaklineRelief**

```xml
<xs:complexType name="BreaklineReliefType">
  <xs:complexContent>
    <xs:restriction base="AbstractReliefComponentType">
      <xs:sequence>
        <xs:element name="ridgeOrValleyLines" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="breaklines" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```
The geometry of a BreaklineRelief can be composed of break lines and ridge/valley lines. Whereas break lines indicate abrupt changes of terrain slope, ridge/valley lines in addition mark a change of the sign of the terrain slope gradient. A BreaklineRelief must have at least one of the two properties.

10.2.6 Conformance requirements

Base requirements

1. The gml:Polygon geometry element describing the extent of validity of a _ReliefComponent element using the extent property (type: gml:PolygonPropertyType) of _ReliefComponent shall be given as 2D footprint polygon which may have inner holes.

Referential integrity

2. The reliefComponent property (type: ReliefComponentPropertyType) of the element ReliefFeature may contain a _ReliefComponent element inline or an XLink reference to a remote _ReliefComponent element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the reliefComponent property may only point to a remote _ReliefComponent element (where remote _ReliefComponent elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

3. The tin property (type: tinPropertyType) of the element TINRelief may contain a gml:TriangulatedSurface element inline or an XLink reference to a remote gml:TriangulatedSurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the tin property may only point to a remote gml:TriangulatedSurface element (where remote gml:TriangulatedSurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

4. The grid property (type: gridPropertyType) of the element RasterRelief may contain a gml:RectifiedGridCoverage element inline or an XLink reference to a remote gml:RectifiedGridCoverage element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the grid property may only point to a remote gml:RectifiedGridCoverage element (where remote gml:RectifiedGridCoverage elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.3 Building model

The building model is the most detailed thematic concept of CityGML. It allows for the representation of thematic and spatial aspects of buildings, building parts and installations in four levels of detail, LOD1 to LOD4. The building model of CityGML is defined by the thematic extension module Building (cf. chapter 7). Fig. 24 provides examples of 3D city models for each LOD.

![Building model examples](image)

Fig. 24: Examples for city or building models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left), and LOD4 (lower right) (source: District of Recklinghausen, m-g-h ingenieure+architekten GmbH).

The UML diagram of the building model is depicted in Fig. 25, for the XML schema definition see annex A.3 and below. The pivotal class of the model is _AbstractBuilding, which is a subclass of the thematic class _Site (and transitively of the root class _CityObject). _AbstractBuilding is specialised either to a Building or to a BuildingPart. Since an _AbstractBuilding consists of BuildingParts, which again are _AbstractBuildings, an aggregation hierarchy of arbitrary depth may be realised. As subclass of the root class _CityObject, an _AbstractBuilding inherits all properties from _CityObject like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences (cf. chapter 6.7). Further properties not explicitly covered by _AbstractBuilding may be modelled as GenericAttributes provided by the CityGML Generics module (cf. chapter 10.10).

Building complexes, which consist of a number of distinct buildings like a factory site or hospital complex, should be aggregated using the concept of CityObjectGroups (see chapter 6.8). The main building of the complex can be denoted by providing “main building” as the role name of the corresponding object.

Both classes Building and BuildingPart inherit the attributes of _AbstractBuilding: the class of the building, the function (e.g. residential, public, or industry), the usage, the year of construction, the year of demolition, the roof type, the measured height, and the number and individual heights of the storeys above and below ground. This set of parameters is suited for roughly reconstructing the three-dimensional shape of a building and can be provided by cadastral systems. Furthermore, Addresses can be assigned to Buildings or BuildingParts.
The geometric representation and semantic structure of an AbstractBuilding is shown in Fig. 25. The model is successively refined from LOD1 to LOD4. Therefore, not all components of a building model are represented equally in each LOD and not all aggregation levels are allowed in each LOD. In CityGML, all object classes are associated to the LODs with respect to the minimum acquisition criteria for each LOD (cf. chapter 6.5). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

In LOD1, a building model consists of a geometric representation of the building volume. Optionally, a MultiCurve representing the TerrainIntersectionCurve (see chapter 6.5) can be specified. This geometric representation is refined in LOD2 by additional MultiSurface and MultiCurve geometries, used for modelling architectural details like a roof overhang, columns, or antennas. In LOD2 and higher LODs the outer facade of a building can also be differentiated semantically by the classes BoundarySurface and BuildingInstallation. A
_BoundarySurface_ is a part of the building’s exterior shell with a special function like wall (WallSurface), roof (RoofSurface), ground plate (GroundSurface) or ClosureSurface. The BuildingInstallation class is used for building elements like balconies, chimneys, dormers or outer stairs, strongly affecting the outer appearance of a building. A BuildingInstallation may have the attributes class, function and usage (cf. Fig. 25).

In LOD3, the openings in _BoundarySurface_ objects (doors and windows) can be represented as thematic objects. In LOD4, the highest level of resolution, also the interior of a building, composed of several rooms, is represented in the building model by the class Room. This enlargement allows a virtual accessibility of buildings, e.g. for visitor information in a museum (“Location Based Services”), the examination of accommodation standards or the presentation of daylight illumination of a building. The aggregation of rooms according to arbitrary, user defined criteria (e.g. for defining the rooms corresponding to a certain storey) is achieved by employing the general grouping concept provided by CityGML (see chapter 10.3.6). Interior installations of a building, i.e. objects within a building which (in contrast to furniture) cannot be moved, are represented by the class InitBuildingInstallation. If an installation is attached to a specific room (e.g. radiators or lamps), they are associated with the Room class, otherwise (e.g. in case of rafters or pipes) with _AbstractBuilding_. A Room may have the attributes class, function and usage referenced to external code lists (chapter 10.3.7 and annex C.1). The class attribute allows a classification of rooms with respect to the stated function, e.g. commercial or private rooms, and occurs only once. The function attribute is intended to express the main purpose of the room, e.g. living room, kitchen. The attribute usage can be used if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The visible surface of a room is represented geometrically as a Solid or MultiSurface. Semantically, the surface can be structured into specialised _BoundarySurfaces_, representing floor (FloorSurface), ceiling (CeilingSurface), and interior walls (InteriorWallSurface). Room furniture, like tables and chairs, can be represented in the CityGML building model with the class BuildingFurniture. A BuildingFurniture may have the attributes class, function and usage. An example CityGML dataset containing simple buildings can be found in annex F.1.

**XML namespace**

The XML namespace of the CityGML Building module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/building/1.0. Within the XML Schema definition of the Building module, this URI is also used to identify the default namespace.

### 10.3.1 Building and building part

**BuildingType, Building**

```xml
c<wx:complexType name="BuildingType">
  <wx:complexContent>
    <wx:extension base="AbstractBuildingType">%
      <wx:sequence>
        <wx:element name="Building" type="BuildingType" substitutionGroup="_AbstractBuilding"/>
      </wx:sequence>
    </wx:extension>
  </wx:complexType>
</wxmlargetype>```

The **Building** class is one of the two subclasses of _AbstractBuilding_. If a building only consists of one (homogeneous) part, this class shall be used. A building composed of structural segments differing in, for example the number of storeys or the roof type has to be separated into one Building having one or more additional BuildingParts (see Fig. 26). The geometry and non-spatial properties of the central part of the building should be represented in the aggregating Building feature.

**BuildingPartType, BuildingPart**

```xml
c<wx:complexType name="BuildingPartType">
  <wx:complexContent>
</wxmlargetype>```

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
The class **BuildingPart** is derived from **AbstractBuilding**. It is used to model a structural part of a building (see Fig. 26). A **BuildingPart** object should be uniquely related to exactly one building or building part object.

![Building with two building parts](image1.png)  ![Building consisting of one part](image2.png)

**Fig. 26: Examples of buildings consisting of one and two building parts (source: City of Coburg).**

**AbstractBuildingType, _AbstractBuilding**

```xml
<xs:complexType name="AbstractBuildingType" abstract="true">
  <xs:complexContent>
    <xs:restriction base="core:AbstractSiteType">
      <xs:sequence>
        <xs:element name="BuildingClassType" type="gml:ClassificationType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="BuildingFunctionType" type="gml:FunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="BuildingUsageType" type="gml:UsageType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="BuildingPart" type="BuildingPartType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="_GenericApplicationPropertyOfBuildingPart" type="gml:ComplexPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfBuildingPart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```
The abstract class _AbstractBuilding contains properties for building attributes, purely geometric representations, and geometric/semantic representations of the building or building part on different levels of detail. The attributes describe:

a) The classification of the building or building part (class), the different functions (function), and the usage (usage). The permitted values for these property types are specified in a separate XML file, using the dictionary concept of GML3.

b) The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the building or building part. These attributes can be used to describe the chronology of the building development within a city model. The points of time refer to real world time.

c) The roof type of the building or building part (roofType). The permitted values for the RoofTypeType are specified in a separate XML file, using the dictionary concept of GML.

d) The measured relative height (measuredHeight) of the building or building part ridge line (highest point).

e) The number of storeys above (storeyHeightsAboveGround) and below (storeyHeightsBelowGround) ground level. The first value in a list denotes the height of the nearest storey wrt. to the ground level and last value the height of the farthest.

Spanning the different levels of detail, the building model differs in the complexity and granularity of the geometric representation and the thematic structuring of the model into components with a special semantic meaning. This is illustrated in Fig. 27, showing the same building in four different LODs. The class _AbstractBuilding has a number of properties which are associated with certain LODs.
Tab. 5 shows the correspondence of the different geometric and semantic themes of the building model to LODs. In each LOD, the volume of a building can be expressed by a *SolidGeometry* and/or a *MultiSurfaceGeometry*. The definition of a 3D Terrain Intersection Curve (TIC), used to integrate buildings from different sources with the Digital Terrain Model, is also possible in all four LODs. The TIC can – but does not have to – build closed rings around the building or building parts.

In LOD1 (see Fig. 27 a), the different structural entities of a building are aggregated to simple blocks and not differentiated in detail. The volumetric and surface parts of the exterior building shell are identical and only one of the corresponding properties (*lod1Solid* or *lod1MultiSurface*) must be used.

In LOD2 and higher levels of detail, the exterior shell of a building is not only represented geometrically as *SolidGeometry* and/or *MultiSurfaceGeometry*, it can also be composed of semantic objects. The base class for all objects semantically structuring the building shell is *BoundarySurface* (see chapter 10.3.2), which is associated with a *MultiSurfaceGeometry*. If in a building model there is both a geometric representation of the exterior shell as volume or surface model and a semantic representation by means of thematic *BoundarySurfaces*, the geometric representation must not explicitly define the geometry, but has to reference the *MultiSurfaceGeometry* of the *BoundarySurfaces*.

<table>
<thead>
<tr>
<th>Geometric / semantic theme</th>
<th>Property type</th>
<th>LOD 1</th>
<th>LOD 2</th>
<th>LOD 3</th>
<th>LOD 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume part of the building shell</td>
<td>gml:SolidType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Surface part of the building shell</td>
<td>gml:MultiSurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Terrain Intersection Curve</td>
<td>gml:MultiCurveType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Curve part of the building shell</td>
<td>gml:MultiCurveType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>BoundarySurfaces (chapter 10.3.2)</td>
<td>_BoundarySurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Outer building installations (chapter 10.3.3)</td>
<td>BuildingInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Openings (chapter 10.3.4)</td>
<td>_OpeningType</td>
<td>•</td>
<td></td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Rooms (chapter 10.3.5)</td>
<td>RoomType</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interior building installations (chapter 10.3.5)</td>
<td>IntBuildingInstallationType</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tab. 5: Semantic themes of the class AbstractBuilding.

Apart from *BuildingParts*, smaller features of the building (“outer building installations”) can also strongly affect the building characteristic. These features are modelled by the class *BuildingInstallation* (see chapter 10.3.3). Typical candidates for this class are chimneys (see Fig. 27 c), dormers (see Fig. 26), balconies, outer stairs, or antennas. *BuildingInstallations* may only be included in LOD2 models, if their extents exceed the minimum dimensions as specified in chapter 6.2. For the geometrical representation of the class *BuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used.

The class *AbstractBuilding* has no additional properties for LOD3. Besides the higher requirements on geometric precision and smaller minimum dimensions, the main difference of LOD2 and LOD3 buildings concerns the class *BoundarySurface* (see chapter 10.3.2). In LOD3, openings in a building corresponding with windows or doors (see Fig. 27 c) are modelled by the (abstract) class *Opening* and the derived classes *Window* and *Door* (see chapter 10.3.4).

With respect to the exterior building shell, the LOD4 data model is identical to that of LOD3. But LOD4 provides for a possibility to describe the interior structure of a building with the classes *IntBuildingInstallation* and *Room* (see chapter 10.3.5).

Each *Building* or *BuildingPart* feature may be assigned zero or more addresses using the *address* property. The corresponding *AddressPropertyType* is defined within the CityGML core module (cf. chapter 10.1.4).
10.3.2 Boundary surfaces

**AbstractBoundarySurfaceType, _BoundarySurface**

_BoundarySurface_ is the (abstract) base class for several thematic classes, structuring the exterior shell of a building and the visible surface of a room. It is a subclass of _CityObject_ and thus inherits all properties like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences. From _BoundarySurface_, the thematic classes RoofSurface, WallSurface, GroundSurface, ClosureSurface, FloorSurface, InteriorWallSurface, and CeilingSurface are derived. The thematic classification of building surfaces is illustrated in Fig. 28 and subsequently specified.

For each LOD between 2 and 4, the geometry of a _BoundarySurface_ may be defined by a different MultiSurfaceGeometry. In LOD2, this surface geometry must be simply connected, which means that the components of the MultiSurface (e.g. gml:Polygon) must not have inner holes (gml:interior).

In LOD3 and LOD4, a _BoundarySurface_ may reference _Openings_ (see 10.3.4) like doors and windows. If the geometric location of _Openings_ topologically lies within a surface component (e.g. gml:Polygon) of the MultiSurfaceGeometry, these _Openings_ must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. If such an opening is sealed by a Door, a Window, or a ClosureSurface, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. However, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface’s normal vector). The embrasure surfaces of an Opening belong to the relevant adjacent _BoundarySurface_. If, for example a door seals the Opening, the embrasure surface on the one side of the door belongs to the InteriorWallSurface and on the other side to the WallSurface (Fig. 28 on the right).

---

**Fig. 28: Classification of BoundarySurfaces (left), in particular for Openings (right) (graphic: IGG Uni Bonn).**
RoofSurfaceType, RoofSurface

The major roof parts of a building or building part are expressed by the class RoofSurface. Secondary parts of a roof with a specific semantic meaning like dormers or chimneys should be modelled as BuildingInstallations.

WallSurfaceType, WallSurface

All parts of the building facade visible from the outside are modelled by the class WallSurface.

GroundSurfaceType, GroundSurface

The ground plate of a building or building part is modelled by the class GroundSurface. The polygon defining the ground plate is congruent with the building’s footprint. However, the surface normal of the ground plate is pointing downwards.

ClosureSurfaceType, ClosureSurface

The ground plate of a building or building part is modelled by the class GroundSurface. The polygon defining the ground plate is congruent with the building’s footprint. However, the surface normal of the ground plate is pointing downwards.
An opening in a building not filled by a door or window can be sealed by a virtual surface called *ClosureSurface* (see chapter 6.4). Hence, buildings with open sides like a barn or a hangar, can be virtually closed in order to be able to compute their volume. *ClosureSurfaces* are also used in the interior building model. If two rooms with a different function (e.g. kitchen and living room) are directly connected without a separating door, a *ClosureSurface* should be used to separate or connect the volumes of both rooms.

**FloorSurfaceType, FloorSurface**

The class *FloorSurface* must only be used in the LOD4 interior building model for modelling the floor of a room.

**InteriorWallSurfaceType, InteriorWallSurface**

The class *InteriorWallSurface* must only be used in the LOD4 interior building model for modelling the visible surfaces of the room walls.

**CeilingSurfaceType, CeilingSurface**
The class CeilingSurface must only be used in the LOD4 interior building model for modelling the ceiling of a room.

### 10.3.3 Outer building installations

**BuildingInstallationType, BuildingInstallation**

A BuildingInstallation is an outer component of a building which has not the significance of a BuildingPart, but which strongly affects the outer characteristic of the building. Examples are chimneys, stairs, antennas, balconies or attached roofs above stairs and paths. A BuildingInstallation optionally has attributes class, function and usage. The attribute class - which can only occur once - represents a general classification of the installation. With the attributes function and usage, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values is specified in a GML dictionary. For the geometrical representation of a BuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used. A BuildingInstallation object should be uniquely related to exactly one building or building part object.

### 10.3.4 Openings

**AbstractOpeningType, _Opening**

The class _Opening is the (abstract) base class for semantically describing openings like doors or windows in outer or inner walls. Openings only exist in models of LOD3 or LOD4. Each _Opening is associated with a MultiSurfaceGeometry.

**WindowType, Window**
The class *Window* is used for modelling windows in the exterior shell of a building, or hatches between adjacent rooms. The formal difference between the classes *Window* and *Door* is that – in normal cases – *Windows* are not specifically intended for the transit of people or vehicles.

**DoorType, Door**

The class *Door* is used for modelling doors in the exterior shell of a building, or between adjacent rooms. Doors can be used by people to enter or leave a building or room. In contrast to a *ClosureSurface* a door may be closed, blocking the transit of people. A *Door* may be assigned zero or more addresses. The corresponding *AddressPropertyType* is defined within the CityGML core module (cf. chapter 10.1.4).

### 10.3.5 Building interior

**RoomType, Room**

A *Room* is a semantic object for modelling the free space inside a building and should be uniquely related to exactly one building or building part object. It should be closed (if necessary by using *ClosureSurfaces*) and the geometry normally will be described by a solid (*lod4Solid*). However, if the topological correctness of the
boundary cannot be guaranteed, the geometry can alternatively be given as a MultiSurface \((\text{lod4MultiSurface})\). The surface normals of the outer shell of a GML solid must point outwards. This is important to consider when Room surfaces should be assigned Appearances. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the room.

In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialised BoundarySurfaces (FloorSurface, CeilingSurface, InteriorWallSurface, see chapter 10.3.2).

A special task is the modelling of passages between adjacent rooms. The room solids are topologically connected by the surfaces representing hatches, doors or closure surfaces that seal open doorways. Rooms are defined as being adjacent, if they have common Openings or ClosureSurfaces. The surface that represents the opening geometrically is part of the boundaries of the solids of both rooms, or the opening is referenced by both rooms on the semantic level. This adjacency implies an accessibility graph, which can be employed to determine the spread of e.g. smoke or gas, but which can also be used to compute escape routes using classical shortest path algorithms (see Fig. 29).

Rooms may have BuildingFurnitures and IntBuildingInstallations. A BuildingFurniture is a movable part of a room, such as a chair or furniture. A BuildingFurniture object should be uniquely related to exactly one room object.

**BuildingFurnitureType, BuildingFurniture**

```xml
<xs:complexType name="BuildingFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="BuildingFurnitureClassType" minOccurs="0"/>
        <xs:element name="function" type="BuildingFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="BuildingFurnitureUsageType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfBuildingFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="BuildingFurniture" type="BuildingFurnitureType" substitutionGroup="core:CityObject"/>
```

**IntBuildingInstallationType, IntBuildingInstallation**

```xml
<xs:complexType name="IntBuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

---

Fig. 29: Accessibility graph derived from topological adjacencies of room surfaces (graphic: IGG Uni Bonn).

---

OGC 08-007r1

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
An IntBuildingInstallation is an object inside a building with a specialised function or semantic meaning. In contrast to BuildingFurniture, IntBuildingInstallations are permanently attached to the building structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class IntBuildingInstallation can either be associated with a room (class Room), or with the complete building / building part (class AbstractBuilding, see chapter 10.3.1). However, they should be uniquely related to exactly one room or one building / building part object. An IntBuildingInstallation optionally has attributes class, function and usage. The attribute class, which can only occur once, represents a general classification of the internal building component. With the attributes function and usage, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values is specified in a GML dictionary. For the geometrical representation of an IntBuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used.

10.3.6 Modelling building storeys using CityObjectGroups

CityGML does not provide a specific concept for the representation of storeys as it is available in the AEC/FM standard IFC (IAI 2006). However, a storey can be represented as an explicit aggregation of all building features on a certain height level using CityGML’s notion of CityObjectGroups (cf. chapter 10.9). This would include Rooms, Doors, Windows, IntBuildingInstallations and BuildingFurniture. If thematic surfaces like walls and interior walls should also be associated to a specific storey, this might require the vertical fragmentation of these surfaces (one per storey), as in virtual 3D city models they typically span the whole façade.

In order to model building storeys with CityGML’s generic grouping concept, a nested hierarchy of CityGMLGroup objects has to be used. On the first level, all semantic objects belonging to a specific storey are grouped. The attributes of the corresponding CityObjectGroup object are set as follows:

- The class attribute shall be assigned the value “building separation”.
- The function attribute shall be assigned the value “lodXStorey” with X between 1 and 4 in order to denote that this group represents a storey wrt. a specific LOD.
- The storey name or number can be stored in the gml:name property.

On the second level, the CityGMLGroup objects representing different storeys are grouped itself. By using the generic aggregation concept of CityGMLGroup, the “storeys group” is associated with the corresponding Building or BuildingPart object. The class attribute of the storeys group shall be assigned the value “building storeys”.

10.3.7 External code lists

The building model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.1):

- BuildingClassType
- BuildingFunctionType
- BuildingUsageType
- RoofTypeType
- BuildingInstallationClassType
- BuildingInstallationFunctionType
• BuildingInstallationUsageType
• IntBuildingInstallationClassType
• IntBuildingInstallationFunctionType
• IntBuildingInstallationUsageType
• BuildingFurnitureClassType
• BuildingFurnitureFunctionType
• BuildingFurnitureUsageType
• RoomClassType
• RoomFunctionType
• RoomUsageType

10.3.8 Conformance requirements

Base requirements

1. If a building only consists of one (homogeneous) part, it shall be represented by the element Building. However, if a building is composed of individual structural segments, it shall be modelled as a Building element having one or more additional BuildingPart elements. Only the geometry and non-spatial properties of the main part of the building should be represented within the aggregating Building element.

Usage restriction of building model components according to different LODs

2. The lodXSolid and lodXMultiSurface, $X \in [1..4]$, properties (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType) of AbstractBuilding may be used to geometrically represent the exterior shell of a building (as volume or surface model) within each LOD. For LOD1, either lod1Solid or lod1MultiSurface must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.

3. Starting from LOD2, the exterior shell of an AbstractBuilding may be semantically decomposed into BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of AbstractBuilding. Only RoofSurface, WallSurface, GroundSurface, and ClosureSurface as subclasses of BoundarySurface are allowed. The boundedBy property (not to be confused with the gml:boundedBy property) shall not be used if the building is only represented in LOD1.

If the exterior shell is represented by BoundarySurface elements, an additional geometric representation as volume or surface model using the lodXSolid and lodXMultiSurface, $X \in [2..4]$, properties shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of BoundarySurface within each LOD using the XLink concept of GML 3.1.1.

4. Starting from LOD2, curve parts of the building shell may be represented using the lodXMultiCurve, $X \in [2..4]$, property of AbstractBuilding. This property shall not be used if the building is only represented in LOD1.

5. Starting from LOD2, the outerBuildingInstallation property (type: BuildingInstallationPropertyType) of AbstractBuilding may be used to model BuildingInstallation elements. BuildingInstallation elements shall only be used to represent outer characteristics of a building which do not have the significance of building parts. The outerBuildingInstallation property shall not be used if the building is only represented in LOD1.

6. Starting from LOD3, openings of BoundarySurface elements may be modelled using the opening property (type: OpeningPropertyType) of BoundarySurface. This property shall not be used for BoundarySurface elements only represented in LOD2. Accordingly, the surface geometry representing a BoundarySurface in LOD2 must be simply connected and, thus, must not have inner holes.

The opening property of BoundarySurface may contain or reference Opening elements. If the geometric location of an Opening element topologically lies within a surface component of the BoundarySurface, the opening must also be represented as inner hole of that surface. The embrasure surface of an Opening element shall belong to the relevant adjacent BoundarySurface.
7. Starting from LOD4, the interiorRoom property (type: InteriorRoomPropertyType) of _AbstractBuilding may be used to semantically model the free space inside the building by Room elements. This property shall not be used if the building is only represented in LOD 1 – 3. The Room element may be geometrically represented as a surface or volume model, using its lod4Solid or lod4MultiSurface property (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType).

8. Starting from LOD4, the interiorBuildingInstallation property (type: IntBuildingInstallationPropertyType) of _AbstractBuilding may be used to represent immovable objects inside the building that are permanently attached to the building structure. The interiorBuildingInstallation property shall not be used if the building is only represented in LOD 1 – 3. Furthermore, the interiorBuildingInstallation property shall only be used if the object cannot be associated with a Room element. In the latter case, the roomInstallation property (type: IntBuildingInstallationPropertyType) of the corresponding Room element shall be used to represent the object.

Referential integrity

9. The boundedBy property (type: BoundarySurfacePropertyType) of the element _AbstractBuilding may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only RoofSurface, WallSurface, GroundSurface, and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of _AbstractBuilding.

10. The outerBuildingInstallation property (type: BuildingInstallationPropertyType) of the element _AbstractBuilding may contain a BuildingInstallation element inline or an XLink reference to a remote BuildingInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the outerBuildingInstallation property may only point to a remote BuildingInstallation element (where remote BuildingInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

11. The interiorBuildingInstallation property (type: IntBuildingInstallationPropertyType) of the element _AbstractBuilding may contain an IntBuildingInstallation element inline or an XLink reference to a remote IntBuildingInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorBuildingInstallation property may only point to a remote IntBuildingInstallation element (where remote IntBuildingInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

12. The interiorRoom property (type: InteriorRoomPropertyType) of the element _AbstractBuilding may contain a Room element inline or an XLink reference to a remote Room element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorRoom property may only point to a remote Room element (where remote Room elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

13. The consistsOfBuildingPart property (type: BuildingPartPropertyType) of the element _AbstractBuilding may contain a BuildingPart element inline or an XLink reference to a remote BuildingPart element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the consistsOfBuildingPart property may only point to a remote BuildingPart element (where remote
BuildingPart elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

14. The address property (type: core:AddressPropertyType) of the element _AbstractBuilding may contain an core:Address element inline or an XLink reference to a remote core:Address element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the address property may only point to a remote core:Address element (where remote core:Address elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

15. The opening property (type: OpeningPropertyType) of the element _BoundarySurface may contain an _Opening element inline or an XLink reference to a remote _Opening element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the opening property may only point to a remote _Opening element (where remote _Opening elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

16. The address property (type: core:AddressPropertyType) of the element Door may contain an core:Address element inline or an XLink reference to a remote core:Address element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the address property may only point to a remote core:Address element (where remote core:Address elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

17. The boundedBy property (type: BoundarySurfacePropertyType) of the element Room may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of Room.

18. The interiorFurniture property (type: InteriorFurniturePropertyType) of the element Room may contain an BuildingFurniture element inline or an XLink reference to a remote BuildingFurniture element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorFurniture property may only point to a remote BuildingFurniture element (where remote BuildingFurniture elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

19. The roomInstallation property (type: IntBuildingInstallationPropertyType) of the element Room may contain an IntBuildingInstallation element inline or an XLink reference to a remote IntBuildingInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the roomInstallation property may only point to a remote IntBuildingInstallation element (where remote IntBuildingInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.4 Water bodies

Waters have always played an important role in urbanisation processes and cities were built preferably at rivers and places where landfall seemed to be easy. Obviously, water is essential for human alimentation and sanitation. Water bodies present the most economical way of transportation and are barriers at the same time, that avoid instant access to other locations. Bridging waterways caused the first efforts of construction and resulted in high-tech bridges of today. The landscapes of many cities are dominated by water, which directly relates to 3D city models. Furthermore, water bodies are important for urban life as subject of recreation and possible hazards as e.g. floods.

The distinct character of water bodies compared with the permanence of buildings, roadways, and terrain is considered in this thematic model. Water bodies are dynamic surfaces. Tides occur regularly, but irregular events predominate with respect to natural forces, for example flood events. The visible water surface changes in height and its covered area with the necessity to model its semantics and geometry distinct from adjacent objects like terrain or buildings.

This first modelling approach of water bodies fulfils the requirements of 3D city models. It does not inherit any hydrological or other dynamic aspects. In these terms it does not claim to be complete. However, the semantic and geometric description given here allows further enhancements of dynamics and conceptually different descriptions. The water bodies model of CityGML is embraced by the thematic extension module WaterBody (cf. chapter 7).

The water bodies model represents the thematic aspects and three-dimensional geometry of rivers, canals, lakes, and basins. In the LOD 2-4 water bodies are bounded by distinct thematic surfaces. These surfaces are the obligatory WaterSurface, defined as the boundary between water and air, the optional WaterGroundSurface, defined as the boundary between water and underground (e.g. DTM or floor of a 3D basin object), and zero or more WaterClosureSurfaces, defined as virtual boundaries between different water bodies or between water and the end of a modelled region (see Fig. 30). A dynamic element may be the WaterSurface to represent temporarily changing situations of tidal flats.

![Fig. 30: Illustration of a water body defined in CityGML (graphic: IGG Uni Bonn).](image)

The UML diagram of the water body model is depicted in Fig. 31, for XML schema definition see below and annex A.11. Each WaterBody object may have the attributes class, function and usage referencing to external code lists (cf. chapter 10.4.3 and annex C.7). The attribute class defines the classification of the object, e.g. lake, river, or fountain and can occur only once. The attribute function contains the purpose of the object like, for example national waterway or public swimming, while the attribute usage defines the actual usages, e.g. whether the water body is navigable. The latter two attributes can occur multiple times.

WaterBody is a subclass of _WaterObject and thus of the root class _CityObject. The class _WaterObject can be differentiated in further subclasses of water objects in the future. The geometrical representation of the WaterBody varies through the different levels of detail. Since WaterBody is a subclass of _CityObject and hence a feature, it inherits the attribute gml:name. The WaterBody can be differentiated semantically by the class _WaterBoundarySurface. A _WaterBoundarySurface is a part of the water body’s exterior shell with a special
function like WaterSurface, WaterGroundSurface or WaterClosureSurface. As with any _CityObject, WaterBody objects as well as WaterSurface, WaterGroundSurface, and WaterClosureSurface may be assigned External-References (cf. chapter 6.7) and may be augmented by GenericAttributes using CityGML’s Generics module (cf. chapter 10.10).

The optional attribute waterLevel of a WaterSurface can be used to describe the water level, for which the given 3D surface geometry was acquired. This is especially important when the water body is influenced by the tide. The allowed values are defined in the respective external code list.

Both LOD0 and LOD1 represent a low level of illustration and high grade of generalisation. Here the rivers are modelled as MultiCurve geometry and brooks are omitted. Seas, oceans and lakes with significant extent are represented as a MultiSurface (Fig. 31). Every WaterBody may be assigned a combination of geometries of different types. Linear water bodies are represented as a network of 3D curves. Each curve is composed of straight line segments, where the line orientation denotes the flow direction (water flows from the first point of a curve, e.g. a gml:LineString, to the last). Areal objects like lakes or seas are represented by 3D surface geometries of the water surface.

Starting from LOD1 water bodies may also be modelled as water filled volumes represented by Solids. If a water body is represented by a Solid in LOD2 or higher, the surface geometries of the corresponding thematic WaterClosureSurface, WaterGroundSurface, and WaterSurface objects must coincide with the exterior shell of the Solid. This can be ensured, if for one LOD X the respective lodXSurface elements (where X is between 2 and 4) of WaterClosureSurface, WaterGroundSurface, and WaterSurface do not redundantly define gml:Polygons, but instead reference the corresponding polygons (using XLink) within the CompositeSurface that defines the exterior shell of the Solid.

LOD2 to LOD4 demand a higher grade of detail and therefore any WaterBody will be outlined by thematic surfaces or a solid composed of the surrounding thematic surfaces.

Every object of the class WaterSurface, WaterClosureSurface, and WaterGroundSurface must have at least one associated surface geometry. This means, that every WaterSurface, WaterClosureSurface, and WaterGroundSurface feature within a CityGML instance document must contain at least one of the following properties: lod2Surface, lod3Surface, lod4Surface.

The water body model implicitly includes the concept of TerrainIntersectionCurves (TIC), e.g. to specify the exact intersection of the DTM with the 3D geometry of a WaterBody or to adjust a WaterBody or WaterSurface to the surrounding DTM (see chapter 6.5). The rings defining the WaterSurface polygons implicitly delineate the intersection of the water body with the terrain or basin.

XML namespace

The XML namespace of the CityGML WaterBody module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/waterbody/1.0. Within the XML Schema definition of the WaterBody module, this URI is also used to identify the default namespace.
10.4.1 Water body

AbstractWaterObjectType, WaterObject

WaterBodyType, WaterBody

10.4.2 Boundary surfaces

With respect to different functions and characteristics three boundary classes for water are defined to build a solid or composite surface geometry (Fig. 30).

1. Boundary class “Air to Water”. The WaterSurface is mandatory to the model and usually is registered using photogrammetric analysis or mapping exploration. The representation may vary due to tidal flats or changing water levels, which can be reflected by including different static water surfaces having different waterLevels (WaterLevelType), as for example highest flooding event, mean sea level, or minimum water level, given in an external code list. This offers the opportunity to describe significant water surfaces due to levels that are important for certain representations e.g. in tidal zones.

2. Boundary class “Water to Ground”. The WaterGroundSurface may be known by sonar exploration or other depth measurements. Also part of the ground surface is the boundary “Water to Building”. The ground surface might be identical to the underwater terrain model, but also describes the contour to other underwater objects. The usefulness of this concept arises from the existence of water defence buildings like sluices, sills, flood barrage or tidal power stations. The use of WaterGroundSurface as boundary layer to buildings is relevant in urban situations, where buildings enclose the defined water completely such as fountains and swimming pools. Together, the WaterSurface and WaterGroundSurface enclose the WaterBody as a volume.
3. Boundary class “Water to Water”. The WaterClosureSurface is an optional feature that comes in use, when the union of the WaterSurfaces and WaterGroundSurfaces of a water body does not define a closed volume. The WaterClosureSurface is then used to complete the enclosure of water volumes and to separate water volumes from those where only the surface is known. This might occur, where the cross section and ground surface of rivers is partly available during its course.

_WaterBoundarySurfaces should only be included as parts of corresponding WaterBody objects and should not be used as stand-alone objects within a CityGML model.

AbstractWaterBoundarySurfaceType, _WaterBoundarySurface

WaterSurfaceType, WaterSurface

WaterGroundSurfaceType, WaterGroundSurface

WaterClosureSurfaceType, WaterClosureSurface
10.4.3 External code lists

The water bodies model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.7):

- WaterLevelType
- WaterBodyClassType
- WaterBodyFunctionType
- WaterBodyUsageType

10.4.4 Conformance requirements

Base requirements

1. For LOD0 and LOD1, the geometry of a WaterBody may be modelled as a linear network using gml:MultiCurve geometry elements. In that case, each gml:MultiCurve shall be composed of straight line segments, where the line orientation denotes the flow direction. The flow direction is from the first point of a line segment to its last point.

2. Starting from LOD2, the exterior shell of a WaterBody may be semantically decomposed into WaterBoundarySurface elements using the boundedBy property (type: BoundedByWaterSurfacePropertyType) of WaterBody. The boundedBy property shall not be used if the water body is only represented in lower LODs.

3. Each WaterBoundarySurface element must have at least one associated surface geometry given by the lodXSurface, X ∈ [2..4], properties of WaterBoundarySurface. They may not be given as stand-alone city objects within a CityGML model.

Referential integrity

5. The boundedBy property (type: BoundedByWaterSurfacePropertyType) of the element WaterBody may contain a WaterBoundarySurface element inline or an XLink reference to a remote WaterBoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote WaterBoundarySurface element (where remote WaterBoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.5 Transportation objects

The transportation model of CityGML is a multi-functional, multi-scale model focusing on thematic and functional as well as on geometrical/topological aspects. Transportation features are represented as a linear network in LOD0. Starting from LOD1, all transportation features are geometrically described by 3D surfaces. The areal modelling of transportation features allows for the application of geometric route planning algorithms. This can be useful to determine restrictions and manoeuvres required along a transportation route. This information can also be employed for trajectory planning of mobile robots in the real world or the automatic placement of avatars (virtual people) or vehicle models in 3D visualisations and training simulators. The transportation model of CityGML is provided by the thematic extension module Transportation (cf. chapter 7).

The main class is TransportationComplex, which represents, for example, a road, a track, a railway, or a square. Fig. 32 illustrates the four different thematic classes.

Fig. 32: Representations of TransportationComplex (from left to right: examples of road, track, rail, and square)  
(source: Rheinmetall Defence Electronics).

A TransportationComplex is composed of the parts TrafficArea and AuxiliaryTrafficArea. Fig. 33 depicts an example for a LOD2 TransportationComplex configuration within a virtual 3D city model. The Road consists of several TrafficAreas for the sidewalks, road lanes, parking lots, and of AuxiliaryTrafficAreas below the raised flower beds.

Fig. 33: Example for the representation of a TransportationComplex in LOD2 in CityGML: a road, which is the aggregation of TrafficAreas and AuxiliaryTrafficAreas (source: City of Solingen, IGG Uni Bonn).

Fig. 34 depicts the UML diagram of the transportation model, for XML schema definition see annex A.9.
The road itself is represented as a TransportationComplex, which is further subdivided into TrafficAreas and AuxiliaryTrafficAreas. The TrafficAreas are those elements, which are important in terms of traffic usage, like car driving lanes, pedestrian zones and cycle lanes. The AuxiliaryTrafficAreas are describing further elements of the road, like kerbstones, middle lanes, and green areas.

TransportationComplex objects can be thematically differentiated using the subclasses Track, Road, Railway, and Square. Every TransportationComplex has the attributes function and usage, referencing to external code lists (chapter 10.5.4 and annex C.6). The attribute function describes the purpose of the object like, for example national motorway, country road, or airport, while the attribute usage can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

In addition every TrafficArea may have the attributes function, usage, and surfaceMaterial. The function describes, if the object may be a car driving lane, a pedestrian zones, or a cycle lane, while the usage attribute indicates which modes of transportation can use it (e.g. pedestrian, car, tram, roller skates). The attribute surfaceMaterial specifies the type of pavement and may also be used for AuxiliaryTrafficAreas (e.g. asphalt, concrete, gravel, soil, rail, grass etc.). The function attribute of the AuxiliaryTrafficArea defines, for example kerbstones, middle lanes, or green areas. The possible values are also specified in external code lists.

The shape of each traffic area is defined by an area geometry. Additional metadata may be defined by using attributes from pre-defined catalogues. This affects the function of the area, the usage and surface material definition for each area. The attribute catalogues may be customer- or country-specific. The following tables show examples for various kinds of TrafficArea:

<table>
<thead>
<tr>
<th>Example:</th>
<th>TransportationComplex – Function</th>
<th>Country road</th>
<th>Motorway entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TrafficArea – Usage</td>
<td>car, truck, bus, taxi, motorcycle</td>
<td>car, truck, bus, taxi, motorcycle</td>
</tr>
<tr>
<td></td>
<td>TrafficArea – Function</td>
<td>driving lane</td>
<td>motorway_entry</td>
</tr>
<tr>
<td></td>
<td>TrafficArea – SurfaceMaterial</td>
<td>asphalt</td>
<td>concrete</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
<th>Runway of an airport</th>
<th>Apron of an airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransportationComplex – Function</td>
<td>road</td>
<td>apron</td>
</tr>
<tr>
<td>TrafficArea – Usage</td>
<td>aeroplane</td>
<td>aeroplane, car, truck, bus, pedestrian</td>
</tr>
<tr>
<td>TrafficArea – Function</td>
<td>airport – runway</td>
<td>airport – apron</td>
</tr>
<tr>
<td>TrafficArea – SurfaceMaterial</td>
<td>concrete</td>
<td>concrete</td>
</tr>
</tbody>
</table>
TransportationComplex is a subclass of _TransportationObject and of the root class _CityObject. The geometrical representation of the TransportationComplex varies through the different levels of detail. Since TransportationComplex is a subclass of _CityObject and hence a feature, it inherits the attribute gml:name. The street name is also stored within the gml:name property of the Road feature.

In the coarsest LOD0 the transportation complexes are modelled by line objects establishing a linear network. On this abstract level, path finding algorithms or similar analyses can be executed. It also can be used to generate schematic drawings and visualisations of the transport network. Since this abstract definition of transportation network does not contain explicit description of the transportation objects, it may be task of the viewer application to generate the graphical visualisation, for example by using a library with style-definitions (width, color resp. texture) for each transportation object.

Starting from LOD1 a TransportationComplex provides an explicit surface geometry, reflecting the actual shape of the object, not just its centerline. In LOD2 to LOD4, it is further subdivided thematically into TrafficAreas, which are used by transportation, such as cars, trains, public transport, airplanes, bicycles or pedestrians and in AuxiliaryTrafficAreas, which are of minor importance for transportation purposes, for example road markings, green spaces or flower tubs. The different representations of a TransportationComplex for each LOD are illustrated in Fig. 35.

In LOD0 areal transportation objects like squares should be modelled in the same way as in GDF, the ISO standard for transportation networks, which is used in most car navigation systems. In GDF a square is typically represented as a ring surrounding the place and to which the incident roads connect. CityGML does not cover further functional aspects of transportation network models (e.g. speed limits) as it is intended to complement and not replace existing standards like GDF. However, if specific functional aspects have to be associated with CityGML transportation objects, GenericAttributes provided by CityGML’s Generics module (cf. chapter 10.10) can be used. Moreover, further objects of interest can be added from other information systems by the use of ExternalReferences (see chapter 6.11). For example, GDF datasets, which provide additional information for car navigation, can be used for simulation and visualisation of traffic flows. The values of the object attributes can be augmented using the concept of dictionaries (see chapter 6.6). These directories may be country- or user-specific (especially for country-specific road signs and signals).
Fig. 36: TransportationComplex in LOD 2-4: representation of a road with a complex cross-section profile (example shows urban road) (source: Rheinmetall Defence Electronics).

The following example shows a complex urban crossing. The picture on the left is a screenshot of an editor application for a training simulator, which allows the definition of road networks consisting of transportation objects, external references, buildings and vegetation objects. On the right, the 3D representation of the defined crossing is shown including all referenced static and dynamic models.

Fig. 37: Complex urban intersection (left: linear transportation network with surface descriptions and external references, right: generated scene) (source: Rheinmetall Defence Electronics).

XML namespace

The XML namespace of the CityGML Transportation module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/transportation/1.0. Within the XML Schema definition of the Transportation module, this URI is also used to identify the default namespace.

10.5.1 Transportation complex

AbstractTransportationObjectType, _TransportationObject

```xml
<xs:complexType name="AbstractTransportationObjectType" abstract="true">
  <xs:complexContent>
  </xs:complexContent>
</xs:complexType>
```
_TransportationObject represents the abstract superclass for transportation objects. Future extensions of the CityGML transportation model shall be modelled as subclasses of this class.

**TransportationComplexType, TransportationComplex**

This type and element describes transportation complexes like roads or railways which may be aggregated from different thematic components (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As a subclass of _CityObject, TransportationComplex inherits all attributes and relations, in particular an id, names, external references, and generalisation relations. Furthermore, it represents the superclass for thematically distinct types of transportation complexes.

### 10.5.2 Subclasses of transportation complexes

**TrackType, Track**

A *Track* is a small path mainly used by pedestrians. It is a subclass of *TransportationComplex* and thus inherits all its attributes and relations.
RoadType, Road

```xml
<xs:complexType name="RoadType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoad" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Road is intended to be used to represent transportation features that are mainly used by vehicles like cars, for example streets, motorways, and country roads. It is a subclass of TransportationComplex and thus inherits all its attributes and relations.

RailwayType, Railway

```xml
<xs:complexType name="RailwayType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRailway" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Railway represents routes that are utilised by rail vehicles like trams or trains. It is a subclass of TransportationComplex and thus inherits all its attributes and relations.

SquareType, Square

```xml
<xs:complexType name="SquareType">
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfSquare" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A Square is an open area commonly found in cities (e.g. a plaza, market square). It is a subclass of TransportationComplex and thus inherits all its attributes and relations.

10.5.3 Subdivisions of transportation complexes

TrafficAreaType, TrafficArea

```xml
<xs:complexType name="TrafficAreaType">
</xs:complexType>
```
10.5.4 External code lists

The transportation model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.6):

- TransportationComplexFunctionType
- TransportationComplexUsageType
- TrafficAreaFunctionType
- TrafficAreaUsageType
- AuxiliaryTrafficAreaFunctionType
- TrafficSurfaceMaterialType

10.5.5 Conformance requirements

Base requirements

1. For LOD0, the geometry of a TransportationComplex shall be modelled using GML line objects representing the centerline of the transportation complex. The line objects shall establish a linear network. Thus, the lod0Network property (type: gml:GeometricComplexPropertyType) of the element TransportationComplex may only contain or reference an appropriate curve geometry element.

2. Starting from LOD2, the trafficArea property (type: TrafficAreaPropertyType) as well as the auxiliaryTrafficArea property (type: AuxiliaryTrafficAreaPropertyType) of the element TransportationComplex may be used. These properties shall not be used if the transportation complex is only represented in lower LODs.
Referential integrity

3. The trafficArea property (type: TrafficAreaPropertyType) of the element TransportationComplex may contain a TrafficArea element inline or an XLink reference to a remote TrafficArea element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the trafficArea property may only point to a remote TrafficArea element (where remote TrafficArea elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

4. The auxiliaryTrafficArea property (type: TrafficAreaPropertyType) of the element TransportationComplex may contain an AuxiliaryTrafficArea element inline or an XLink reference to a remote AuxiliaryTrafficArea element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the auxiliaryTrafficArea property may only point to a remote AuxiliaryTrafficArea element (where remote AuxiliaryTrafficArea elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
Vegetation features are important components of a 3D city model, since they support the recognition of the surrounding environment. By the analysis and visualisation of vegetation objects, statements on their distribution, structure and diversification can be made. Habitats can be analysed and impacts on the fauna can be derived. The vegetation model may be used as a basis for simulations of, for example forest fire, urban aeration or micro climate. The model could be used, for example to examine forest damage, to detect obstacles (e.g. concerning air traffic) or to perform analysis tasks in the field of environmental protection. The vegetation model of CityGML is defined by the thematic extension module Vegetation (cf. chapter 7).

The vegetation model of CityGML distinguishes between solitary vegetation objects like trees and vegetation areas, which represent biotopes like forests or other plant communities (Fig. 38). Single vegetation objects are modelled by the class SolitaryVegetationObject, while for areas filled with a specific vegetation the class PlantCover is used. The geometry representation of a PlantCover feature may be a MultiSurface or a MultiSolid, depending on the vertical extent of the vegetation. For example regarding forests, a MultiSolid representation might be more appropriate. The UML diagram of the vegetation model is depicted in Fig. 39, for XML schema definition see below and annex A.10.

![Fig. 38: Example for vegetation objects of the classes SolitaryVegetationObject and PlantCover (graphic: District of Recklinghausen).](image)

A SolitaryVegetationObject may have the attributes class, species, function, height, trunkDiameter and crownDiameter. The attribute class contains the coarse classification of the object or plant habit, e.g. tree, bush, grass, and can occur only once (see external code list in chapter 10.6.4 and annex C.5). The attribute species defines the species’ name, for example “Abies alba”, and can occur at most once (see external code list in chapter 10.6.4 and annex C.5). The hierarchy between class and species is not reflected in the external code lists, thus inconsistencies have to be checked by application tools. The optional attribute function denotes the purpose of the object, for example botanical monument, and can occur multiple times. The attribute height contains the relative height of the object. The attributes crownDiameter and trunkDiameter represent the plant crown and trunk diameter respectively. The trunk diameter is often used in regulations of municipal cadastre (e.g. tree management rules).

A PlantCover feature may have the attributes class, function and averageHeight. The plant community of a PlantCover is represented by the attribute class. The values of this attribute are enumerated in an external code list (cf. chapter 10.6.4 and annex C.5), where each value describes not only one plant type or species, but denotes a typical mixture of plant types in a plant community. This information can be used in particular to generate realistic 3D visualisations, where the PlantCover region is automatically, perhaps randomly, filled with a corresponding mixture of 3D plant objects. The attribute function indicates the purpose of the object, for example national forest, and can occur multiple times. The attribute averageHeight denotes the average relative vegetation height.
Since both SolitaryVegetationObject and PlantCover are CityObjects, they inherit all attributes of a city object, in particular a name (gml:name) and an ExternalReference to a corresponding object in an external information system, which may contain botanical information from public environmental agencies (see chapter 6.7).

The geometry of a SolitaryVegetationObject may be defined in LOD 1-4 explicitly by a GML geometry having absolute coordinates, or prototypically by an ImplicitGeometry (see chapter 8.2). Solitary vegetation objects probably are one of the most important features where implicit geometries are appropriate, since the shape of the most types of vegetation objects, such as trees of the same species, can be treated as identical in most cases. Furthermore, season dependent appearances may be mapped using ImplicitGeometries. For visualisation purposes, only the content of the library object defining the object’s shape and appearance has to be swapped (cf. Fig. 40).

A SolitaryVegetationObject or a PlantCover may have a different geometry in each LOD. Whereas a SolitaryVegetationObject is associated with the _Geometry class representing an arbitrary GML geometry (by the relation lodXGeometry, X ∈ [1..4]), a PlantCover is restricted to be either a MultiSolid or a MultiSurface. An example of a PlantCover modelled as MultiSolid is a ‘solid forest model’, see Fig. 41.
Fig. 41: Example for the visualisation/modelling of a solid forest (source: District of Recklinghausen).

XML namespace

The XML namespace of the CityGML Vegetation module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/vegetation/1.0. Within the XML Schema definition of the Vegetation module, this URI is also used to identify the default namespace.

10.6.1 Vegetation object

AbstractVegetationObjectType, _VegetationObject

```
<xs:complexType name="AbstractVegetationObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

10.6.2 Solitary vegetation objects

SolitaryVegetationObjectType, SolitaryVegetationObject

```
<xs:complexType name="SolitaryVegetationObjectType">
  <xs:complexContent>
    <xs:extension base="AbstractVegetationObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfSolitaryVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
The following two excerpts of a CityGML dataset contain a plant community (PlantCover). The solitary tree has the attributes: class = 1070 (deciduous tree), species = 1040 (Fagus/beech), height = 8 m, trunkDiameter = 0.7 m, crownDiameter = 8.0 m. The plant community has the attributes: class = 1180 (isoeto-nanojuncetea), averageHeight = 0.5 m.
10.6.6 Conformance requirements

Referential integrity

- The $lodX$ImplicitRepresentation, $X \in [1..4]$, property (type: $core:ImplicitRepresentationPropertyType$) of the element $SolitaryVegetationObject$ may contain a $core:ImplicitGeometry$ element inline or an XLink reference to a remote $core:ImplicitGeometry$ element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the $lodX$ImplicitRepresentation, $X \in [1..4]$, property may only point to a remote $core:ImplicitGeometry$ element (where remote $core:ImplicitGeometry$ elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.7 City furniture

City furniture objects are immovable objects like lanterns, traffic lights, traffic signs, flower buckets, advertising columns, benches, delimitation stakes, or bus stops (Fig. 42, Fig. 43). City furniture objects can be found in traffic areas, residential areas, on squares or in built-up areas. The modelling of city furniture objects is used for visualisation of, for example city traffic, but also for analysing local structural conditions. The recognition of special locations in a city model is improved by the use of these detailed city furniture objects, and the city model itself becomes more alive and animated. The city furniture model of CityGML is defined by the thematic extension module CityFurniture (cf. chapter 7).

City furniture objects can have an important influence on simulations of, for example city traffic situations. Navigation systems can be realised, for example for visually handicapped people using a traffic light as routing target. Or city furniture objects are important to plan a heavy vehicle transportation, where the exact position and further conditions of obstacles must be known.

![Fig. 42: Real situation showing a bus stop (left). The advertising billboard and the refuge are modelled as CityFurniture objects in the right image (source: 3D city model of Barkenberg).](image)

![Fig. 43: Real situation showing lanterns and delimitation stakes (left). In the right image they are modelled as CityFurniture objects with ImplicitGeometries (source: 3D city model of Barkenberg).](image)

The UML diagram of the city furniture model is depicted in Fig. 44, for XML schema definition see below and annex A.4.

The class CityFurniture may have the attributes class and function. Their possible values are specified in the respective external code lists (chapter 10.7.2 and annex C.2). The class attribute allows an object classification like traffic light, traffic sign, delimitation stake, or garbage can, and can occur only once. The function attribute describes, to which thematic area the city furniture object belongs (e.g. transportation, traffic regulation, architecture etc.), and can occur multiple times. The hierarchy between class and function is not reflected in the external code lists. Inconsistencies have to be checked by the application tools.
Fig. 44: UML diagram of city furniture objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML CityFurniture module.

Since CityFurniture is a subclass of _CityObject and hence is a feature, it inherits the attribute gml:name. As with any _CityObject, CityFurniture objects may be assigned ExternalReferences (cf. chapter 6.7) and may be augmented by GenericAttributes using CityGML’s Generics module (cf. chapter 10.10). For ExternalReferences city furniture objects can have links to external thematic databases. Thereby, semantical information of the objects, which can not be modelled in CityGML, can be transmitted and used in the 3D city model for further processing, for example information from systems of powerlines or pipelines, traffic sign cadaster, or water resources for disaster management.

City furniture objects can be represented in city models with their specific geometry, but in most cases the same kind of object has an identical geometry. The geometry of CityFurniture objects in LOD 1-4 may be represented by an explicit geometry (lodXGeometry where X is between 1 and 4) or an ImplicitGeometry object (lodXImplicitRepresentation with X between 1 and 4). In the concept of ImplicitGeometry the geometry of a prototype city furniture object is stored only once in a local coordinate system and referenced by a number of features (see chapter 8.2). Spatial information of city furniture objects can be taken from city maps (called “Stadtgrundkarte” in Germany) or from public and private external information systems.

In order to specify the exact intersection of the DTM with the 3D geometry of a city furniture object, the latter can have a TerrainIntersectionCurve (TIC) for each LOD (cf. chapter 6.5). This allows for ensuring a smooth transition between the DTM and the city furniture object.

XML namespace

The XML namespace of the CityGML CityFurniture module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/cityfurniture/1.0. Within the XML Schema definition of the CityFurniture module, this URI is also used to identify the default namespace.

10.7.1 City furniture object

CityFurnitureType, CityFurniture

```xml
<xs:complexType name="CityFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="CityFurnitureClassType" minOccurs="0"/>
        <xs:element name="function" type="CityFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
10.7.2 External code lists

The city furniture model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.2):

- CityFurnitureFunctionType
- CityFurnitureClassType

10.7.3 Example CityGML dataset

The following example of a CityGML dataset is an extract of the modelling of a delimitation stake in LOD3 and contains the attributes: class = 1000, function = 1520 (delimitation stake). The delimitation stake with the object ID stake0815 is referencing by urn:adv:oid:DEHE123400007001 to an cadastre object in the German ALKIS database (www.adv-online.de).

This example shows the geometry of Cover Surface (on the top of the stake) and of the left Surface left (Fig. 45). The Cover Surface has the material (color) white and the Surface left has the texture stake.gif with the relevant texture coordinates.

![stake.png](http://www.adv-online.de)/stake.gif

Fig. 45: Example of a simple city furniture object (source: District of Recklinghausen).

```xml
<xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
<xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
<xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
<xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
<xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
<xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
<xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
<xs:element ref="_GenericApplicationPropertyOfCityFurniture" type="xs:anyType" abstract="true"/>
```
10.7.4 Conformance requirements

Referential integrity

- The lodXImplicitRepresentation, X ∈ [1..4], property (type: core:ImplicitRepresentationPropertyType) of the element CityFurniture may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, X ∈ [1..4], property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.8 Land use

LandUse objects describe areas of the earth’s surface dedicated to a specific land use. They can be employed to represent parcels in 3D. Fig. 46 shows the UML diagram of land use objects, for the XML schema definition see chapter 10.8.1 and annex A.7. The land use model of CityGML is provided by the thematic extension module LandUse (cf. chapter 7).

![UML diagram of land use objects in CityGML](image)

Every LandUse object may have the attributes class, function, and usage. The class attribute is used to represent the classification of land use objects, like settlement area, industrial area, farmland etc., and can occur only once. The possible values are specified in an external code list (see annex C.3). The attribute function defines the purpose of the object, like e.g. cornfield, while the attribute usage can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The LandUse object is defined for all LOD 0-4 and may have different geometries in any LOD. The surface geometry of a LandUse object is required to have 3D coordinate values. It must be a GML3 MultiSurface, which might be assigned appearance properties like textures or colors (using CityGML’s appearance model, cf. chapter 9).

LandUse objects can be employed to establish a coherent geometric/semantical tesselation of the earth’s surface. In this case topological relations between neighbouring LandUse objects should be made explicit by defining the boundary LineStrings only once and by referencing them in the corresponding Polygons using XLinks (cf. chapter 8.1). Fig. 47 shows a land use tesselation, where the geometries of the land use objects are represented as triangulated surfaces. In fact, they are the result of a constrained triangulation of a DTM with consideration of breaklines defined by a 2D vector map of land use classifications.

![LOD0 regional model consisting of land use objects in CityGML](image)
XML namespace
The XML namespace of the CityGML LandUse module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/landuse/1.0. Within the XML Schema definition of the LandUse module, this URI is also used to identify the default namespace.

10.8.1 Land use object

LandUseType, LandUse

```xml
<x:s:complexType name="LandUseType">
  <x:s:extension base="core:AbstractCityObjectType">
    <x:s:element name="class" type="LandUseClassType" minOccurs="0"/>
    <x:s:element name="function" type="LandUseFunctionType" minOccurs="0" maxOccurs="unbounded"/>
    <x:s:element name="usage" type="LandUseUsageType" minOccurs="0" maxOccurs="unbounded"/>
    <x:s:element ref="_GenericApplicationPropertyOfLandUse" minOccurs="0" maxOccurs="unbounded"/>
  </x:s:extension>
  </x:s:complexType>
</x:s:complexType>
```

10.8.2 External code lists

The land use model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 6.6 and annex C.3):

- LandUseClassType
- LandUseFunctionType
- LandUseUsageType

10.8.3 Conformance requirements

Base requirements

- The gml:MultiSurface geometry element representing the geometry of a LandUse object must be given with 3D coordinate values within each LOD.
10.9 City object groups

The grouping concept has already been introduced in chapter 6.8. CityObjectGroups are modelled using the Composite Design Pattern from software engineering (cf. Gamma et al. 1995): CityObjectGroups aggregate CityObjects and furthermore are defined as special CityObjects. This implies that a group may become a member of another group realizing a recursive aggregation schema. However, in a CityGML instance document it has to be ensured (by the generating application) that no cyclic groupings are included. Fig. 48 shows the UML diagram for the class CityObjectGroup, for the XML schema see annex A.5. The grouping concept of CityGML is defined by the thematic extension module CityObjectGroup (cf. chapter 7).

The class CityObjectGroup has the optional attributes class, function and usage. In contrast to the other thematic classes, no code lists are defined for these attributes, because the reasons for groupings can not completely be foreseen. The class attribute allows a group classification with respect to the stated function and may occur only once. The function attribute is intended to express the main purpose of a group, possibly to which thematic area it belongs (e.g. site, building, transportation, architecture, unknown etc.). The attribute usage can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times. Each member of a group may be qualified by a role name, reflecting the role each CityObject plays in the context of the group. Furthermore, a CityObjectGroup can optionally be assigned an arbitrary geometry object from the GML3 subset shown in Fig. 8 in chapter 8.1. This may be used to represent a generalised geometry generated from the members geometries.

The parent association linking a CityObjectGroup to a CityObject allows for the modelling of a generic hierarchical grouping concept. Named aggregations of components (CityObjects) can be added to specific CityObjects considered as the parent object. The parent association links to the aggregate, while the parts are given by the group members. This concept is used, for example, to represent storeys in buildings (see section 10.3.6: Modelling building storeys using CityObjectGroups).

XML namespace

The XML namespace of the CityGML CityObjectGroup module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/cityobjectgroup/1.0. Within the XML Schema definition of the CityObjectGroup module, this URI is also used to identify the default namespace.

10.9.1 City object group

CityObjectGroupType, CityObjectGroup

```xml
<xs:complexType name="CityObjectGroupType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="role" type="xs:string" maxOccurs="1" />
        <xs:element name="class" type="xs:string" maxOccurs="1" />
        <xs:element name="function" type="xs:string" maxOccurs="1" />
        <xs:element name="usage" type="xs:string" maxOccurs="1" />
        <xs:element name="geometry" type="gml:GeometryType" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
10.9.2 Conformance requirements

Base requirements

1. No cyclic groupings shall be included within a CityGML instance document.

Referential integrity

2. The groupMember property (type: CityObjectGroupMemberType) of the element CityObjectGroup may contain a core:_CityObject element inline or an XLink reference to a remote core:_CityObject element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the groupMember property may only point to a remote core:_CityObject element (where remote core:_CityObject elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

3. The parent property (type: CityObjectGroupParentType) of the element CityObjectGroup may contain a core:_CityObject element inline or an XLink reference to a remote core:_CityObject element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the parent property may only point to a remote core:_CityObject element (where remote core:_CityObject elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
### 10.10 Generic city objects and attributes

The concept of generic city objects and attributes allows for the storage and exchange of 3D objects which are not covered by any explicitly modelled thematic class within CityGML or which require attributes not represented in CityGML. These generic extensions to the CityGML data model are realised by the classes `GenericCityObject` and `GenericAttribute` defined within the thematic extension module `Generics` (cf. chapter 7). In order to avoid problems concerning semantic interoperability, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

Fig. 49 shows the UML diagram of generic objects and attributes. For XML schema definition see below and annex A.6.

A `GenericCityObject` may have the attributes `class`, `function` and `usage` defined as `string`. The `class` attribute allows an object classification within the thematic area such as bridge, tunnel, pipe, power line, dam, or unknown. The `function` attribute describes to which thematic area the `GenericCityObject` belongs (e.g. site, transportation, architecture, energy supply, water supply, unknown etc.). The attribute `usage` can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

In order to represent generic attributes, the abstract base class `GenericAttribute` defined within the CityGML Core module is augmented by the additional property element `genericAttribute` using CityGML’s Application Domain Extension mechanism (cf. chapter 6.12). By this means, each thematic subclass of `GenericCityObject` inherits this property and, thus, may be assigned an arbitrary number of generic attributes in order to represent additional properties of features not represented by the explicitly modelled thematic classes of the CityGML data model. Thus, the `Generics` module has a deliberate impact on all CityGML extension modules defining thematic subclasses of `GenericCityObject`. Data types of generic attributes may be `String`, `Integer`, `Double` (floating point number), `URI` and `Date`. The attribute type is defined by the selection of the particular subclass of `GenericAttribute`, for example `StringAttribute`, `IntAttribute`, etc.

The geometry of a `GenericCityObject` can either be an explicit GML3 geometry or an `ImplicitGeometry` (see chapter 8.2). In the case of an explicit geometry the object can have only one geometry for each LOD, which may be an arbitrary 3D GML geometry object (class `Geometry`, which is the base class of all GML geometries, `lodXGeometry`, X ∈ [0..4]). Absolute coordinates according to the reference system of the city model must be given for the explicit geometry. In the case of an `ImplicitGeometry`, a reference point (anchor point) of the object...
and optionally a transformation matrix must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates must be added. The shape of an ImplicitGeometry can be given as an external resource with a proprietary format, e.g. a VRML or DXF file from a local file system or an external web service. Alternatively the shape can be specified as a 3D GML3 geometry with local cartesian coordinates using the property relativeGeometry (further details are given in chapter 8.2).

In order to specify the exact intersection of the DTM with the 3D geometry of a GenericCityObject, the latter can have TerrainIntersectionCurves for every LOD (cf. chapter 6.5). This is important for 3D visualisation but also for certain applications like driving simulators. For example, if a bridge should be represented as a GenericCityObject, a smooth transition between the DTM and the road on the bridge would have to be ensured (in order to avoid unrealistic bumps).

XML namespace

The XML namespace of the CityGML Generics module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/generics/1.0. Within the XML Schema definition of the Generics module, this URI is also used to identify the default namespace.

10.10.1 Generic city object

GenericCityObjectType, GenericCityObject

```
<xs:complexType name="GenericCityObjectType">
  <xs:complexContent>
    <xs:extension name="GenericCityObject" type="GenericCityObjectType" substitutionGroup="#core:CityObject"/>
  </xs:complexContent>
</xs:complexType>
```

10.10.2 Generic attributes

AbstractGenericAttributeType, _genericAttribute, StringAttributeType, stringAttribute, etc.

```
<xs:complexType name="AbstractGenericAttributeType" abstract="true">
  <xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
</xs:complexType>
```
10.10.3 Conformance requirements

Usage restriction for generic city objects and attributes

1. The *GenericCityObject* element may only be used to model thematic city objects which are not provided by any other CityGML module and, thus, are not covered by the overall CityGML data model. If an appropriate thematic class is available though, this thematic class shall be used instead, and consequently, the corresponding CityGML module has to be employed by the CityGML instance document.

2. The _genericAttribute_ property of the element _core:_CityObject shall only be used to describe additional properties of features not represented by the explicitly modelled thematic classes of the CityGML data model. Thus, generic attributes shall only be modelled if the appropriate thematic class representing the feature does not offer a feasible property.
Referential integrity

3. The $lodXImplicitRepresentation$, $X \in [0..4]$, property (type: core:ImplicitRepresentationPropertyType) of the element GenericCityObject may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the $lodXImplicitRepresentation$, $X \in [0..4]$, property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.11 Application Domain Extensions (ADE)

CityGML has been designed as an application independent information model and exchange format for 3D city and landscape models. However, specific applications typically have additional information needs to be modelled and exchanged. In general, there are two different approaches to combine city model data and application data:

1. Embed the CityGML objects into a (larger) application framework and establish the connection between application data and CityGML data within the application framework. For example, CityGML data fragments may be embedded into the application’s XML data files or stored as attributes of application objects according to the application’s data model.

2. Incorporate application specific information into the CityGML instance documents. This approach is especially feasible, if the application specific information follows essentially the same structure as defined by the CityGML schema. This is the case, if the application data could be represented by additional attributes of CityGML objects and only some new feature types would have to be defined.

In the following, we will focus on the second option, as only this approach lies within the scope of this specification. Generic attributes and objects have been already introduced as a first possibility to support the exchange of application specific data (see section 10.10). Whereas they allow to extend CityGML without changing its XML schema definition, this flexibility has some disadvantages:

- Generic attributes and objects may occur arbitrarily in the CityGML instance documents, but there is no formal specification of the names, datatypes, and multiplicities. Thus, there is no guarantee for an application that a specific instance of a generic attribute is included a minimum or maximum number of times per CityGML feature. Unlike the predefined CityGML objects, the concrete layout and occurrence of generic objects and attributes cannot be validated by an XML parser. This may reduce semantic interoperability.

- Naming conflicts of generic attributes or objects can occur, if the CityGML instance documents should be augmented by specific information from different applications simultaneously.

- There is only a limited number of predefined data types that can be used for generic attributes. Also the structure of generic objects might not be appropriate to represent more complex objects.

If application specific information are well-structured, it is desirable to represent them in a systematic way, i.e. by the definition of an extra formal schema based on the CityGML schema definitions. Such an XML schema is called a CityGML Application Domain Extension (ADE). It allows to validate instance documents both against the extended CityGML and the ADE schema and therefore helps to maintain semantic and syntactic interoperability between different systems working in the same application field. In order to prevent naming conflicts, every ADE has to be defined within its own namespace which must differ from the namespaces associated with the CityGML modules. An ADE schema may extend one or more CityGML module schemas. The relevant CityGML module schemas have to be imported by the ADE schema.

The ADE concept defines a special way of extending existing CityGML feature types which allows to use different ADEs within the same instance document simultaneously (see below). For example, the specification of ADEs can be useful in the following application fields: cultural heritage (extension of abstract class _CityObject e.g. by time period information and monument protection status); representation of subsurface objects (tunnel, underpass) or city lighting (light sources like street lamps and house lights); real estate management (economic parameters of the CityGML features; inclusion of attributes defined for real estate assets as defined by OSCRE); utility networks (as topographic features); additional building properties as defined by the U.S. national building information model standard (NBIMS).

10.11.1 Technical principle of ADEs

Each ADE is specified by its own XML schema file. The target namespace is provided by the information community who specifies the CityGML ADE. This is typically not the OGC or the SIG 3D. The namespace should be in the control of this information community and must be given as a previously unused and globally
unique URI. This URI will be used in CityGML ADE instance documents to distinguish extensions from CityGML base elements. As the URI refers to the information community it also denotes the originator of the employed ADE.

The ADE’s XML schema file must be available (or accessible on the Internet) to everybody creating and parsing CityGML instance documents including these ADE specific augmentations.

An ADE XML schema can define various extensions to CityGML. However, all extensions shall belong to one of the two following categories:

1. New feature types are defined within the ADE namespace and are based on CityGML abstract or concrete classes. In general, this mechanism follows the same principles as the definition of application schemas for GML. This means, that new feature types have to be derived from existing (here: CityGML) feature types. For example, new feature types could be defined by classes derived from the abstract classes like _CityObject or _AbstractBuilding or the concrete class CityFurniture. The new feature types then automatically inherit all properties (i.e. attributes) and associations (i.e. relations) from the respective CityGML superclasses.

2. Existing CityGML feature types are extended by application specific properties (in the ADE namespace). These properties may have simple or complex data types. Also geometries or embedded features (feature properties) are possible. The latter can also be used to model relations to other features.

In this case, extension of the CityGML feature type is not being realised by the inheritance mechanism of XML schema. Instead, every CityGML feature type provides a “hook” in its XML schema definition, that allows to attach additional properties to it by ADEs. This “hook” is implemented as a GML property of the form “_GenericApplicationPropertyOf<Featuretypename>” where <Featuretypename> is equal to the name of the feature type definition in which it is included. The datatype for these kinds of properties is always “xsd:anyType” from the XSD namespace. The minimum occurrence of the “_GenericApplicationPropertyOf<Featuretypename>” is 0 and the maximum occurrence unbounded. This means, that the CityGML schema allows that every CityGML feature may have an arbitrary number of additional properties with arbitrary XML content with the name “_GenericApplicationPropertyOf<Featuretypename>”. For example, the last property in the definition of the CityGML feature type LandUse is the element _GenericApplicationPropertyOfLandUse (see section 10.8.1).

Such properties are called “hooks” to attach application specific properties, because they are used as the head of a substitution group by ADEs. Whenever an ADE wants to add an extra property to an existing CityGML feature type, it should declare the respective element with the appropriate datatype within the ADE namespace. In the element declaration this element shall be explicitly assigned to the substitution group defined by the corresponding “_GenericApplicationPropertyOf<Featuretypename>” in the corresponding CityGML module namespace. An example is given in the following subsection.

By following this concept, it is possible to specify different ADEs for different information communities. Every ADE may add their specific properties to the same CityGML feature type as they all can belong to the same substitution group. This allows to have CityGML instance documents where CityGML features contain additional information from different ADEs simultaneously.

Please note that usage of ADEs introduces an extra level of complexity as data files may contain mixed information (features, properties) from different namespaces, not only from the GML and CityGML module namespaces. However, extended CityGML instance documents are quite easy to handle by applications that are not “schema-aware”, i.e. applications that do not parse and interpret GML application schemas in a generic way. These applications can simply skip anything from a CityGML instance document that is not from a CityGML module or GML namespace. Thus, a building is still represented by the <bldg:Building> element with the standard CityGML properties, but with possibly some extra properties from different namespaces. Also features from a different namespace than those declared by CityGML modules or GML could be skipped (e.g. by a viewer application).

10.11.2 Example ADE

In this section, the ADE mechanism is illustrated by a short example, which deals with the application of virtual 3D city models to generate noise pollution maps. In our example, two extensions of CityGML are required for
this task: buildings have to be extended to represent a “noise reflection correction” value and the number of inhabitants. As a new feature type noise barriers have to be defined which also have a “noise reflection correction” value.

The XSD schema which has to be defined to implement this model declares a new namespace for the noise extension (http://www.citygml.org/ade/noise_de). Additionally, the namespaces of the extended CityGML modules are declared (for corresponding prefixes see chapter 4.3 and chapter 7), and the respective schema definition files are imported. The XML schema adds the elements buildingReflectionCorrection and buildingHabitants, both being members of the substitution group bldg:_GenericApplicationPropertyOfAbstractBuilding which is defined by the CityGML Building module. Thus, both elements may be used as child elements of CityGML building features. Noise barriers are represented as NoiseCityFurnitureSegment elements. The corresponding type NoiseCityFurnitureSegmentType is defined as subtype of the CityGML abstract type core:AbstractCityObjectType provided by the CityGML Core module, applying the usual subtyping mechanism of XML and XSD. A further element noiseCityFurnitureSegmentProperty is added as a member of the substitution group fnr:_GenericApplicationPropertyOfCityFurniture. By this means, noise barriers may be modelled as child elements of CityGML city furniture objects.

The XSD file for this example CityGML Noise ADE is given by the following excerpt (the complete CityGML Noise ADE is given in Annex G):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.citygml.org/ade/noise_de"
  xmlns:core="http://www.opengis.net/citygml/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  xmlns:frn="http://www.opengis.net/citygml/cityfurniture/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml" targetNamespace="http://www.citygml.org/ade/noise_de" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  !--
  <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="http://www.citygml.org/citygml/1.0/cityGMLBase.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/building/1.0" schemaLocation="http://www.citygml.org/citygml/building/1.0/building.xsd"/>
  <xsd:import namespace="http://www.opengis.net/citygml/cityfurniture/1.0" schemaLocation="http://www.citygml.org/citygml/cityfurniture/1.0/cityfurniture.xsd"/>
  !--
  <xsd:element name="buildingReflectionCorrection" type="gml:MeasureType"
    substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xsd:element name="buildingHabitants" type="xsd:positiveInteger"
    substitutionGroup="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  !--
  <xsd:element name="noiseCityFurnitureSegmentProperty" type="NoiseCityFurnitureSegmentPropertyType"
    substitutionGroup="fnr:_GenericApplicationPropertyOfCityFurniture"/>
  !--
  <xsd:complexType name="NoiseCityFurnitureSegmentPropertyType">
    <xsd:sequence minOccurs="0"/>
    <xsd:attributeGroup ref="frn:NoiseCityFurnitureSegment" minOccurs="0"/>
  </xsd:complexType>
  !--
  <xsd:complexType name="NoiseCityFurnitureSegmentType">
    <xsd:complexContent>
      <xsd:extension base="core:AbstractCityObjectType">
        <xsd:sequence>
          <xsd:element name="reflection" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="reflectionCorrection" type="gml:MeasureType" minOccurs="0"/>
          <xsd:element name="height" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="distance" type="gml:LengthType" minOccurs="0"/>
          <xsd:element name="lod0BaseLine" type="gml:CurvePropertyType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  !--
  <xsd:complexType name="NoiseCityFurnitureSegment" type="NoiseCityFurnitureSegmentType" substitutionGroup="core:CityObject"/>
  !--
</xsd:schema>
```
An example for a feature collection in a corresponding instance document is depicted below. Two CityGML buildings contain application specific properties distinguished from CityGML properties by the namespace prefix `noise`. The other properties, function and geometry, are defined by corresponding CityGML modules. In addition to the buildings, a noise barrier as child of a city furniture element is included in the feature collection.

Please note, that the order of the child elements in the sequence is not arbitrary: the child elements defined by an ADE subschema have to occur after the child elements defined by CityGML modules. There is, however, no specific order of the ADE properties.

```
...<core:cityObjectMember>
  <bldg:Building gml:id="aa">
    <bldg:function>1004</bldg:function>
    <bldg:lod1Solid>...<bldg:lod1Solid>
      <noise:buildingHabitants>14</noise:buildingHabitants>
      <noise:buildingReflectionCorrection uom="dB">4.123</noise:buildingReflectionCorrection>
    </bldg:Building>
  </core:cityObjectMember>
<core:cityObjectMember>
  <bldg:Building gml:id="aaa">
    <bldg:function>1004</bldg:function>
    <bldg:lod1Solid>...<bldg:lod1Solid>
      <noise:buildingHabitants>6</noise:buildingHabitants>
      <noise:buildingReflectionCorrection uom="dB">3.123</noise:buildingReflectionCorrection>
    </bldg:Building>
  </core:cityObjectMember>
<core:cityObjectMember>
  <frn:CityFurniture gml:id="CFUR_0815">
    <frn:function>1520</frn:function>
    <frn:lod1Geometry>...<frn:lod1Geometry>
      <noise:NoiseCityFurnitureSegmentProperty>
        <noise:NoiseCityFurnitureSegment gml:id="CFRS_0815">
          <noise:type>1</noise:type>
          <noise:reflection>absorbierende Lärmschutzwand</noise:reflection>
          <noise:reflectionCorrection uom="dB">4.123</noise:reflectionCorrection>
          <noise:height uom="m">7.123</noise:height>
          <noise:distance uom="m">21.123</noise:distance>
          <noise:lod0BaseLine>
          </noise:lod0BaseLine>
        </noise:NoiseCityFurnitureSegment>
      </noise:NoiseCityFurnitureSegmentProperty>
    </frn:CityFurniture>
  </core:cityObjectMember>
...
10.12 Definition of code lists

For the representation of city object attributes having an enumerative range of values, the concept of simple dictionaries is used, which is provided as a profile based on GML (cf. chapter 6.6). For each such attribute, the values are defined in a single file, which is named according to the attribute name. These files are located in a folder named Codelists, which comes with the CityGML schema document, but is not a normative part of this schema, since it may be modified, augmented, or replaced by other communities. The actual values in the files in the folder Codelists are a suggestion of the SIG 3D.

The external code list files define attribute values and assign an unique identifier to each value. In a CityGML instance document, an attribute value is denoted by an identifier of a value, not by the value itself. Thus typos are avoided and it is ensured that the same concept is denoted the same way, by the same identifier and not by two different terms with identical meaning. Thus the use of code lists facilitates semantic and syntactic interoperability, since they define common terms within an information community. Furthermore, the simple dictionary concept enables more than one term to be assigned to the same dictionary entry, thus the same concept may be explained in different languages. To differentiate between the languages, code spaces may be used.

An example for an enumerative attribute is RoofType, which is defined by the following excerpt of the external code list file:

```xml
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/gml ../3.1.1/profiles/simpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
    gml:id="RoofTypeType">
    <name>RoofTypeType</name>
    <dictionaryEntry>
        <gml:Definition gml:id="id357">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1000</gml:name>
            <gml:name>flat roof</gml:name>
            <gml:Definition/>
        </dictionaryEntry>
        <dictionaryEntry>
            <gml:Definition gml:id="id358">
                <gml:description/>
                <gml:name codeSpace="urn:d_nrw_sig3d">1010</gml:name>
                <gml:name>monopitch roof</gml:name>
                <gml:Definition/>
            </dictionaryEntry>
            ...
    </gml:Dictionary>
```

In the simple dictionary concept, the values of an attribute are represented by a Dictionary element, where each value is given by a dictionaryEntry resp. Definition entry. In CityGML, a definition entry is identified by the name element, which is qualified by the SIG 3D code space. The unqualified name element represents the value of the attribute. An optional description explains the value. CityGML does not use GML identifiers (gml:id) to link to attribute values, since IDs are restricted syntactically, and must be globally unique, which is not feasible for code lists.
Annex A
(normative)

XML Schema definition

A.1 CityGML Core module

The CityGML Core module is defined within the XML Schema definition file cityGMLBase.xsd. The target namespace 
http://www.opengis.net/citygml/1.0 is associated with the core module.

```xml
<xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

<xs:complexType name="CityModelType">
  <xs:documentation> Type describing the "root" element of any city model file. It is a collection whose members are
  restricted to be features of a city model. All features are included as cityObjectMember. </xs:documentation>

  <xs:annotation/>

  <xs:complexContent internalSchemaLocation="/A.1.1/base/gml.xsd"/>

  <xs:complexType name="AbstractCityObjectType" abstract="true">
    <xs:documentation> Type describing the abstract superclass of most CityGML features. Its purpose is to provide a
    creation and a termination date as well as a reference to corresponding objects in other information systems. A
    generalization relation may be used to relate features, which represent the same real-world object in different
    Levels-of-Detail, i.e. a feature and its generalized counterpart(s). The direction of this relation is from the
    feature to the corresponding generalized feature. </xs:documentation>

    <xs:annotation/>

    <xs:complexContent internalSchemaLocation="/A.1.1/base/gml.xsd"/>

    <xs:element name="_CityObject" type="AbstractCityObjectType" abstract="true" substitutionGroup="gml:_FeatureCollection"/>
    <xs:element name="GeneralizationRelationType" type="gml:FeaturePropertyType" internalSchemaLocation="/A.1.1/base/gml.xsd"/>
  </xs:complexType>

  <xs:complexType name="_GenericApplicationPropertyOfCityModel">
    <xs:documentation> Generalization relations. </xs:documentation>
  </xs:complexType>

  <xs:complexType name="AbstractApplicationPropertyOfCityModel">
    <xs:documentation> Type describing the abstract superclass for buildings, facilities, etc. Future extensions of CityGML
    like bridges and tunnels would be modelled as subclasses of _Site. As subclass of _CityObject, a _Site inherits
    all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
  </xs:complexType>
</xs:schema>
```
<xs:extension base="AbstractCityObjectType">
  <xs:sequence>
    <xs:element ref="_Generic/ApplicationPropertyOfSite" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="_Site">
  <xs:complexType abstract="true">
    <xs:annotation>
      <xs:documentation>
        Denotes the relation of a _CityObject to its corresponding _CityObject in higher LOD, i.e. to the _CityObjects representing the same real world object in higher LOD. The GeneralizationRelationType element must either carry a reference to a _CityObject object or contain a _CityObject object inline, but neither both nor none.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          Type describing the reference to an corresponding object in an other information system, for example in the german cadastrs ALKIS, the german topographic information system or ATKIS, or the OS MasterMap. The reference consists of the name of the external information system, represented by an URL, and the reference of the external object, given either by a string or by an URI. If the informationSystem element is missing in the ExternalReference, the ExternalObjectReference must be an URI, which contains an indication of the informationSystem.
        </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:restriction base="gml:AssociationType">
          <xs:sequence minOccurs="0">
            <xs:element ref="_CityObject"/>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </xs:complexType>
</xs:element>

<xs:complexType name="GeneralizationRelationType">
  <xs:documentation>
    Denotes the relation of a _CityObject to its corresponding _CityObject in higher LOD, i.e. to the _CityObjects representing the same real world object in higher LOD. The GeneralizationRelationType element must either carry a reference to a _CityObject object or contain a _CityObject object inline, but neither both nor none.
  </xs:documentation>
</xs:complexType>

<xs:complexType name="ExternalReferenceType">
  <xs:annotation>
    <xs:documentation>
      Type describing the reference to an corresponding object in an other information system, for example in the german cadastrs ALKIS, the german topographic information system or ATKIS, or the OS MasterMap. The reference consists of the name of the external information system, represented by an URL, and the reference of the external object, given either by a string or by an URI. If the informationSystem element is missing in the ExternalReference, the ExternalObjectReference must be an URI, which contains an indication of the informationSystem.
    </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType name="ExternalObjectReferenceType">
  <xs:choice>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="uri" type="xs:anyURI"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="AddressPropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of an _CityObject to its addresses. The AddressPropertyType element must either carry a reference to an Address object or contain an Address object inline, but neither both nor none.
    </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType name="AddressType">
  <xs:annotation>
    <xs:documentation>
      Type for addresses. It references the xAL address standard issued by the OASIS consortium. Please note, that addresses are modelled as GML features. Every address can be assigned zero or more 2D or 3D point geometries (one gml:MultiPoint geometry) locating the entrance(s).
    </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType base="gml:AbstractFeatureType">
  <xs:sequence>
    <xs:element name="_Address" type="xal:AddressPropertyType"/>
    <xs:element name="multiPoint" type="gml:MultiPointPropertyType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="_Generic/ApplicationPropertyOfAddress" minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>
ImplicitGeometry type for the implicit representation of a geometry. An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, e.g. a tree or other vegetation object, a traffic light or a traffic sign. This prototypic geometry object is re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cartesian coordinate system), by a transformation matrix that is multiplied with each 3D coordinate tuple of the prototype, and by an anchor point denoting the base point of the object in the world coordinate reference system. In order to determine the absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way even a projection might be modelled by the transformation matrix. The concept of implicit geometries is an enhancement of the geometry model of GML3.
encoding, for example. </xs:documentation>
</xs:annotation>
<xs:list itemType="doubleBetween0and1"/>
</xs:restriction>
</xs:simpleType>
</xs:annotation>
</xs:simpleType>
<xs:simpleType name="TransformationMatrix4x4Type">
<xs:annotation>
<xs:documentation>Used for implicit geometries. The Transformation matrix is a 4 by 4 matrix, thus it must be a list with 16 items. The order the matrix element are represented is row-major, i.e. the first 4 elements represent the first row, the fifth to the eight element the second row,...</xs:documentation>
</xs:annotation>
<xs:restriction base="gml:doubleList">
<xs:length value="16"/>
</xs:restriction>
</xs:simpleType>
</xs:annotation>
</xs:simpleType>
<xs:simpleType name="TransformationMatrix2x2Type">
<xs:annotation>
<xs:documentation>Used for georeferencing. The Transformation matrix is a 2 by 2 matrix, thus it must be a list with 4 items. The order the matrix element are represented is row-major, i.e. the first 2 elements represent the first row, the fifth to the eight element the second row,...</xs:documentation>
</xs:annotation>
<xs:restriction base="gml:doubleList">
<xs:length value="4"/>
</xs:restriction>
</xs:simpleType>
</xs:annotation>
</xs:simpleType>
<xs:simpleType name="TransformationMatrix3x4Type">
<xs:annotation>
<xs:documentation>Used for texture parameterization. The Transformation matrix is a 3 by 4 matrix, thus it must be a list with 12 items. The order the matrix element are represented is row-major, i.e. the first 4 elements represent the first row, the fifth to the eight element the second row,...</xs:documentation>
</xs:annotation>
<xs:restriction base="gml:doubleList">
<xs:length value="12"/>
</xs:restriction>
</xs:simpleType>
</xs:annotation>
<xs:simpleType name="integerBetween0and4">
<xs:annotation>
<xs:documentation>Type for integer values, which are greater or equal than 0 and less or equal than 4. Used for encoding of the LOD number.</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:integer">
<xs:minInclusive value="0"/>
<xs:maxInclusive value="4"/>
</xs:restriction>
</xs:simpleType>
</xs:annotation>
<xs:simpleType name="MimeTypeType">
<xs:annotation>
<xs:documentation>MIME type of a geometry in an external library file. MIME types are defined by the IETF (Internet Engineering Task Force). The values of this type are defined in the XML file MimeTypeType.xml, according to the dictionary concept of GML3.</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:annotation>
</xs:schema>
A.2 Appearance module

The CityGML Appearance module is defined within the XML Schema definition file appearance.xsd. The target namespace http://www.opengis.net/citygml/appearance/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/appearance/1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.opengis.net/citygml/appearance/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:complexType name="AppearanceType">
    <xs:annotation>
      <xs:documentation>Named container for all surface data (texture/material). All appearances of the same name ("theme") within a CityGML file are considered a group. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="theme" type="xs:string" minOccurs="0"/>
          <xs:element name="surfaceDataMember" type="SurfaceDataPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="AppearancePropertyType">
    <xs:documentation>Denotes the relation of a _CityObject resp. CityModel to its appearances. The AppearancePropertyType element must either carry a reference to a Appearance object or contain a Appearance object inline, but neither both nor none. </xs:documentation>
    <xs:annotation>
      <xs:documentation>
        <xs:element name="appearanceMember" type="AppearancePropertyType" substitutionGroup="gml:featureMember"/>
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:FeaturePropertyType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element name="Appearance" type="AppearanceType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="AbstractSurfaceDataPropertyType">
    <xs:annotation>
      <xs:documentation>Base class for textures and material. Contains only isFront-flag. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="isFront" type="xs:boolean" default="true" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="SurfaceDataPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of an Appearance to its surface data. The SurfaceDataPropertyType element must either carry a reference to a _SurfaceData object or contain a _SurfaceData object inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
  </xs:complexType>
</xs:schema>
```
<xs:complexType name="AbstractTextureType">
  <xs:documentation>Base class for textures. "imageURI" can contain any valid URI from references to a local file to preformatted web service requests. The linking to geometry and texture parameterization is provided by derived classes.</xs:documentation>
</xs:complexType>

<xs:complexType name="AbstractSurfaceDataType">
  <!-- Base class for <gml:AbstractSurfaceType> and its decendants of gml:AbstractSurfaceType are valid targets. As property of the link, a texture parameterization is provided by </xs:documentation>  
</xs:complexType>

<xs:complexType name="ParameterizedTextureType">
  <xs:advertisement>Specialization for georeferenced textures, i.e. textures using a planimetric projection. Such textures contain an implicit parameterization (either stored within the image file, in an accompanying world file, or using the "referencePoint" and "orientation"-elements). A georeference provided by "referencePoint" and "orientation" always takes precedence. The search order for an external georeference is determined by the boolean flag preferWorldFile. If this flag is set to true (its default value), a world file is looked for first and only if it is not found the georeference from the image data is used. If preferWorldFile is false, the world file is used only if no georeference from the image data is available. The "boundedBy"-property should contain the bounding box of the projected image data. Since a georeferenced texture has a unique parameterization, "target" only provides links to surface geometry without any additional texture parameterization. Only gml:MultiSurface or decendants of gml:AbstractSurfaceType are valid targets.</xs:advertisement>
</xs:complexType>
<xs:element name="textureCoordinates" type="gml:doubleList"/>
</xs:extension>
</xs:complexType>

<xs:complexType name="TexCoordListType">
  <xs:annotation>
    <xs:documentation>
      Texture parameterization using texture coordinates: Each gml:LinearRing that is part of the surface requires a separate "textureCoordinates"-entry with 2 doubles per ring vertex. The "ring"-attribute provides the gml:id of the target LinearRing. It is prohibited to link texture coordinates to any other object type than LinearRing. Thus, surfaces not consisting of LinearRings cannot be textured this way. Use transformation matrices (see below) or georeferenced textures instead.
    </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType name="AbstractTextureParameterizationType" abstract="true">
  <xs:annotation>
    <xs:documentation>
      Base class for augmenting a link "texture" to a surface with texture parameterization. Subclasses of this class define concrete parameterizations. Currently, texture coordinates and texture coordinate generation using a transformation matrix are available.
    </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true" substitutionGroup="_Texture">
  <xs:documentation>
    Denotes the relation of a texture to a surface, that is augmented by a TextureParameterization object. The TextureAssociationType element must either carry a reference to a _TextureParameterization object or contain a TextureParameterization object inline, but neither both nor none.
  </xs:documentation>
</xs:complexType>

<xs:complexType name="GeoreferencedTextureType">
</xs:complexType>

<xs:complexType name="GenericApplicationPropertyOfGeoreferencedTexture" type="xs:anyType" abstract="true" substitutionGroup="#GenericApplicationPropertyOfTextureParameterization">
  <xs:documentation>
    Using a transformation matrix are available.
  </xs:documentation>
</xs:complexType>

<xs:complexType name="AbstractTextureParameterizationType" type="gml:AbstractGMLType">
  <xs:documentation>
    For augmenting a link "texture" to a surface with texture parameterization. Subclasses of this class define concrete parameterizations. Currently, texture coordinates and texture coordinate generation using transformation matrices are available.
  </xs:documentation>
</xs:complexType>

<xs:element name="textureCoordinates" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true" substitutionGroup="#AbstractTextureParameterizationType">
  <xs:documentation>
    Using a transformation matrix are available.
  </xs:documentation>
</xs:complexType>

<xs:element name="textureCoordinates" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true" substitutionGroup="#AbstractTextureParameterizationType">
  <xs:documentation>
    Using a transformation matrix are available.
  </xs:documentation>
</xs:complexType>

<xs:element name="textureCoordinates" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true" substitutionGroup="#AbstractTextureParameterizationType">
  <xs:documentation>
    Using a transformation matrix are available.
  </xs:documentation>
</xs:complexType>

<xs:element name="textureCoordinates" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="_TextureParameterization" type="AbstractTextureParameterizationType" abstract="true" substitutionGroup="#AbstractTextureParameterizationType">
  <xs:documentation>
    Using a transformation matrix are available.
  </xs:documentation>
</xs:complexType>

<xs:element name="textureCoordinates" maxOccurs="unbounded"/>
</xs:complexType>
Textures can be qualified by the attribute textureType. The textureType differentiates between textures, which are specific for a certain object and are only used for that object (specific), and prototypic textures being typical for that kind of object and are used many times for all objects of that kind (typical). A typical texture may be replaced by a specific, if available. Textures may also be classified as unknown.

Additional "isSmooth" provides a hint for value interpolation. The link to surface geometry is established via the "target" property. Only gml:MultiSurface or descendents of gml:AbstractSurfaceType are valid targets.

Perspective transformation matrix "worldToTexture" can be used to derive texture coordinates from an object's location. This 3x4 matrix T computes the coordinates (s,t) from a homogeneous world position p as (s,t) = (s/q', t/q') with (s', t', q') = T*p. Thus, perspective projections can be specified. The SRS can be specified using standard attributes. If an object is given in a different reference system, it is transformed to the SRS before applying the transformation. A transformation matrix can be used for whole surfaces. It is not required to specify it per LinearRing.
<xs:simpleType name="Color">
  <xs:annotation>
    <xs:documentation>List of three values (red, green, blue), separated by spaces. The values must be in the range between zero and one.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="core:doubleBetween0and1List">
    <xs:length value="3"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ColorPlusOpacity">
  <xs:annotation>
    <xs:documentation>List of three or four values (red, green, blue, opacity), separated by spaces. The values must be in the range between zero and one. If no opacity is given, it is assumed as 1.0.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="core:doubleBetween0and1List">
    <xs:minLength value="3"/>
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
A.3 Building module

The CityGML Building module is defined within the XML Schema definition file building.xsd. The target namespace http://www.opengis.net/citygml/building/1.0 is associated with this extension module.

```xml
<xs:element name="Building" type="AbstractBuildingType" minOccurs="0" maxOccurs="unbounded"/>
```

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
<xs:complexType name="AbstractBuilding">
  <xs:complexContent>
    <xs:extension base="AbstractBuildingType">
      <xs:element ref="_GenericApplicationPropertyOfBuilding" minOccurs="0" maxOccurs="unbounded"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="GenericApplicationPropertyOfAbstractBuilding">
  <xs:complexContent>
    <xs:extension base="AbstractBuildingType">
      <xs:element ref="_GenericApplicationPropertyOfBuilding" minOccurs="0" maxOccurs="unbounded"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="Building">
  <xs:complexContent>
    <xs:extension base="AbstractBuilding">  
      <xs:element ref="_GenericApplicationPropertyOfBuilding" minOccurs="0" maxOccurs="unbounded"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BuildingPart">
  <xs:complexContent>
    <xs:extension base="AbstractBuildingType">
      <xs:element ref="_GenericApplicationPropertyOfBuildingPart" minOccurs="0" maxOccurs="unbounded"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BuildingPartProperty">
  <xs:documentation>Denotes the relation of an _AbstractBuilding to its building parts. The BuildingPartPropertyType element must either carry a reference to a BuildingPart object or contain a BuildingPart object inline, but neither both nor none.</xs:documentation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="_GenericApplicationPropertyOfBuildingPart" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:complexType>
A BuildingInstallation is a part of a Building which has not the significance of a BuildingPart. Examples are stairs, antennas, balconies or small roofs. As subclass of _CityObject, a BuildingInstallation inherits all attributes and relations, in particular an id, names, external references, and generalization relations. <xs:documentation>
</xs:documentation>
<xs:extension base="core:AbstractCityObjectType">
  <xs:sequence>
    <xs:element name="BuildingInstallation" type="BuildingInstallationType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="BuildingInstallationClassType">
  <xs:documentation>Class of a building installation. The values of this type are defined in the XML file BuildingInstallationClassType.xml, according to the dictionary concept of GML3. </xs:documentation>
</xs:complexType>
<xs:complexType name="BuildingInstallationFunctionType">
  <xs:documentation>Function of a building installation. The values of this type are defined in the XML file BuildingInstallationFunctionType.xml, according to the dictionary concept of GML3. </xs:documentation>
</xs:complexType>
<xs:complexType name="BuildingInstallationUsageType">
  <xs:documentation>Actual usage of a building installation. The values of this type are defined in the XML file BuildingInstallationUsageType.xml, according to the dictionary concept of GML3. </xs:documentation>
</xs:complexType>
<xs:complexType name="BuildingInstallationPropertyType">
  <xs:documentation>Denotes the relation of an _AbstractBuilding to its building installations. The BuildingInstallationPropertyType element must either carry a reference to a BuildingInstallation object or contain a BuildingInstallation object inline, but neither both nor none. </xs:documentation>
</xs:complexType>
<xs:complexType name="IntBuildingInstallationType">
  <xs:documentation>An IntBuildingInstallation is an interior part of a Building which has a specific function or semantical meaning. Examples are interior stairs, railings, radiators or pipes. As subclass of _CityObject, a nIntBuildingInstallation inherits all attributes and relations, in particular an id, names, external references,
and generalization relations.  

</xs:documentation>
</xs:annotation>
</xs:complexType>

<xs:complexType>
<xs:complexContent>
<xs:restriction basis="gml:AbstractCityObjectType">
前进
<xs:element name="class" type="IntBuildingInstallationClassType" minOccurs="0"/>
<xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
<xs:element ref="GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType>
<xs:complexContent>
<xs:restriction basis="gml:AbstractCityObjectType">
前进
<xs:element name="class" type="IntBuildingInstallationClassType" minOccurs="0"/>
<xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
<xs:element ref="GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType>
<xs:complexContent>
<xs:restriction basis="gml:AbstractCityObjectType">
前进
<xs:element name="class" type="IntBuildingInstallationClassType" minOccurs="0"/>
<xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
<xs:element ref="GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType>
<xs:complexContent>
<xs:restriction basis="gml:AbstractCityObjectType">
前进
<xs:element name="class" type="IntBuildingInstallationClassType" minOccurs="0"/>
<xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
<xs:element ref="GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType>
<xs:complexContent>
<xs:restriction basis="gml:AbstractCityObjectType">
前进
<xs:element name="class" type="IntBuildingInstallationClassType" minOccurs="0"/>
<xs:element name="function" type="IntBuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
<xs:element ref="GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:sequence>
    <xs:element name="_BoundarySurface" type="AbstractBoundarySurfaceType" abstract="true" substitutionGroup="core:_CityObject"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="GenericApplicationPropertyOfBoundarySurface" type="xs:anyType" abstract="true"/>

<xs:complexType name="RootSurfaceType">
  <xs:complexType>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRootSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:complexType>
</xs:complexType>

<xs:complexType name="WallSurfaceType">
  <xs:complexType>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:complexType>
</xs:complexType>

<xs:complexType name="GroundSurfaceType">
  <xs:complexType>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:complexType>
</xs:complexType>

<xs:complexType name="ClosureSurfaceType">
  <xs:complexType>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:complexType>
</xs:complexType>

<xs:complexType name="FloorSurfaceType">
  <xs:complexType>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:complexType>
</xs:complexType>
<xs:complexType name="InteriorWallSurfaceType">
    <xs:complexContent>
        <xs:restriction base="AbstractBoundarySurfaceType">
            <xs:sequence>
                <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="CeilingSurfaceType">
    <xs:complexContent>
        <xs:restriction base="AbstractBoundarySurfaceType">
            <xs:sequence>
                <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="BoundarySurfacePropertyType">
    <xs:annotation>
        <xs:documentation>
            Denotes the relation of an _AbstractBuilding to its bounding thematic surfaces (walls, roofs, ...).
            The BoundarySurfacePropertyType element must either carry a reference to a _BoundarySurface object or contain a _BoundarySurface object inline, but neither both nor none. There is no differentiation between interior surfaces bounding rooms and outer ones bounding buildings (one reason is, that ClosureSurfaces belong to both types). It has to be made sure by additional integrity constraints that, e.g. an _AbstractBuilding is not related to CeilingSurfaces or a room not to RoofSurfaces.
        </xs:documentation>
    </xs:annotation>
</xs:complexType>

<xs:complexType name="OpeningPropertyType">
    <xs:annotation>
        <xs:documentation>
            Denotes the relation of an _BoundarySurface to its openings (doors, windows). The OpeningPropertyType element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor none.
        </xs:documentation>
    </xs:annotation>
</xs:complexType>

<xs:complexType name="AbstractOpeningType" abstract="true">
    <xs:annotation>
        <xs:documentation>
            Type for openings (doors, windows) in walls. Used in LOD3 and LOD4 only. As subclass of _CityObject, an _Opening inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
        </xs:documentation>
    </xs:annotation>
</xs:complexType>

<xs:complexType name="core:AbstractCityObjectType">
    <xs:complexContent>
        <xs:restriction base="gml:MultiSurfacePropertyType">
            <xs:sequence>
                <xs:element ref="_BoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="lod4MultiSurface" abstract="true">
    <xs:restriction base="lod3MultiSurface">
        <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfOpening" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:restriction>
</xs:complexType>

<xs:complexType name="lod3MultiSurface" abstract="true">
    <xs:restriction base="gml:MultiSurfacePropertyType">
        <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:restriction>
</xs:complexType>
<xs:complexType>
    <xs:documentation>Type for windows in walls. Used in LOD3 and LOD4 only. As subclass of _CityObject, a window inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
</xs:complexType>

<xs:complexType name="WindowType">
    <xs:annotation>
      <xs:documentation>Type for windows in walls. Used in LOD3 and LOD4 only. As subclass of _CityObject, a window inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWindow" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="DoorType">
    <xs:annotation>
      <xs:documentation>Type for doors in walls. Used in LOD3 and LOD4 only. As subclass of _CityObject, a Door inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractOpeningType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfDoor" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="RoomType">
    <xs:annotation>
      <xs:documentation>A Room is a thematic object for modelling the closed parts inside a building. It has to be closed, if necessary by using closure surfaces. The geometry may be either a solid, or a MultiSurface if the boundary is not topologically clean. The room connectivity may be derived by detecting shared thematic openings or closure surfaces: two rooms are connected if both use the same opening object or the same closure surface. The thematic surfaces bounding a room are referenced by the boundedBy property. As subclass of _CityObject, a Room inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="RoomClassType" minOccurs="0"/>
          <xs:element name="function" type="RoomFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="RoomUsageType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="IoISolid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="IoIMultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="roomInstallation" type="IntBuildingInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfRoom" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>
BuildingFurnitureFunctionType is a simpleType with the following elements:

- `class`: A string that represents the class of the function. The values of this type are defined in RoomClassType.xml, according to the dictionary concept of GML3.

- `function`: A string that represents the function of the room. The values of this type are defined in RoomFunctionType.xml, according to the dictionary concept of GML3.

- `usage`: An xs:sequence of xs:string elements that represent the usage of the room. The values of this type are defined in RoomUsageType.xml, according to the dictionary concept of GML3.

- `primaryRoomPropertyType`: A xs:sequence of xs:element elements that represent the primary room property type. The values of this type are defined in the XML file BuildingFurnitureClassType.xml, according to the dictionary concept of GML3.

- `lod4Geometry`: A xs:element of xs:complexType type that represents the geometry of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4ImplicitRepresentation`: A xs:element of xs:complexType type that represents the implicit representation of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractCityObject`: A xs:element of xs:complexType type that represents the abstract city object of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuilding`: A xs:element of xs:complexType type that represents the abstract building of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingClass`: A xs:element of xs:complexType type that represents the abstract building class of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingClass`: A xs:element of xs:complexType type that represents the abstract building class of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingClass`: A xs:element of xs:complexType type that represents the abstract building class of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingClass`: A xs:element of xs:complexType type that represents the abstract building class of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingClass`: A xs:element of xs:complexType type that represents the abstract building class of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingClass`: A xs:element of xs:complexType type that represents the abstract building class of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingFunction`: A xs:element of xs:complexType type that represents the abstract building function of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.

- `lod4AbstractBuildingUsage`: A xs:element of xs:complexType type that represents the abstract building usage of the building furniture. The values of this type are defined in BuildingFurnitureType.xml, according to the dictionary concept of GML3.
<xs:restriction base="gml:AssociationType">
  <xs:sequence minOccurs="0"><xs:element ref="Room"/></xs:sequence>
</xs:restriction>
</xs:complexContent>
</xs:complexType>

<xs:complexType name="InteriorFurniturePropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of a Room to its interior furnitures (movable). The InteriorFurniturePropertyType element must either carry a reference to a BuildingFurniture object or contain a BuildingFurniture object inline, but neither both nor none.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent base="gml:AssociationType">
    <xs:sequence minOccurs="0"><xs:element ref="BuildingFurniture"/></xs:sequence>
  </xs:restriction>
</xs:complexContent>
</xs:complexType>
A.4 CityFurniture module

The CityGML CityFurniture module is defined within the XML Schema definition file cityFurniture.xsd. The target namespace http://www.opengis.net/citygml/cityfurniture/1.0 is associated with this extension module.

```xml
<xs:schema xmlns=http://www.opengis.net/citygml/cityfurniture/1.0
    xmlns:core=http://www.opengis.net/citygml/1.0
    targetNamespace=http://www.opengis.net/citygml/cityfurniture/1.0
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="/3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="cityGMLBase.xsd"/>
  <xs:complexType name="CityFurnitureType">
    <xs:annotation>
      <xs:documentation>Type describing city furnitures, like traffic lights, benches, ... As subclass of _CityObject, a
      CityFurniture inherits all attributes and relations, in particular an id, names, external references, and
generalization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:restriction base="core:AbstractCityObjectType">
        <xs:restriction base="core:GenericApplicationPropertyOfCityFurniture">
          <xs:sequence>
            <xs:element name="class" type="CoreFurnitureClassType" minOccurs="0"/>
            <xs:element name="function" type="CityFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
            <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
            <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
            <xs:element ref="_GenericApplicationPropertyOfCityFurniture" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:restriction>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

The CityFurniture module is defined within the XML Schema definition file cityFurniture.xsd. The target namespace http://www.opengis.net/citygml/cityfurniture/1.0 is associated with this extension module. The CityFurniture module is defined within the XML Schema definition file cityFurniture.xsd. The target namespace http://www.opengis.net/citygml/cityfurniture/1.0 is associated with this extension module.
A.5 CityObjectGroup module

The CityGML CityObjectGroup module is defined within the XML Schema definition file cityObjectGroup.xsd. The target namespace http://www.opengis.net/citygml/cityobjectgroup/1.0 is associated with this extension module.

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/cityobjectgroup/1.0" xmlns:core="http://www.opengis.net/citygml/1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml" targetNamespace="http://www.opengis.net/citygml/cityobjectgroup/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="cityGMLBase.xsd"/>

  <xs:element name="CityObjectGroupGroupType">
    <xs:complexType>
      <xs:documentation>
        A group may be used to aggregate arbitrary CityObjects according to some user-defined criteria. Examples for groups are the buildings in a specific region, the result of a query, or objects put together for visualization purposes. Each group has a name (inherited from AbstractGMLType), functions (e.g., building group), a class and zero or more usages. A geometry may optionally be attached to a group, if the geometry of the whole group differs from the geometry of the parts. Each member of a group may be qualified by a role name, reflecting the role each CityObject plays in the context of the group. As subclass of CityObject, a CityObjectGroup inherits all attributes and relations, in particular an id, names, external references, and generalization relations. As CityObjectGroup itself is a CityObject, it may also be contained by another group. <xs:documentation/>
      </xs:documentation>
    </xs:complexType>
  </xs:element>

  <xs:element name="CityObjectGroupMemberType">
    <xs:complexType>
      <xs:documentation>
        Denotes the relation of a CityObjectGroup to its members, which are CityObjects. The CityObjectGroupMemberType element must either carry a reference to a CityObject object or contain a CityObject object inline, but neither both nor none. <xs:documentation/>
      </xs:documentation>
    </xs:complexType>
  </xs:element>

  <xs:element name="CityObjectGroupParentType">
    <xs:complexType>
      <xs:documentation>
        Denotes the relation of a CityObjectGroup to its parent, which is a CityObject. The CityObjectGroupParentType element must either carry a reference to a CityObject object or contain a CityObject object inline, but neither both nor none. The parent association allows for modelling of a generic hierarchical grouping concept. Named aggregations of components (CityObjects) can be added to specific CityObjects considered as the parent object. The parent association links to the aggregate, while the parts are given by the group members. <xs:documentation/>
      </xs:documentation>
    </xs:complexType>
  </xs:element>
</xs:schema>
A.6 Generics module

The CityGML Generics module is defined within the XML Schema definition file generics.xsd. The target namespace http://www.opengis.net/citygml/generics/1.0 is associated with this extension module.

```xml
<xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="http://www.opengis.net/citygml/generics/1.0/generics.xsd">
  <xs:annotation>
    <xs:documentation>Nongenetic objects may be used to model complex, fine-grained thematic classes in CityGML. Otherwise, problems concerning semantic interoperability may arise. As subclass of CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:element name="GenericCityObject" type=" GenericCityObjectType"/>
  <xs:complexType name="GenericCityObjectType">
    <xs:sequence>
      <xs:element name="class" type="xs:string" minOccurs="0"/>
      <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="usage" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>      <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>      <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>      <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>      <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>      <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>      <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>      <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>      <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>      <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>      <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>      <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Generics module

Generic (user defined) city objects may be used to model features which are not covered explicitly by the CityGML schema. Generic objects must be used with care; they shall only be used if there is no appropriate thematic class available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations. Other generic attributes must be used with care; they shall only be used if there is no appropriate attribute available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Generic attributes may be used to represent attributes which are not covered explicitly by the CityGML schema. Generic attributes must be used with care; they shall only be used if there is no appropriate attribute available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Generic attributes may be used to represent attributes which are not covered explicitly by the CityGML schema. Generic attributes must be used with care; they shall only be used if there is no appropriate attribute available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Generic attributes may be used to represent attributes which are not covered explicitly by the CityGML schema. Generic attributes must be used with care; they shall only be used if there is no appropriate attribute available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Generic attributes may be used to represent attributes which are not covered explicitly by the CityGML schema. Generic attributes must be used with care; they shall only be used if there is no appropriate attribute available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
<xs:element name="stringAttribute" type="StringAttributeType" substitutionGroup="_genericAttribute"/>
<xs:complexType name="IntAttributeType">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:integer"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="intAttribute" type="IntAttributeType" substitutionGroup="_genericAttribute"/>
<xs:complexType name="DoubleAttributeType">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:double"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="doubleAttribute" type="DoubleAttributeType" substitutionGroup="_genericAttribute"/>
<xs:complexType name="DateAttributeType">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:date"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="dateAttribute" type="DateAttributeType" substitutionGroup="_genericAttribute"/>
<xs:complexType name="UriAttributeType">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="uriAttribute" type="UriAttributeType" substitutionGroup="_genericAttribute"/>
</xs:schema>
A.7 LandUse module

The CityGML LandUse module is defined within the XML Schema definition file landUse.xsd. The target namespace http://www.opengis.net/citygml/landuse/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/landuse/1.0" xmlns:core="http://www.opengis.net/citygml/1.0"
   targetNamespace="http://www.opengis.net/citygml/landuse/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.opengis.net/gml" schemaLocation="/3.1.1/base/gml.xsd"/>
   <xs:schemaType name="LandUseType">
      <xs:annotation>
         <xs:documentation>Type describing the class for Land Use in all LOD. LandUse objects describe areas of the earth’s surface dedicated to a specific land use. The geometry must consist of 3-D surfaces. As subclass of _CityObject, a LandUse inherits all attributes and relations, in particular an id, names, external references, and generalization relations. <xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="class" type="LandUseClassType" minOccurs="0"/>
            <xs:element name="function" type="LandUseFunctionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="usage" type="LandUseUsageType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
            <xs:element ref="GenericApplicationPropertyOfLandUse" minOccurs="0" maxOccurs="unbounded"/>
            <xs:sequence/>
         </xs:sequence>
      </xs:complexType>
   </xs:schemaType>
</xs:schema>
```
A.8 Relief module

The CityGML Relief module is defined within the XML Schema definition file relief.xsd. The target namespace http://www.opengis.net/citygml/relief/1.0 is associated with this extension module.
relations.  

<x:complexType>
  <x:annotation>
    <x:documentation>Type describing the raster component of a relief feature. As subclass of _CityObject, a RasterRelief inherits all attributes and relations, in particular an id, names, external references, and generalization relations.</x:documentation>
  </x:annotation>
  <x:complexContent>
    <x:extension base="AbstractReliefComponentType">
      <x:sequence>
        <x:element name="grid" type="gridPropertyType"/>
        <x:element ref="_GenericApplicationPropertyOfRasterRelief" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
      <x:extension/>
    </x:complexContent>
  </x:complexType>
</x:element>

<x:complexType name="MassPointReliefType">
  <x:annotation>
    <x:documentation>Type describing the mass point component of a relief feature. As subclass of _CityObject, a MassPointRelief inherits all attributes and relations, in particular an id, names, external references, and generalization relations.</x:documentation>
  </x:annotation>
  <x:complexContent>
    <x:extension base="AbstractReliefComponentType">
      <x:sequence>
        <x:element name="reliefPoints" type="gml:MultiPointPropertyType"/>
        <x:element ref="_GenericApplicationPropertyOfMassPointRelief" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
      <x:extension/>
    </x:complexContent>
  </x:complexType>
</x:element>

<x:complexType name="BreaklineReliefType">
  <x:annotation>
    <x:documentation>Type describing the break line Component of a relief feature. A break line relief consists of break lines or ridgeOrValleyLines. As subclass of _CityObject, a BreaklineRelief inherits all attributes and relations, in particular an id, names, external references, and generalization relations.</x:documentation>
  </x:annotation>
  <x:complexContent>
    <x:extension base="AbstractReliefComponentType">
      <x:sequence>
        <x:element name="ridgeOrValleyLines" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <x:element ref="_GenericApplicationPropertyOfBreaklineRelief" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
      <x:extension/>
    </x:complexContent>
  </x:complexType>
</x:element>
<xs:complexType name="tinPropertyType">
    <xs:annotation>
        <xs:documentation>Denotes the relation of a TINRelief to its components. The tinPropertyType element must either carry a reference to a gml:TriangulatedSurface object or contain a gml:TriangulatedSurface object inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:restriction base="gml:AssociationType">
            <xs:sequence minOccurs="0">
                <xs:element ref="gml:TriangulatedSurface"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="gridPropertyType">
    <xs:annotation>
        <xs:documentation>Denotes the relation of a RasterReliefType to its components. The gridPropertyType element must either carry a reference to a gml:RectifiedGridCoverage object or contain a gml:RectifiedGridCoverage object inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:restriction base="gml:AssociationType">
            <xs:sequence minOccurs="0">
                <xs:element ref="gml:RectifiedGridCoverage"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>

<xs:element name="Elevation" type="gml:LengthType" substitutionGroup="gml:_Object"/>
</xs:schema>
A.9 Transportation module

The CityGML Transportation module is defined within the XML Schema definition file transportation.xsd. The target namespace http://www.opengis.net/citygml/transportation/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.opengis.net/citygml/transportation/1.0"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="../1.0/gml.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/1.0" schemaLocation="cityGMLBase.xsd"/>
  <xs:complexType name="AbstractTransportationObjectType" abstract="true">
    <xs:documentation>
      Type describing the abstract superclass for transportation objects.
    </xs:documentation>
    <xs:documentation>...</xs:documentation>
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>
          Type describing the abstract superclass for transportation objects.
        </xs:documentation>
      </xs:annotation>
      <xs:complexContent>
        <xs:extension base="core:AbstractCityObjectType">
          <xs:sequence>
            <xs:element ref="_GenericApplicationPropertyOfTransportationObject" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:complexType>
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>
        Type describing transportation complexes, which are aggregated features, e.g. roads, which consist of parts (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As subclass of _CityObject, a TransportationComplex inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTransportationObjectType">
        <xs:sequence>
          <xs:element name="_TransportationObject" type="AbstractTransportationObjectType" substitutionGroup="core:CityObject"/>
          <xs:element name="_GenericApplicationPropertyOfTransportationObject" type="xs:anyType" abstract="true"/>
          <xs:complexType name="TransportationComplexType">
            <xs:documentation>
              Type describing transportation complexes, which are aggregated features, e.g. roads, which consist of parts (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As subclass of _CityObject, a TransportationComplex inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
            </xs:documentation>
            <xs:complexContent>
              <xs:extension base="AbstractTransportationObjectType">
                <xs:sequence>
                  <xs:element name="_TransportationObject" type="AbstractTransportationObjectType" substitutionGroup="core:CityObject"/>
                </xs:sequence>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="TrafficAreaType">
    <xs:annotation>
      <xs:documentation>
        Type describing the class for traffic Areas. Traffic areas are the surfaces where traffic actually takes place. As subclass of _CityObject, a TrafficArea inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="AbstractTransportationObjectType">
        <xs:sequence>
          <xs:element name="usage" type="TrafficAreaUsageType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="function" type="TrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" minOccurs="0"/>
          <xs:element name="id2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```
<xs:complexType
    name="_GenericApplicationPropertyOfAuxiliaryTrafficArea"
>
    <xs:annotation>
        <xs:documentation>
            Type describing the class for auxiliary traffic Areas. These are the surfaces where no traffic actually takes place, but which belong to a transportation object. Examples are kerbstones, road markings and grass stripes. As subclass of _CityObject, an AuxiliaryTrafficArea inherits all attributes and relations, in particular an id, names, external references, and generalization relations. 
        </xs:documentation>
    </xs:annotation>

    <xs:complexContent
    base="AbstractTransportationObjectType"
>
        <xs:restriction
            base="gml:AssociationType"
            minOccurs="0" maxOccurs="unbounded" />

        <xs:sequence
            minOccurs="0" maxOccurs="unbounded" />

        <xs:restriction
            base="TrafficAreaType"
            minOccurs="0" maxOccurs="1" />

        <xs:element
            name="Function",
            type="AuxiliaryTrafficAreaFunctionType" />

        <xs:element
            name="SurfaceMaterial",
            type="TrafficSurfaceMaterialType" />

        <xs:element
            name="TrafficSurface",
            type="gml:MultiSurfacePropertyType" />

    </xs:complexContent>
</xs:complexType>

<xs:complexType
    name="AuxiliaryTrafficAreaPropertyType"
>
    <xs:annotation>
        <xs:documentation>
            Denotes the relation of TransportationComplex to its parts, which are traffic areas. The TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object inline, but neither both nor none. 
        </xs:documentation>
    </xs:annotation>

    <xs:complexContent
    base="gml:AssociationType"
>
        <xs:restriction
            base="TrafficAreaType"
            minOccurs="0" maxOccurs="1" />

        <xs:element
            name="Function",
            type="AuxiliaryTrafficAreaFunctionType" />

        <xs:element
            name="SurfaceMaterial",
            type="TrafficSurfaceMaterialType" />

        <xs:element
            name="TrafficSurface",
            type="gml:MultiSurfacePropertyType" />

    </xs:complexContent>
</xs:complexType>

<xs:complexType
    name="AuxiliaryTrafficAreaType"
>
    <xs:annotation>
        <xs:documentation>
            As subclass of _CityObject, an AuxiliaryTrafficArea inherits all attributes and relations, in particular an id, names, external references, and generalization relations. 
        </xs:documentation>
    </xs:annotation>

    <xs:complexContent
    base="AbstractTransportationObjectType"
>
        <xs:restriction
            base="gml:AssociationType"
            minOccurs="0" maxOccurs="unbounded" />

        <xs:sequence
            minOccurs="0" maxOccurs="unbounded" />

        <xs:restriction
            base="AbstractTransportationObjectPropertyType"
            minOccurs="0" maxOccurs="1" />

        <xs:element
            name="Function",
            type="AuxiliaryTrafficAreaFunctionType" />

        <xs:element
            name="SurfaceMaterial",
            type="TrafficSurfaceMaterialType" />

        <xs:element
            name="TrafficSurface",
            type="TrafficSurfaces" />

        <xs:element
            name="TrafficSurface"
            type="gml:MultiSurfacePropertyType" />

    </xs:complexContent>
</xs:complexType>

<xs:complexContent
    base="gml:AssociationType"
>
    <xs:restriction
        base="TrafficAreaType"
        minOccurs="0" maxOccurs="1" />

    <xs:element
        name="Function",
        type="TrafficAreaFunctionType" />

    <xs:element
        name="SurfaceMaterial",
        type="TrafficSurfaceMaterialType" />

    <xs:element
        name="TrafficSurface",
        type="TrafficSurfaces" />

</xs:complexContent>
<xs:complexType>
  <xs:sequence>
    <xs:element name="Track" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RoadType">
  <xs:sequence>
    <xs:element name="Road" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RailwayType">
  <xs:sequence>
    <xs:element name="Railway" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SquareType">
  <xs:sequence>
    <xs:element name="Square" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TransportationComplexFunctionType">
  <xs:sequence>
    <xs:element name="TransportationComplexFunctionType" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TransportationComplexUsageType">
  <xs:sequence>
    <xs:element name="TransportationComplexUsageType" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TransportationComplex">
  <xs:sequence>
    <xs:element name="TransportationComplex" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TransportationComplexType">
  <xs:sequence>
    <xs:element name="TransportationComplexType" type="" xs:documentation="TransportationComplex"/>
  </xs:sequence>
</xs:complexType>
TransportationComplexUsageType.xml, according to the dictionary concept of GML3. 

```xml
<xs:simpleType name="TrafficAreaFunctionType">
  <xs:annotation>
    <xs:documentation>
      Function of a traffic area. The values of this type are defined in the XML file
      TrafficAreaFunctionType.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="AuxiliaryTrafficAreaFunctionType">
  <xs:annotation>
    <xs:documentation>
      Function of an auxiliary traffic area. The values of this type are defined in the XML file
      AuxiliaryTrafficAreaFunctionType.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="TrafficAreaUsageType">
  <xs:annotation>
    <xs:documentation>
      Usage of a traffic area. The values of this type are defined in the XML file
      TrafficAreaUsageId.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="TrafficSurfaceMaterialType">
  <xs:annotation>
    <xs:documentation>
      Type for surface materials of transportation objects. The values of this type are defined in the XML file
      TrafficSurfaceMaterialType.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

A.10 Vegetation module

The CityGML Vegetation module is defined within the XML Schema definition file vegetation.xsd. The target namespace http://www.opengis.net/citygml/vegetation/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/vegetation/1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.opengis.net/citygml/vegetation/1.0" elementFormDefault="qualif" attributeFormDefault="unqualified">

<xsd:complexType name="AbstractVegetationObjectType" abstract="true">
  <xsd:annotation>
    <xsd:documentation>��pe describing the abstract superclass for vegetation objects. A subclass is either a SolitaryVegetationObject or a PlantCover. </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="core:AbstractCityObjectType">
      <xsd:sequence>
        <xsd:element ref="_GenericApplicationPropertyOfVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="PlantCoverType">
  <xsd:annotation>
    <xsd:documentation>��pe describing Plant Covers resp. Biotopes. As subclass of _CityObject, a VegetationObject inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:restriction base="AbstractVegetationObjectType">
      <xsd:sequence>
        <xsd:element name="class" type="PlantCoverClassType" minOccurs="0"/>
        <xsd:element name="function" type="PlantCoverFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="averageHeight" type="gml:LengthType" minOccurs="0"/>
        <xsd:element name="Iod" type="gml:MultiSolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
PlantClassType

PlantClassType is a class of a SolitaryVegetationObject. The values of this type are defined in the XML file PlantClassType.xml, according to the dictionary concept of GML3.

Extension of SolitaryVegetationObject

- Class of a SolitaryVegetationObject: The values of this type are defined in the XML file PlantClassType.xml, according to the dictionary concept of GML3.
- Function of a PlantType: The values of this type are defined in the XML file PlantFunctionType.xml, according to the dictionary concept of GML3.
- Species of a Species: The values of this type are defined in the XML file SpeciesType.xml, according to the dictionary concept of GML3.
A.11 WaterBody module

The CityGML WaterBody module is defined within the XML Schema definition file waterBody.xsd. The target namespace http://www.opengis.net/citygml/waterbody/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns=http://www.opengis.net/citygml/waterbody/1.0" xmlns:xs=http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.opengis.net/citygml/waterbody/1.0" elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:import namespace=http://www.opengis.net/gml" schemaLocation="/3.1.1/base/gml.xsd"/>
<xs:complexType name="AbstractWaterObjectType" abstract="true">
  <xs:annotation>
    <xs:documentation>Abstract WaterBodyType</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="WaterBody" type="AbstractWaterObjectType" substitutionGroup=":_WaterObject"/>
</xs:complexType>
```

The CityGML WaterBody module is defined within the XML Schema definition file waterBody.xsd. The target namespace http://www.opengis.net/citygml/waterbody/1.0 is associated with this extension module.
WaterBodyFunctionType.xml, according to the dictionary concept of GML3.  

WaterBodyUsageType.xml, according to the dictionary concept of GML3.

WaterBodyFunctionType.xml, according to the dictionary concept of GML3.

WaterBoundarySurfaceType.xml, according to the dictionary concept of GML3.

WaterBodyUsageType.xml, according to the dictionary concept of GML3.

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
<xs:element name="WaterSurface" type="WaterSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<xs:element name="_GenericApplicationPropertyOfWaterSurface" type="xs:anyType" abstract="true"/>
<xs:complexType name="WaterGroundSurfaceType">
  <xs:annotation>
    <xs:documentation>Type describing the ground surface of a water body, i.e. the boundary to the digital terrain model. As subclass of _CityObject, a WaterGroundSurface inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="WaterGroundSurface" type="WaterGroundSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<xs:element name="_GenericApplicationPropertyOfWaterGroundSurface" type="xs:anyType" abstract="true"/>
<xs:complexType name="WaterClosureSurfaceType">
  <xs:annotation>
    <xs:documentation>Type describing the closure surface between water bodys. As subclass of _CityObject, a WaterClosureSurface inherits all attributes and relations, in particular an id, names, external references, and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="WaterClosureSurface" type="WaterClosureSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<xs:element name="_GenericApplicationPropertyOfWaterClosureSurface" type="xs:anyType" abstract="true"/>
A.12 TexturedSurface module [deprecated]

The CityGML TexturedSurface module is deprecated within the XML Schema definition file texturedSurface.xsd. The target namespace http://www.opengis.net/citygml/texturedsurface/1.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>

targetNamespace="http://www.opengis.net/citygml/texturedsurface/1.0"
elementFormDefault="qualified"
attributeFormDefault="unqualified"
import namespace="http://www.opengis.net/gml" schemaLocation="../3.1.1/base/gml.xsd">
<xs:complexType name="TexturedSurfaceType">
  <xs:annotation>
    <xs:appinfo>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. The concept of positioning textures on surfaces complies with the standard X3D. Because there has been no appropriate texturing concept in GML3, CityGML adds the class TexturedSurface to the geometry model of GML 3. A texture is specified as a raster image referenced by an URI, and can be an arbitrary resource, even in the internet. Textures are positioned by employing the concept of texture coordinates, i.e. each texture coordinate matches with exactly one 3D coordinate of the TexturedSurface. The use of texture coordinates allows an exact positioning and trimming of the texture on the surface geometry. Each surface may be assigned one or more appearances, each referring to one side of the surface. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="gml:OrientableSurfaceType">
      <xs:documentation>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. A property that has an Appearance as its value domain, which can either be a Material (Color,...) or a Texture. The Appearance Element can either be encapsulated in an element of this type or an XLink reference to a remote Appearance element (where remote geometry elements are located in another document or elsewhere in the same document). Either the reference or the contained element must be given, but neither both nor none. The side of the surface the Appearance refers to is given by the orientation attribute, which refers to the corresponding sign attribute of the orientable surface: `+` means the side with positive orientation, and `-` the side with negative orientation. </xs:documentation>
      <xs:sequence>
        <xs:element ref="Appearance" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
</xs:import>
<xs:complexType name="AppearancePropertyType">
  <xs:annotation>
    <xs:appinfo>Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead. A appearance that has an Appearance as its value domain, which can either be a Material (Color,...) or a Texture. The Appearance Element can either be encapsulated in an element of this type or an XLink reference to a remote Appearance element (where remote geometry elements are located in another document or elsewhere in the same document). Either the reference or the contained element must be given, but neither both nor none. The side of the surface the Appearance refers to is given by the orientation attribute, which refers to the corresponding sign attribute of the orientable surface: `+` means the side with positive orientation, and `-` the side with negative orientation. </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="orientation" type="gml:SignType" default="+"/>
  </xs:sequence>
</xs:complexType>
</xs:complexType>
</xs:schema>
```
<xs:complexType name="MaterialType">
  <xs:annotation>
    <xs:deprecation>
      <xs:appinfo>
        Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead.
      </xs:appinfo>
      <xs:documentation>Adopted from X3D standard (http://www.web3d.org/x3d/)</xs:documentation>
    </xs:deprecation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractAppearanceType">
      <xs:sequence>
        <xs:element name="textureMap" type="gml:anyURI" minOccurs="0"/>
        <xs:element name="textureCoordinates" type="gml:doubleList" minOccurs="0"/>
        <xs:element name="textureType" type="TextureTypeType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="SimpleTextureType">
  <xs:annotation>
    <xs:deprecation>
      <xs:appinfo>
        Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead.
      </xs:appinfo>
      <xs:documentation>Adopted from X3D standard (http://www.web3d.org/x3d/). ToDo: repeat</xs:documentation>
    </xs:deprecation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractAppearanceType">
      <xs:sequence>
        <xs:element name="emissiveColor" type="Color" minOccurs="0"/>
        <xs:element name="diffuseColor" type="Color" minOccurs="0"/>
        <xs:element name="specularColor" type="Color" minOccurs="0"/>
        <xs:element name="ambientIntensity" type="core:doubleBetween0and1" minOccurs="0"/>
        <xs:element name="specularExponent" type="core:integer" minOccurs="0"/>
        <xs:element name="shininess" type="core:doubleBetween0and1" minOccurs="0"/>
        <xs:element name="transparency" type="core:doubleBetween0and1" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="TextureTypeType">
  <xs:annotation>
    <xs:deprecation>
      <xs:appinfo>
        Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead.
      </xs:appinfo>
      <xs:documentation>Textures can be qualified by the attribute textureType. The textureType differentiates between textures, which are specific for a certain object and are only used for that object (specific), and prototypic textures being typical for that kind of object and are used many times for all objects of that kind (typical). A typical texture may be replaced by a specific, if available. Textures may also be classified as unknown. </xs:documentation>
    </xs:deprecation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="specific"/>
    <xs:enumeration value="typical"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="Color">
  <xs:annotation>
    <xs:deprecation>
      <xs:appinfo>
        Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead.
      </xs:appinfo>
      <xs:documentation>List of three values (red, green, blue), separated by spaces. The values must be in the range between zero and one. </xs:documentation>
    </xs:deprecation>
  </xs:annotation>
  <xs:restriction base="core:doubleBetween0and1List">
    <xs:length value="3"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>
A.13  Schematron rules on referential integrity

The CityGML Schema repository provides Schematron rules precisely describing referential integrity constraints on CityGML property elements denoting the relation between CityGML objects. If not stated otherwise, these property elements are generally restricted to either carry a reference to a remote object using the XLink concept of GML 3.1.1 (where remote objects are located in another document or elsewhere in the same document) or contain an object inline, but neither both nor none. Within this standard document, conformance requirements on referential integrity are stated within a separate clause at the end of each chapter covering a CityGML module (e.g., see clause 10.3.8 for the CityGML Building module).

Schematron is an auxiliary constraint language. The constraints apply to context nodes within a CityGML instance document (typically an XML element) which are addressed based on XPath criteria. The rules themselves are given as XPath expressions which are evaluated for each of these nodes in order to check their validity. A CityGML instance document is considered invalid if any Schematron rule is violated. Please note, that for version 1.0 of CityGML the provided Schematron rules only limit property elements to act in either by-reference or by-value mode, but neither both nor none. Neither further restrictions on property elements nor other conformance requirements are covered. In future versions of CityGML, further Schematron rules may be added.

In CityGML, the Schematron schema used to express referential integrity rules is based on the Schematron Assertion Language version 1.5 which is also employed by GML 3.1.1. The schema is shipped as a separate file within the CityGML schema package and is accessible by the name referentialIntegrity.sch. It may be used to check conformance requirements on referential integrity of CityGML property elements within a CityGML instance document according to the CityGML abstract test suite provided in Annex B. However, this requires an XML validator capable of automatically processing the Schematron rules provided by the schema. Otherwise, the Schematron code can be treated merely as a formal description of the required constraints.

The following excerpt of referentialIntegrity.sch illustrates the abstract base rule hrefOrContent. This rule states XPath expressions to ensure that target objects of property elements are either given by reference or by value. It is identically specified as the corresponding rule provided by the XSD schema file gmlBase.xsd of GML 3.1.1. Since the hrefOrContent rule is abstract, it does not apply to any specific context node within a CityGML instance document but is referenced by concrete rules within the schema to ensure consistency.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.ascc.net/xml/schematron">
  <title>Schematron validation rules checking referential integrity of CityGML association types</title>
  <ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <ns prefix="bldg" uri="http://www.opengis.net/citygml/building/1.0"/>
  ...<pattern name="Check either href or content not both">
    <rule abstract="true" id="hrefOrContent">
      <report test="@xlink:href and (*|text())">Property element may not carry both a reference to an object and contain an object.</report>
      <assert test="@xlink:href or (*|text())">Property element must either carry a reference to an object or contain an object.</assert>
    </rule>
  </pattern>
  ...
</schema>
```

For each CityGML property element being subject to referential integrity requirements, referentialIntegrity.sch contains further concrete rules based on the abstract rule hrefOrContent. For example, the following schema snippet applies the hrefOrContent rule to the property element bldg:consistsOfBuildingPart defined within the CityGML Building module (cf. chapter 10.3). The bldg:consistsOfBuildingPart element denotes the relation of a bldg:_AbstractBuilding to its building part objects which may only be given inline or by reference.

```xml
...<pattern name="hrefOrContent check on bldg:consistsOfBuildingPart">
  <rule context="bldg:consistsOfBuildingPart">
    <extends rule="hrefOrContent"/>
  </rule>
</pattern>
...```
Annex B
(normative)

Abstract test suite for CityGML instance documents

B.1 Test cases for mandatory conformance requirements

B.1.1 Valid CityGML instance document

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of each CityGML module that is part of the CityGML profile employed by the instance document. This may be any combination of CityGML extension modules in conjunction with the CityGML core module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definitions of all employed CityGML modules. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the respective XML Schema specification of the employed CityGML modules. |
| c) Reference | Annex A. |
| d) Test type | Basic Test. |

B.1.2 Valid CityGML profile

B.1.2.1 CityGML profile definition embedded inline the CityGML instance document

| a) Test purpose | Verify that a profile employed by a CityGML instance document is a valid CityGML profile in accordance with the rules and guidelines stated in chapter 7.2. For CityGML profile definitions embedded inline the CityGML instance document (referenced as first approach in chapter 7.2), verify that the CityGML instance document denotes all schema definitions and corresponding XML namespaces of CityGML modules that are used to represent the data within the instance document and, thus, are part of the employed CityGML profile. |
| b) Test method | Inspect the instance document and check that it satisfies the rules for employing CityGML profiles described in chapter 7.2 (first approach). |
| c) Reference | Annex A, chapter 7.2 (first approach). |
| d) Test type | Basic Test. |

B.1.2.2 CityGML profile definition provided by a separate XML Schema definition file

| a) Test purpose | Verify that a profile employed by a CityGML instance document is a valid CityGML profile in accordance with the rules and guidelines stated in chapter 7.2. For CityGML profile definitions provided by a separate XML Schema definition file (referenced as second approach in chapter 7.2), verify that the profile’s XML Schema definition file itself is valid and imports all schema definitions of CityGML modules that are used to represent the data within the instance document and, thus, are part of the employed CityGML profile. The target namespace of the profile’s XML Schema definition must differ from the namespaces of the imported CityGML modules and must be given as previously unused and globally unique URI. The profile’s XML Schema definition must not contain any further content. |
| b) Test method | Validate the XML Schema definition of the CityGML profile. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the respective XML Schema specification of the CityGML profile. Inspect the instance document and check that it satisfies the rules for employing CityGML profiles. |
### B.1.3 Conformance classes related to CityGML modules

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify the validity of the CityGML instance document against the conformance classes of each CityGML module that is part of the CityGML profile employed by the instance document. This may be any combination of CityGML extension modules in conjunction with the CityGML core module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Follow the test cases provided by the conformance classes for each CityGML module in annex B.2.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Annex B.2.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Basic Test.</td>
</tr>
</tbody>
</table>

### B.1.4 Spatial geometry objects

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that all spatial geometry objects within a CityGML instance document adhere to the XML Schema definition of the Geography Markup Language version 3.1.1 and to the CityGML spatial model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that spatial geometry objects are valid with respect to the XML Schema definition of GML version 3.1.1 and satisfy the rules of to the CityGML spatial model described in chapter 8.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>OGC Document No. 03-105r1, Annex A, chapter 8.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

### B.1.5 Spatial topology relations

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that all spatial topology relations between spatial geometry objects are expressed using the XML concept of XLinks provided by GML version 3.1.1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that spatial topology relations between spatial geometry objects are valid with respect to the XLinks concept introduced by GML version 3.1.1 and satisfy the rules of to the CityGML spatial model described in chapter 8.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>OGC Document No. 03-105r1, Annex A, chapter 8.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

### B.1.6 Address objects

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that all thematic objects representing address information within a CityGML instance document adhere to the XML Schema definition of the Extensible Address Language (xAL) issued by OASIS and to the rules for representing address information in CityGML.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that thematic objects representing address information are valid with respect to the XML Schema definition of the OASIS Extensible Address Language and satisfy the rules for representing address information within CityGML described in chapter 10.1.4.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>
### B.2 Conformance classes related to CityGML modules

#### B.2.1 CityGML Core module

##### B.2.1.1 Mandatory conformance requirements

<table>
<thead>
<tr>
<th></th>
<th>Test purpose</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>Test purpose</td>
<td>Verify that the CityGML instance document follows the <em>CityGML Core</em> module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents.</td>
<td></td>
</tr>
<tr>
<td>b)</td>
<td>Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the <em>CityGML Core</em> module described in chapter 10.1 and 8.2, and especially the conformance requirements stated in chapter 0 and 8.2.3. Conformance requirements on referential integrity of CityGML property elements defined within the <em>CityGML Core</em> module may be additionally validated using the constraints provided by the Schematron schema <code>referentialIntegrity.sch</code> in accordance with the rules and guidelines stated in annex A.13.</td>
<td></td>
</tr>
<tr>
<td>c)</td>
<td>Reference</td>
<td>Chapter 10.1, 0, 8.2, 8.2.3, annex A.13.</td>
<td></td>
</tr>
<tr>
<td>d)</td>
<td>Test type</td>
<td>Basic Test.</td>
<td></td>
</tr>
</tbody>
</table>

##### B.2.1.2 Valid CityGML instance document

<table>
<thead>
<tr>
<th></th>
<th>Test purpose</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>Test purpose</td>
<td>Verify the validity of the CityGML instance document against the XML Schema definition of the <em>CityGML Core</em> module. This test case is mandatory for all CityGML instance documents.</td>
<td></td>
</tr>
<tr>
<td>b)</td>
<td>Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the <em>CityGML Core</em> module in annex A.1. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the <em>CityGML Core</em> module.</td>
<td></td>
</tr>
<tr>
<td>c)</td>
<td>Reference</td>
<td>Annex A.1.</td>
<td></td>
</tr>
<tr>
<td>d)</td>
<td>Test type</td>
<td>Basic Test.</td>
<td></td>
</tr>
</tbody>
</table>

#### B.2.2 Appearance module

##### B.2.2.1 Mandatory conformance requirements

<table>
<thead>
<tr>
<th></th>
<th>Test purpose</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>Test purpose</td>
<td>Verify that the CityGML instance document follows the <em>Appearance</em> module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the <em>Appearance</em> module. Conformance requirements on referential integrity of CityGML property elements defined within the <em>Appearance</em> module may be additionally validated using the constraints provided by the Schematron schema <code>referentialIntegrity.sch</code> in accordance with the rules and guidelines stated in annex A.13.</td>
<td></td>
</tr>
<tr>
<td>b)</td>
<td>Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the <em>Appearance</em> module described in chapter 9, and especially the conformance requirements stated in chapter 9.6.</td>
<td></td>
</tr>
<tr>
<td>c)</td>
<td>Reference</td>
<td>Chapter 9, 9.6, annex A.13.</td>
<td></td>
</tr>
<tr>
<td>d)</td>
<td>Test type</td>
<td>Capability Test.</td>
<td></td>
</tr>
</tbody>
</table>

##### B.2.2.2 Valid CityGML instance document

<table>
<thead>
<tr>
<th></th>
<th>Test purpose</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>Test purpose</td>
<td>Verify the validity of the CityGML instance document against the XML Schema definition of the <em>Appearance</em> module. This test case is mandatory for all CityGML instance documents employing elements defined within the <em>Appearance</em> module.</td>
<td></td>
</tr>
</tbody>
</table>
b) Test method
Validate the CityGML XML instance document against the XML Schema definition of the Appearance module in annex A.2. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Appearance module.

c) Reference
Annex A.2.

d) Test type
Capability Test.

B.2.3 Building module

B.2.3.1 Mandatory conformance requirements

a) Test purpose
Verify that the CityGML instance document follows the Building module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Building module. Conformance requirements on referential integrity of CityGML property elements defined within the Building module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.13.

b) Test method
Inspect the instance document and check that it satisfies the rules of the Building module described in chapter 10.3, and especially the conformance requirements stated in chapter 10.3.8.

c) Reference
Chapter 10.3, 10.3.8, annex A.13.

d) Test type
Capability Test.

B.2.3.2 Valid CityGML instance document

a) Test purpose
Verify the validity of the CityGML instance document against the XML Schema definition of the Building module. This test case is mandatory for all CityGML instance documents employing elements defined within the Building module.

b) Test method
Validate the CityGML XML instance document against the XML Schema definition of the Building module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Building module.

c) Reference
Annex A.3.

d) Test type
Capability Test.

B.2.4 CityFurniture module

B.2.4.1 Mandatory conformance requirements

a) Test purpose
Verify that the CityGML instance document follows the CityFurniture module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the CityFurniture module. Conformance requirements on referential integrity of CityGML property elements defined within the CityFurniture module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.13.

b) Test method
Inspect the instance document and check that it satisfies the rules of the CityFurniture module described in chapter 10.7, and especially the conformance requirements stated in chapter 10.7.4.

c) Reference
Chapter 10.7, 10.7.4, annex A.13.
### B.2.4.2 Valid CityGML instance document

<table>
<thead>
<tr>
<th>Test type</th>
<th>Capability Test.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a)</strong> Test purpose</td>
<td>Verify the validity of the CityGML instance document against the XML Schema definition of the CityFurniture module. This test case is mandatory for all CityGML instance documents employing elements defined within the CityFurniture module.</td>
</tr>
<tr>
<td><strong>b)</strong> Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the CityFurniture module in annex A.4. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the CityFurniture module.</td>
</tr>
<tr>
<td><strong>d)</strong> Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

### B.2.5 CityObjectGroup module

#### B.2.5.1 Mandatory conformance requirements

<table>
<thead>
<tr>
<th>Test type</th>
<th>Capability Test.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a)</strong> Test purpose</td>
<td>Verify that the CityGML instance document follows the CityObjectGroup module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the CityObjectGroup module. Conformance requirements on referential integrity of CityGML property elements defined within the CityObjectGroup module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.13.</td>
</tr>
<tr>
<td><strong>b)</strong> Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the CityObjectGroup module described in chapter 10.9, and especially the conformance requirements stated in chapter 10.9.2.</td>
</tr>
<tr>
<td><strong>c)</strong> Reference</td>
<td>Chapter 10.9, 10.9.2, annex A.13.</td>
</tr>
<tr>
<td><strong>d)</strong> Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

#### B.2.5.2 Valid CityGML instance document

<table>
<thead>
<tr>
<th>Test type</th>
<th>Capability Test.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a)</strong> Test purpose</td>
<td>Verify the validity of the CityGML instance document against the XML Schema definition of the CityObjectGroup module. This test case is mandatory for all CityGML instance documents employing elements defined within the CityObjectGroup module.</td>
</tr>
<tr>
<td><strong>b)</strong> Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the CityObjectGroup module in annex A.5. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the CityObjectGroup module.</td>
</tr>
<tr>
<td><strong>c)</strong> Reference</td>
<td>Annex A.5.</td>
</tr>
<tr>
<td><strong>d)</strong> Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

### B.2.6 Generics module

#### B.2.6.1 Mandatory conformance requirements

<table>
<thead>
<tr>
<th>Test type</th>
<th>Capability Test.</th>
</tr>
</thead>
</table>
| **a)** Test purpose | Verify that the CityGML instance document follows the Generics module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Generics module. Conformance requirements on referential integrity of CityGML property
<table>
<thead>
<tr>
<th>B.2.6.2 Valid CityGML instance document</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a) Test purpose</strong></td>
</tr>
<tr>
<td><strong>b) Test method</strong></td>
</tr>
<tr>
<td><strong>c) Reference</strong></td>
</tr>
<tr>
<td><strong>d) Test type</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B.2.7 LandUse module</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B.2.7.1 Mandatory conformance requirements</strong></td>
</tr>
<tr>
<td><strong>a) Test purpose</strong></td>
</tr>
<tr>
<td><strong>b) Test method</strong></td>
</tr>
<tr>
<td><strong>c) Reference</strong></td>
</tr>
<tr>
<td><strong>d) Test type</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B.2.7.2 Valid CityGML instance document</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a) Test purpose</strong></td>
</tr>
<tr>
<td><strong>b) Test method</strong></td>
</tr>
<tr>
<td><strong>c) Reference</strong></td>
</tr>
<tr>
<td><strong>d) Test type</strong></td>
</tr>
</tbody>
</table>
B.2.8 Relief module

B.2.8.1 Mandatory conformance requirements

| a) Test purpose | Verify that the CityGML instance document follows the Relief module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Relief module. Conformance requirements on referential integrity of CityGML property elements defined within the Relief module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.13. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the Relief module described in chapter 10.2, and especially the conformance requirements stated in chapter 10.2.6. |
| c) Reference | Chapter 10.2, 10.2.6, annex A.13. |
| d) Test type | Capability Test. |

B.2.8.2 Valid CityGML instance document

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the Relief module. This test case is mandatory for all CityGML instance documents employing elements defined within the Relief module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Relief module in annex A.8. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Relief module. |
| c) Reference | Annex A.8. |
| d) Test type | Capability Test. |

B.2.9 Transportation module

B.2.9.1 Mandatory conformance requirements

| a) Test purpose | Verify that the CityGML instance document follows the Transportation module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Transportation module. Conformance requirements on referential integrity of CityGML property elements defined within the Transportation module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.13. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the Transportation module described in chapter 10.5, and especially the conformance requirements stated in chapter 10.5.5. |
| c) Reference | Chapter 10.5, 10.5.5, annex A.13. |
| d) Test type | Capability Test. |

B.2.9.2 Valid CityGML instance document

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the Transportation module. This test case is mandatory for all CityGML instance documents employing elements defined within the Transportation module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Transportation module. |
**B.2.10 Vegetation module**

**B.2.10.1 Mandatory conformance requirements**

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that the CityGML instance document follows the <em>Vegetation</em> module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the <em>Vegetation</em> module. Conformance requirements on referential integrity of CityGML property elements defined within the <em>Vegetation</em> module may be additionally validated using the constraints provided by the Schematron schema <code>referentialIntegrity.sch</code> in accordance with the rules and guidelines stated in annex A.13.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the <em>Vegetation</em> module described in chapter 10.6, and especially the conformance requirements stated in chapter 10.6.6.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 10.6, 10.6.6, annex A.13.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

**B.2.10.2 Valid CityGML instance document**

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify the validity of the CityGML instance document against the XML Schema definition of the <em>Vegetation</em> module. This test case is mandatory for all CityGML instance documents employing elements defined within the <em>Vegetation</em> module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the <em>Vegetation</em> module in annex A.10. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the <em>Vegetation</em> module.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

**B.2.11 WaterBody module**

**B.2.11.1 Mandatory conformance requirements**

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that the CityGML instance document follows the <em>WaterBody</em> module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the <em>WaterBody</em> module. Conformance requirements on referential integrity of CityGML property elements defined within the <em>WaterBody</em> module may be additionally validated using the constraints provided by the Schematron schema <code>referentialIntegrity.sch</code> in accordance with the rules and guidelines stated in annex A.13.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the <em>WaterBody</em> module described in chapter 10.4, and especially the conformance requirements stated in chapter 10.4.4.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 10.4, 10.4.4, annex A.13.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>
### B.2.11.2 Valid CityGML instance document

**a) Test purpose** Verify the validity of the CityGML instance document against the XML Schema definition of the `WaterBody` module. This test case is mandatory for all CityGML instance documents employing elements defined within the `WaterBody` module.

**b) Test method** Validate the CityGML XML instance document against the XML Schema definition of the `WaterBody` module in annex A.11. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the `WaterBody` module.

**c) Reference** Annex A.11.

**d) Test type** Capability Test.

### B.2.12 TexturedSurface module [deprecated]

#### B.2.12.1 Mandatory conformance requirements

**a) Test purpose** Verify that the CityGML instance document follows the `TexturedSurface` module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the `TexturedSurface` module. Conformance requirements on referential integrity of CityGML property elements defined within the `TexturedSurface` module may be additionally validated using the constraints provided by the Schematron schema `referentialIntegrity.sch` in accordance with the rules and guidelines stated in annex A.13.

**b) Test method** Inspect the instance document and check that it satisfies the rules of the `TexturedSurface` module described in chapter 9.7, and especially the conformance requirements stated in chapter 9.7.2.


**d) Test type** Capability Test.

#### B.2.12.2 Valid CityGML instance documents

**a) Test purpose** Verify the validity of the CityGML instance document against the XML Schema definition of the `TexturedSurface` module. This test case is mandatory for all CityGML instance documents employing elements defined within the `TexturedSurface` module.

**b) Test method** Validate the CityGML XML instance document against the XML Schema definition of the `TexturedSurface` module in annex A.12. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the `TexturedSurface` module.

**c) Reference** Annex A.12.

**d) Test type** Capability Test.
Annex C
(informative)

**External code lists**

In this annex the corresponding values of the external code lists of CityGML (cf. chapter 6.6) are given. The following values are a proposal of the SIG 3D and may be extended or replaced by other communities to fit their needs. The external code list for roof types is given in XML format as example below, while the others are depicted in the following subclauses in tabular form for space reasons. The subclauses are arranged according to thematic fields and comprise the relevant external code lists of these fields.

**External code list for roof types, which are defined in the file RoofTypeType.xml:**

```xml
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/gml ../3.1.1/profiles/simpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
    gml:id="RoofTypeType">
    <name>RoofTypeType</name>
    <dictionaryEntry>
        <gml:Definition gml:id="id357">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1000</gml:name>
            <gml:name>flat roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
        <gml:Definition gml:id="id358">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1010</gml:name>
            <gml:name>monopitch roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
        <gml:Definition gml:id="id359">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1020</gml:name>
            <gml:name>skip pent roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
        <gml:Definition gml:id="id360">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1030</gml:name>
            <gml:name>gabled roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
        <gml:Definition gml:id="id361">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1040</gml:name>
            <gml:name>hipped roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
        <gml:Definition gml:id="id362">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1050</gml:name>
            <gml:name>half-hipped roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
    <dictionaryEntry>
        <gml:Definition gml:id="id363">
            <gml:description/>
            <gml:name codeSpace="urn:d_nrw_sig3d">1060</gml:name>
            <gml:name>mansard roof</gml:name>
        </gml:Definition>
    </dictionaryEntry>
</gml:Dictionary>
```
<dictionaryEntry>
  <gml:Definition gml:id="id364">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">pavilion roof</gml:name>
  </gml:Definition>
</dictionaryEntry>

<dictionaryEntry>
  <gml:Definition gml:id="id365">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">cone roof</gml:name>
  </gml:Definition>
</dictionaryEntry>

<dictionaryEntry>
  <gml:Definition gml:id="id366">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">copula roof</gml:name>
  </gml:Definition>
</dictionaryEntry>

<dictionaryEntry>
  <gml:Definition gml:id="id367">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">shed roof</gml:name>
  </gml:Definition>
</dictionaryEntry>

<dictionaryEntry>
  <gml:Definition gml:id="id368">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">arch roof</gml:name>
  </gml:Definition>
</dictionaryEntry>

<dictionaryEntry>
  <gml:Definition gml:id="id369">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">pyramidal broach roof</gml:name>
  </gml:Definition>
</dictionaryEntry>

<dictionaryEntry>
  <gml:Definition gml:id="id370">
    <gml:description></gml:description>
    <gml:name codeSpace="urn:d_nrw_sig3d">combination of roof forms</gml:name>
  </gml:Definition>
</dictionaryEntry>
### C.1 Building

<table>
<thead>
<tr>
<th>BuildingClassType</th>
<th>Code list derived from German authoritative standards ALKIS/ATKIS (<a href="http://www.adv-online.de">www.adv-online.de</a>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>habitation</td>
</tr>
<tr>
<td>1010</td>
<td>sanitation</td>
</tr>
<tr>
<td>1020</td>
<td>administration</td>
</tr>
<tr>
<td>1030</td>
<td>business, trade</td>
</tr>
<tr>
<td>1040</td>
<td>catering</td>
</tr>
<tr>
<td>1050</td>
<td>recreation</td>
</tr>
<tr>
<td>1060</td>
<td>sport</td>
</tr>
<tr>
<td>1070</td>
<td>culture</td>
</tr>
<tr>
<td>1080</td>
<td>church institution</td>
</tr>
<tr>
<td>1090</td>
<td>agriculture, forestry</td>
</tr>
<tr>
<td>1100</td>
<td>schools, education, research</td>
</tr>
<tr>
<td>1110</td>
<td>maintenance and waste management</td>
</tr>
<tr>
<td>1120</td>
<td>healthcare</td>
</tr>
<tr>
<td>1130</td>
<td>communicating</td>
</tr>
<tr>
<td>1140</td>
<td>security</td>
</tr>
<tr>
<td>1150</td>
<td>storage</td>
</tr>
<tr>
<td>1160</td>
<td>industry</td>
</tr>
<tr>
<td>1170</td>
<td>traffic</td>
</tr>
<tr>
<td>1180</td>
<td>function</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BuildingFunctionType</th>
<th>Code list derived from German authoritative standards ALKIS/ATKIS (<a href="http://www.adv-online.de">www.adv-online.de</a>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>residential building</td>
</tr>
<tr>
<td>1010</td>
<td>tenement</td>
</tr>
<tr>
<td>1020</td>
<td>hostel</td>
</tr>
<tr>
<td>1030</td>
<td>residential- and administration building</td>
</tr>
<tr>
<td>1040</td>
<td>residential- and office building</td>
</tr>
<tr>
<td>1050</td>
<td>residential- and business building</td>
</tr>
<tr>
<td>1060</td>
<td>residential- and plant building</td>
</tr>
<tr>
<td>1070</td>
<td>agrarian- and forestry building</td>
</tr>
<tr>
<td>1080</td>
<td>residential- and commercial building</td>
</tr>
<tr>
<td>1090</td>
<td>forester's lodge</td>
</tr>
<tr>
<td>1100</td>
<td>holiday house</td>
</tr>
<tr>
<td>1110</td>
<td>summer house</td>
</tr>
<tr>
<td>1120</td>
<td>office building</td>
</tr>
<tr>
<td>1130</td>
<td>credit institution</td>
</tr>
<tr>
<td>1140</td>
<td>insurance</td>
</tr>
<tr>
<td>1150</td>
<td>business building</td>
</tr>
<tr>
<td>1160</td>
<td>department store</td>
</tr>
<tr>
<td>1170</td>
<td>shopping centre</td>
</tr>
<tr>
<td>1180</td>
<td>kiosk</td>
</tr>
<tr>
<td>1190</td>
<td>pharmacy</td>
</tr>
<tr>
<td>1200</td>
<td>pavilion</td>
</tr>
<tr>
<td>1210</td>
<td>hotel</td>
</tr>
<tr>
<td>1220</td>
<td>youth hostel</td>
</tr>
<tr>
<td>1230</td>
<td>campsite building</td>
</tr>
<tr>
<td>1240</td>
<td>restaurant</td>
</tr>
<tr>
<td>1250</td>
<td>cantine</td>
</tr>
<tr>
<td>1260</td>
<td>recreational site</td>
</tr>
<tr>
<td>1270</td>
<td>function room</td>
</tr>
<tr>
<td>1280</td>
<td>cinema</td>
</tr>
<tr>
<td>1290</td>
<td>bowling alley</td>
</tr>
<tr>
<td>1300</td>
<td>casino</td>
</tr>
<tr>
<td>1310</td>
<td>industrial building</td>
</tr>
<tr>
<td>1320</td>
<td>factory</td>
</tr>
<tr>
<td>1330</td>
<td>workshop</td>
</tr>
<tr>
<td>1840</td>
<td>rubbish bunker</td>
</tr>
<tr>
<td>1850</td>
<td>building for rubbish incineration</td>
</tr>
<tr>
<td>1860</td>
<td>building for rubbish disposal</td>
</tr>
<tr>
<td>1870</td>
<td>building for agrarian and forestry</td>
</tr>
<tr>
<td>1880</td>
<td>barn</td>
</tr>
<tr>
<td>1890</td>
<td>stall</td>
</tr>
<tr>
<td>1900</td>
<td>equestrian hall</td>
</tr>
<tr>
<td>1910</td>
<td>alpine cabin</td>
</tr>
<tr>
<td>1920</td>
<td>hunting lodge</td>
</tr>
<tr>
<td>1930</td>
<td>arboretum</td>
</tr>
<tr>
<td>1940</td>
<td>glass house</td>
</tr>
<tr>
<td>1950</td>
<td>moveable glass house</td>
</tr>
<tr>
<td>1960</td>
<td>public building</td>
</tr>
<tr>
<td>1970</td>
<td>administration building</td>
</tr>
<tr>
<td>1980</td>
<td>parliament</td>
</tr>
<tr>
<td>1990</td>
<td>guildhall</td>
</tr>
<tr>
<td>2000</td>
<td>post office</td>
</tr>
<tr>
<td>2010</td>
<td>customs office</td>
</tr>
<tr>
<td>2020</td>
<td>court</td>
</tr>
<tr>
<td>2030</td>
<td>embassy or consulate</td>
</tr>
<tr>
<td>2040</td>
<td>district administration</td>
</tr>
<tr>
<td>2050</td>
<td>district government</td>
</tr>
<tr>
<td>2060</td>
<td>tax office</td>
</tr>
<tr>
<td>2070</td>
<td>building for education and research</td>
</tr>
<tr>
<td>2080</td>
<td>comprehensive school</td>
</tr>
<tr>
<td>2090</td>
<td>vocational school</td>
</tr>
<tr>
<td>2100</td>
<td>college or university</td>
</tr>
<tr>
<td>2110</td>
<td>research establishment</td>
</tr>
<tr>
<td>2120</td>
<td>building for cultural purposes</td>
</tr>
<tr>
<td>2130</td>
<td>castle</td>
</tr>
<tr>
<td>2140</td>
<td>theatre or opera</td>
</tr>
<tr>
<td>2150</td>
<td>concert building</td>
</tr>
<tr>
<td>2160</td>
<td>museum</td>
</tr>
<tr>
<td>2170</td>
<td>broadcasting building</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>1340</td>
<td>petrol / gas station</td>
</tr>
<tr>
<td>1350</td>
<td>washing plant</td>
</tr>
<tr>
<td>1360</td>
<td>cold store</td>
</tr>
<tr>
<td>1370</td>
<td>depot</td>
</tr>
<tr>
<td>1380</td>
<td>building for research purposes</td>
</tr>
<tr>
<td>1390</td>
<td>quarry</td>
</tr>
<tr>
<td>1400</td>
<td>salt works</td>
</tr>
<tr>
<td>1410</td>
<td>miscellaneous industrial building</td>
</tr>
<tr>
<td>1420</td>
<td>mill</td>
</tr>
<tr>
<td>1430</td>
<td>windmill</td>
</tr>
<tr>
<td>1440</td>
<td>water mill</td>
</tr>
<tr>
<td>1450</td>
<td>bucket elevator</td>
</tr>
<tr>
<td>1460</td>
<td>weather station</td>
</tr>
<tr>
<td>1470</td>
<td>traffic assets office</td>
</tr>
<tr>
<td>1480</td>
<td>street maintenance</td>
</tr>
<tr>
<td>1490</td>
<td>waiting hall</td>
</tr>
<tr>
<td>1500</td>
<td>signal control box</td>
</tr>
<tr>
<td>1510</td>
<td>engine shed</td>
</tr>
<tr>
<td>1520</td>
<td>signal box or stop signal</td>
</tr>
<tr>
<td>1530</td>
<td>plant building for air traffic</td>
</tr>
<tr>
<td>1540</td>
<td>hangar</td>
</tr>
<tr>
<td>1550</td>
<td>plant building for shipping</td>
</tr>
<tr>
<td>1560</td>
<td>shipyard</td>
</tr>
<tr>
<td>1570</td>
<td>dock</td>
</tr>
<tr>
<td>1580</td>
<td>plant building for canal lock</td>
</tr>
<tr>
<td>1590</td>
<td>boathouse</td>
</tr>
<tr>
<td>1600</td>
<td>plant building for cablecar</td>
</tr>
<tr>
<td>1610</td>
<td>multi-storey car park</td>
</tr>
<tr>
<td>1620</td>
<td>parking level</td>
</tr>
<tr>
<td>1630</td>
<td>garage</td>
</tr>
<tr>
<td>1640</td>
<td>vehicle hall</td>
</tr>
<tr>
<td>1650</td>
<td>underground garage</td>
</tr>
<tr>
<td>1660</td>
<td>building for supply</td>
</tr>
<tr>
<td>1670</td>
<td>waterworks</td>
</tr>
<tr>
<td>1680</td>
<td>pump station</td>
</tr>
<tr>
<td>1690</td>
<td>water basin</td>
</tr>
<tr>
<td>1700</td>
<td>electric power station</td>
</tr>
<tr>
<td>1710</td>
<td>transformer station</td>
</tr>
<tr>
<td>1720</td>
<td>converter</td>
</tr>
<tr>
<td>1730</td>
<td>reactor</td>
</tr>
<tr>
<td>1740</td>
<td>turbine house</td>
</tr>
<tr>
<td>1750</td>
<td>boiler house</td>
</tr>
<tr>
<td>1760</td>
<td>building for telecommunications</td>
</tr>
<tr>
<td>1770</td>
<td>gas works</td>
</tr>
<tr>
<td>1780</td>
<td>heat plant</td>
</tr>
<tr>
<td>1790</td>
<td>pumping station</td>
</tr>
<tr>
<td>1800</td>
<td>building for disposal</td>
</tr>
<tr>
<td>1810</td>
<td>building for effluent disposal</td>
</tr>
<tr>
<td>1820</td>
<td>building for filter plant</td>
</tr>
<tr>
<td>1830</td>
<td>toilet</td>
</tr>
</tbody>
</table>
### BuildingUsageType
Code list identically specified as BuildingFunctionType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>habituation</td>
</tr>
<tr>
<td>1010</td>
<td>sanitation</td>
</tr>
<tr>
<td>1020</td>
<td>administration</td>
</tr>
<tr>
<td>1030</td>
<td>business, trade</td>
</tr>
<tr>
<td>1040</td>
<td>catering</td>
</tr>
<tr>
<td>1050</td>
<td>recreation</td>
</tr>
<tr>
<td>1060</td>
<td>sport</td>
</tr>
<tr>
<td>1070</td>
<td>culture</td>
</tr>
<tr>
<td>1080</td>
<td>church institution</td>
</tr>
<tr>
<td>1090</td>
<td>agriculture, forestry</td>
</tr>
<tr>
<td>1100</td>
<td>schools, education, research</td>
</tr>
<tr>
<td>1110</td>
<td>maintenance, waste management</td>
</tr>
<tr>
<td>1120</td>
<td>healthcare</td>
</tr>
<tr>
<td>1130</td>
<td>communicating</td>
</tr>
<tr>
<td>1140</td>
<td>security</td>
</tr>
<tr>
<td>1150</td>
<td>storage</td>
</tr>
<tr>
<td>1160</td>
<td>industry</td>
</tr>
<tr>
<td>1170</td>
<td>traffic</td>
</tr>
<tr>
<td>1180</td>
<td>function</td>
</tr>
</tbody>
</table>

### BuildingFurnitureClassType
Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>cupboard</td>
</tr>
<tr>
<td>1010</td>
<td>wardrobe</td>
</tr>
<tr>
<td>1020</td>
<td>cabinet</td>
</tr>
<tr>
<td>1030</td>
<td>sideboard</td>
</tr>
<tr>
<td>1040</td>
<td>locker</td>
</tr>
<tr>
<td>1050</td>
<td>tool cabinet</td>
</tr>
<tr>
<td>1100</td>
<td>shelf</td>
</tr>
<tr>
<td>1110</td>
<td>rack</td>
</tr>
<tr>
<td>1120</td>
<td>coat stand</td>
</tr>
<tr>
<td>1200</td>
<td>table</td>
</tr>
<tr>
<td>1210</td>
<td>dining table</td>
</tr>
<tr>
<td>1220</td>
<td>coffee table</td>
</tr>
<tr>
<td>1230</td>
<td>desk</td>
</tr>
<tr>
<td>1240</td>
<td>bedside cabinet</td>
</tr>
<tr>
<td>1250</td>
<td>baby changing table</td>
</tr>
<tr>
<td>1260</td>
<td>bar</td>
</tr>
<tr>
<td>1270</td>
<td>pool table</td>
</tr>
<tr>
<td>1280</td>
<td>snooker table</td>
</tr>
<tr>
<td>1290</td>
<td>roulette table</td>
</tr>
<tr>
<td>1270</td>
<td>work bench</td>
</tr>
<tr>
<td>1300</td>
<td>chair</td>
</tr>
<tr>
<td>1310</td>
<td>bench</td>
</tr>
<tr>
<td>1320</td>
<td>office chair</td>
</tr>
<tr>
<td>1330</td>
<td>sofa</td>
</tr>
<tr>
<td>1340</td>
<td>rocking chair</td>
</tr>
<tr>
<td>1350</td>
<td>bar stool</td>
</tr>
<tr>
<td>1360</td>
<td>armchair</td>
</tr>
<tr>
<td>1400</td>
<td>bed</td>
</tr>
<tr>
<td>1410</td>
<td>crib</td>
</tr>
<tr>
<td>1420</td>
<td>bunk bed</td>
</tr>
<tr>
<td>1430</td>
<td>cradle</td>
</tr>
<tr>
<td>1440</td>
<td>cot</td>
</tr>
<tr>
<td>1450</td>
<td>stretcher</td>
</tr>
<tr>
<td>2000</td>
<td>sink, hand-basin</td>
</tr>
<tr>
<td>2010</td>
<td>water tap</td>
</tr>
<tr>
<td>2020</td>
<td>toilet bowl</td>
</tr>
<tr>
<td>2030</td>
<td>bathtub</td>
</tr>
<tr>
<td>2040</td>
<td>shower</td>
</tr>
<tr>
<td>2050</td>
<td>bidet</td>
</tr>
<tr>
<td>2100</td>
<td>animal park</td>
</tr>
<tr>
<td>2110</td>
<td>aquarium</td>
</tr>
<tr>
<td>2120</td>
<td>cage</td>
</tr>
<tr>
<td>2130</td>
<td>birdcage</td>
</tr>
<tr>
<td>2200</td>
<td>religious equipment</td>
</tr>
<tr>
<td>2220</td>
<td>shop fittings</td>
</tr>
<tr>
<td>2300</td>
<td>sales counter</td>
</tr>
<tr>
<td>2310</td>
<td>glass cabinet</td>
</tr>
<tr>
<td>2320</td>
<td>changing cubicle</td>
</tr>
<tr>
<td>2330</td>
<td>refrigerated counter</td>
</tr>
<tr>
<td>2340</td>
<td>cash desk or till or counter</td>
</tr>
<tr>
<td>2350</td>
<td>box-office</td>
</tr>
<tr>
<td>2360</td>
<td>machines</td>
</tr>
<tr>
<td>2400</td>
<td>ticket machine</td>
</tr>
<tr>
<td>2410</td>
<td>cigarette machine</td>
</tr>
<tr>
<td>2420</td>
<td>cash machine or ATM</td>
</tr>
<tr>
<td>2430</td>
<td>vending machine</td>
</tr>
<tr>
<td>2440</td>
<td>gambling machine</td>
</tr>
<tr>
<td>2450</td>
<td>technical furniture</td>
</tr>
<tr>
<td>2500</td>
<td>heating installation</td>
</tr>
<tr>
<td>2510</td>
<td>tank</td>
</tr>
<tr>
<td>2520</td>
<td>oil tank</td>
</tr>
<tr>
<td>2521</td>
<td>water tank</td>
</tr>
<tr>
<td>2522</td>
<td>gas tank</td>
</tr>
<tr>
<td>2523</td>
<td>fuel tank</td>
</tr>
<tr>
<td>2524</td>
<td>milk tank</td>
</tr>
<tr>
<td>2525</td>
<td>steel tank</td>
</tr>
<tr>
<td>BuildingFurnitureUsageType</td>
<td>Code list identically specified as BuildingFurnitureFunctionType</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>BuildingInstallationClassType</td>
<td>Code list derived from German authoritative standards ALKIS/ATKIS (<a href="http://www.adv-online.de">www.adv-online.de</a>)</td>
</tr>
<tr>
<td>BuildingInstallationFunctionType</td>
<td>Code list derived from German authoritative standards ALKIS/ATKIS (<a href="http://www.adv-online.de">www.adv-online.de</a>)</td>
</tr>
</tbody>
</table>
### BuildingInstallationUsageType
Code list identically specified as BuildingInstallationFunctionType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Heating, Ventilation, Climate</td>
</tr>
<tr>
<td>2000</td>
<td>Safety</td>
</tr>
<tr>
<td>3000</td>
<td>Illumination</td>
</tr>
<tr>
<td>4000</td>
<td>Communication</td>
</tr>
<tr>
<td>5000</td>
<td>Supply and Disposal</td>
</tr>
<tr>
<td>6000</td>
<td>Statics</td>
</tr>
<tr>
<td>7000</td>
<td>Entertainment</td>
</tr>
<tr>
<td>8000</td>
<td>Miscellaneous</td>
</tr>
<tr>
<td>9999</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

### IntBuildingInstallationClassType
Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Heating, Ventilation, Climate</td>
</tr>
<tr>
<td>2000</td>
<td>Safety</td>
</tr>
<tr>
<td>3000</td>
<td>Illumination</td>
</tr>
<tr>
<td>4000</td>
<td>Communication</td>
</tr>
<tr>
<td>5000</td>
<td>Supply and Disposal</td>
</tr>
<tr>
<td>6000</td>
<td>Statics</td>
</tr>
<tr>
<td>7000</td>
<td>Entertainment</td>
</tr>
<tr>
<td>8000</td>
<td>Miscellaneous</td>
</tr>
<tr>
<td>9999</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

### IntBuildingInstallationFunctionType
Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>Radiator</td>
</tr>
<tr>
<td>1020</td>
<td>Oven</td>
</tr>
<tr>
<td>1030</td>
<td>Fireside</td>
</tr>
<tr>
<td>1040</td>
<td>Ventilator</td>
</tr>
<tr>
<td>1050</td>
<td>Air Conditioning</td>
</tr>
<tr>
<td>1100</td>
<td>Pipe</td>
</tr>
<tr>
<td>1110</td>
<td>Lamp</td>
</tr>
<tr>
<td>1120</td>
<td>Air Conditioning</td>
</tr>
<tr>
<td>1130</td>
<td>Lamp</td>
</tr>
</tbody>
</table>

### RoofTypeType
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>flat roof</td>
</tr>
<tr>
<td>1010</td>
<td>monopitch roof</td>
</tr>
<tr>
<td>1020</td>
<td>skip pent roof</td>
</tr>
<tr>
<td>1030</td>
<td>gabled roof</td>
</tr>
<tr>
<td>1040</td>
<td>hipped roof</td>
</tr>
<tr>
<td>1050</td>
<td>half-hipped roof</td>
</tr>
<tr>
<td>1060</td>
<td>mansard roof</td>
</tr>
<tr>
<td>1070</td>
<td>pavilion roof</td>
</tr>
<tr>
<td>1080</td>
<td>cone roof</td>
</tr>
<tr>
<td>1090</td>
<td>copula roof</td>
</tr>
<tr>
<td>1100</td>
<td>shed roof</td>
</tr>
<tr>
<td>1110</td>
<td>arch roof</td>
</tr>
<tr>
<td>1120</td>
<td>pyramidal broach roof</td>
</tr>
<tr>
<td>1130</td>
<td>combination of roof forms</td>
</tr>
</tbody>
</table>

### RoomClassType
Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>habitation</td>
</tr>
<tr>
<td>1010</td>
<td>administration</td>
</tr>
<tr>
<td>1020</td>
<td>business, trade</td>
</tr>
<tr>
<td>1030</td>
<td>catering</td>
</tr>
<tr>
<td>1040</td>
<td>recreation</td>
</tr>
<tr>
<td>1050</td>
<td>church institution</td>
</tr>
<tr>
<td>1060</td>
<td>agriculture, forestry</td>
</tr>
<tr>
<td>1070</td>
<td>schools, education, research</td>
</tr>
<tr>
<td>1080</td>
<td>accommodation, waste management</td>
</tr>
<tr>
<td>1090</td>
<td>healthcare</td>
</tr>
<tr>
<td>1100</td>
<td>communicating</td>
</tr>
<tr>
<td>1110</td>
<td>security</td>
</tr>
<tr>
<td>1120</td>
<td>store</td>
</tr>
<tr>
<td>1130</td>
<td>industry</td>
</tr>
<tr>
<td>1140</td>
<td>traffic</td>
</tr>
<tr>
<td>1150</td>
<td>function</td>
</tr>
</tbody>
</table>

### RoomFunctionType
Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>living room</td>
</tr>
<tr>
<td>1010</td>
<td>bedroom</td>
</tr>
<tr>
<td>1080</td>
<td>showers</td>
</tr>
<tr>
<td>2170</td>
<td>showers</td>
</tr>
<tr>
<td>2200</td>
<td>tribeun</td>
</tr>
<tr>
<td>Code</td>
<td>Room Type</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>1020</td>
<td>kitchen</td>
</tr>
<tr>
<td>1030</td>
<td>hall</td>
</tr>
<tr>
<td>1040</td>
<td>bath, washroom</td>
</tr>
<tr>
<td>1050</td>
<td>toilet</td>
</tr>
<tr>
<td>1060</td>
<td>stairs</td>
</tr>
<tr>
<td>1070</td>
<td>home office</td>
</tr>
<tr>
<td>1080</td>
<td>utility room</td>
</tr>
<tr>
<td>1090</td>
<td>dining room</td>
</tr>
<tr>
<td>1100</td>
<td>common room</td>
</tr>
<tr>
<td>1110</td>
<td>party room</td>
</tr>
<tr>
<td>1120</td>
<td>nursery</td>
</tr>
<tr>
<td>1130</td>
<td>store room</td>
</tr>
<tr>
<td>1140</td>
<td>canteen, common kitchen</td>
</tr>
<tr>
<td>1150</td>
<td>storeroom</td>
</tr>
<tr>
<td>1160</td>
<td>balcony, gallery</td>
</tr>
<tr>
<td>1170</td>
<td>terrace</td>
</tr>
<tr>
<td>1180</td>
<td>drying room</td>
</tr>
<tr>
<td>1190</td>
<td>heatingroom</td>
</tr>
<tr>
<td>1200</td>
<td>fuel depot</td>
</tr>
<tr>
<td>1210</td>
<td>hobby room</td>
</tr>
<tr>
<td>1220</td>
<td>stable, hovel</td>
</tr>
<tr>
<td>1300</td>
<td>cash office</td>
</tr>
<tr>
<td>1310</td>
<td>ticket office</td>
</tr>
<tr>
<td>1320</td>
<td>conference room</td>
</tr>
<tr>
<td>1330</td>
<td>reception</td>
</tr>
<tr>
<td>1340</td>
<td>sales room</td>
</tr>
<tr>
<td>1350</td>
<td>store room</td>
</tr>
<tr>
<td>1360</td>
<td>delivery</td>
</tr>
<tr>
<td>1370</td>
<td>lounge, common room</td>
</tr>
<tr>
<td>1380</td>
<td>escalator</td>
</tr>
<tr>
<td>1390</td>
<td>guest toilet</td>
</tr>
<tr>
<td>1400</td>
<td>strong room</td>
</tr>
<tr>
<td>1500</td>
<td>office</td>
</tr>
<tr>
<td>1510</td>
<td>entrance hall</td>
</tr>
<tr>
<td>1520</td>
<td>elevator</td>
</tr>
<tr>
<td>1530</td>
<td>canteen</td>
</tr>
<tr>
<td>1540</td>
<td>tea kitchen / Coffee kitchen</td>
</tr>
<tr>
<td>1550</td>
<td>archive</td>
</tr>
<tr>
<td>1560</td>
<td>citizen office</td>
</tr>
<tr>
<td>1570</td>
<td>conference hall</td>
</tr>
<tr>
<td>1580</td>
<td>copier room / blueprint room</td>
</tr>
<tr>
<td>1590</td>
<td>information</td>
</tr>
<tr>
<td>1600</td>
<td>computer room</td>
</tr>
<tr>
<td>1610</td>
<td>printer / plotter room</td>
</tr>
<tr>
<td>1700</td>
<td>reception</td>
</tr>
<tr>
<td>1710</td>
<td>guest room</td>
</tr>
<tr>
<td>1720</td>
<td>bar</td>
</tr>
<tr>
<td>1730</td>
<td>breakfast room</td>
</tr>
<tr>
<td>1740</td>
<td>dining room</td>
</tr>
<tr>
<td>1750</td>
<td>celebration room</td>
</tr>
<tr>
<td>1760</td>
<td>pub</td>
</tr>
<tr>
<td>1770</td>
<td>beer garden</td>
</tr>
<tr>
<td>1780</td>
<td>restaurant</td>
</tr>
<tr>
<td>RoomFunctionType</td>
<td>RoomUsageType</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>cool store</td>
<td>1790</td>
</tr>
<tr>
<td>bowling alley, shoot alley</td>
<td>1800</td>
</tr>
<tr>
<td>lounge</td>
<td>1810</td>
</tr>
<tr>
<td>canteen kitchen</td>
<td>1820</td>
</tr>
<tr>
<td>stage</td>
<td>1900</td>
</tr>
<tr>
<td>auditorium</td>
<td>1910</td>
</tr>
<tr>
<td>VIP box</td>
<td>1920</td>
</tr>
<tr>
<td>projection room</td>
<td>1930</td>
</tr>
<tr>
<td>dressing room</td>
<td>1940</td>
</tr>
<tr>
<td>cabin</td>
<td>1950</td>
</tr>
<tr>
<td>showroom</td>
<td>1960</td>
</tr>
<tr>
<td>equipment or props</td>
<td>1970</td>
</tr>
<tr>
<td>make-up room</td>
<td>1980</td>
</tr>
<tr>
<td>recording studio</td>
<td>1990</td>
</tr>
<tr>
<td>sound studio</td>
<td>2000</td>
</tr>
<tr>
<td>music archive</td>
<td>2010</td>
</tr>
<tr>
<td>administration</td>
<td>2020</td>
</tr>
<tr>
<td>ticket office</td>
<td>2030</td>
</tr>
<tr>
<td>library</td>
<td>2040</td>
</tr>
<tr>
<td>media room</td>
<td>2050</td>
</tr>
<tr>
<td>dressing room</td>
<td>2060</td>
</tr>
<tr>
<td>sport room</td>
<td>2070</td>
</tr>
<tr>
<td>equipment room</td>
<td>2080</td>
</tr>
<tr>
<td>platform</td>
<td>2090</td>
</tr>
<tr>
<td>swimming-pool</td>
<td>2100</td>
</tr>
<tr>
<td>slide</td>
<td>2110</td>
</tr>
<tr>
<td>relaxation room</td>
<td>2120</td>
</tr>
<tr>
<td>sauna</td>
<td>2130</td>
</tr>
<tr>
<td>fitness room</td>
<td>2140</td>
</tr>
<tr>
<td>solarium</td>
<td>2150</td>
</tr>
<tr>
<td>catering</td>
<td>2160</td>
</tr>
<tr>
<td>production building</td>
<td>3010</td>
</tr>
<tr>
<td>factory building</td>
<td>3020</td>
</tr>
<tr>
<td>workshop</td>
<td>3030</td>
</tr>
<tr>
<td>storage depot</td>
<td>3040</td>
</tr>
<tr>
<td>cold storage</td>
<td>3050</td>
</tr>
<tr>
<td>store</td>
<td>3060</td>
</tr>
<tr>
<td>station concourse</td>
<td>3100</td>
</tr>
<tr>
<td>track</td>
<td>3110</td>
</tr>
<tr>
<td>ticket office</td>
<td>3120</td>
</tr>
<tr>
<td>waiting hall</td>
<td>3130</td>
</tr>
<tr>
<td>engine shed</td>
<td>3140</td>
</tr>
<tr>
<td>signal box</td>
<td>3150</td>
</tr>
<tr>
<td>departure terminal</td>
<td>3160</td>
</tr>
<tr>
<td>check-out counter</td>
<td>3170</td>
</tr>
<tr>
<td>check-in counter</td>
<td>3180</td>
</tr>
<tr>
<td>check</td>
<td>3190</td>
</tr>
<tr>
<td>baggage carousel</td>
<td>3200</td>
</tr>
<tr>
<td>security check</td>
<td>3210</td>
</tr>
<tr>
<td>classroom</td>
<td>3300</td>
</tr>
<tr>
<td>staff room</td>
<td>3310</td>
</tr>
<tr>
<td>break or recess hall</td>
<td>3320</td>
</tr>
<tr>
<td>laboratory</td>
<td>3330</td>
</tr>
<tr>
<td>utility room</td>
<td>3340</td>
</tr>
<tr>
<td>media room</td>
<td>3350</td>
</tr>
<tr>
<td>science laboratory</td>
<td>3360</td>
</tr>
<tr>
<td>sports hall</td>
<td>3370</td>
</tr>
<tr>
<td>school library</td>
<td>3380</td>
</tr>
<tr>
<td>office</td>
<td>3390</td>
</tr>
<tr>
<td>lecture theatre</td>
<td>3400</td>
</tr>
<tr>
<td>refectory</td>
<td>3410</td>
</tr>
<tr>
<td>function room</td>
<td>3420</td>
</tr>
</tbody>
</table>
### C.2 City furniture

<table>
<thead>
<tr>
<th>CityFurnitureClassType</th>
<th>Code list derived from German authoritative standards ALKIS/ATKIS (<a href="http://www.adv-online.de">www.adv-online.de</a>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 traffic</td>
<td></td>
</tr>
<tr>
<td>1010 communication</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CityFurnitureFunctionType</th>
<th>Code list derived from German authoritative standards ALKIS/ATKIS (<a href="http://www.adv-online.de">www.adv-online.de</a>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 communication fixture</td>
<td></td>
</tr>
<tr>
<td>1010 telephone box</td>
<td></td>
</tr>
<tr>
<td>1020 postbox</td>
<td></td>
</tr>
<tr>
<td>1030 emergency call fixture</td>
<td></td>
</tr>
<tr>
<td>1040 fire detector</td>
<td></td>
</tr>
<tr>
<td>1050 police call post</td>
<td></td>
</tr>
<tr>
<td>1060 switching unit</td>
<td></td>
</tr>
<tr>
<td>1070 road sign</td>
<td></td>
</tr>
<tr>
<td>1080 traffic light</td>
<td></td>
</tr>
<tr>
<td>1090 free-standing sign</td>
<td></td>
</tr>
<tr>
<td>1100 free-standing warning sign</td>
<td></td>
</tr>
<tr>
<td>1110 bus stop</td>
<td></td>
</tr>
<tr>
<td>1120 milestone</td>
<td></td>
</tr>
<tr>
<td>1130 rail level crossing</td>
<td></td>
</tr>
<tr>
<td>1140 gate</td>
<td></td>
</tr>
<tr>
<td>1150 streetlamp, latern or candelabra</td>
<td></td>
</tr>
<tr>
<td>1160 column</td>
<td></td>
</tr>
<tr>
<td>1170 lamp post</td>
<td></td>
</tr>
<tr>
<td>1180 flagpole</td>
<td></td>
</tr>
<tr>
<td>1190 street sink box</td>
<td></td>
</tr>
<tr>
<td>1200 rubbish bin</td>
<td></td>
</tr>
<tr>
<td>1210 clock</td>
<td></td>
</tr>
<tr>
<td>1220 directional spot light</td>
<td></td>
</tr>
<tr>
<td>1230 floodlight mast</td>
<td></td>
</tr>
<tr>
<td>1240 windmill</td>
<td></td>
</tr>
<tr>
<td>1250 solar cell</td>
<td></td>
</tr>
<tr>
<td>1260 water wheel</td>
<td></td>
</tr>
</tbody>
</table>

1020 others
1030 others
1040 others
1050 others
1060 others
1070 others
1080 others
1090 others
1100 others
1110 others
1120 others
1130 others
1140 others
1150 others
1160 others
1170 others
1180 others
1190 others
1200 others
1210 others
1220 others
1230 others
1240 others
1250 others
1260 others
## C.3 Land use

### LandUseClassType

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Settlement Area</td>
<td>3000</td>
<td>Vegetation</td>
</tr>
<tr>
<td>1100</td>
<td>Undeveloped Area</td>
<td>4000</td>
<td>Water</td>
</tr>
<tr>
<td>2000</td>
<td>Traffic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LandUseFunctionType

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>Residential</td>
<td>2050</td>
<td>Track</td>
</tr>
<tr>
<td>1020</td>
<td>Industry and Business</td>
<td>2060</td>
<td>Square</td>
</tr>
<tr>
<td>1030</td>
<td>Mixed use</td>
<td>3010</td>
<td>Grassland</td>
</tr>
<tr>
<td>1040</td>
<td>Special Function Area</td>
<td>3020</td>
<td>Agriculture</td>
</tr>
<tr>
<td>1050</td>
<td>Monument</td>
<td>3030</td>
<td>Forest</td>
</tr>
<tr>
<td>1060</td>
<td>Dump</td>
<td>3040</td>
<td>Grove</td>
</tr>
<tr>
<td>1070</td>
<td>Mining</td>
<td>3050</td>
<td>Heath</td>
</tr>
<tr>
<td>1110</td>
<td>Park</td>
<td>3060</td>
<td>Moor</td>
</tr>
<tr>
<td>1120</td>
<td>Cemetary</td>
<td>3070</td>
<td>Marsh</td>
</tr>
<tr>
<td>1130</td>
<td>Sports, leisure and recreation</td>
<td>3080</td>
<td>Untilled land</td>
</tr>
<tr>
<td>1140</td>
<td>Open pit, quarry</td>
<td>4010</td>
<td>River</td>
</tr>
<tr>
<td>2010</td>
<td>Road</td>
<td>4020</td>
<td>Standing Waterbody</td>
</tr>
<tr>
<td>2020</td>
<td>Railway</td>
<td>4030</td>
<td>Harbour</td>
</tr>
<tr>
<td>2030</td>
<td>Airfield</td>
<td>4040</td>
<td>Sea</td>
</tr>
<tr>
<td>2040</td>
<td>Shipping</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LandUseUsageType

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Civil</td>
<td>1300</td>
<td>Religious</td>
</tr>
<tr>
<td>1100</td>
<td>Private</td>
<td>1400</td>
<td>Military</td>
</tr>
<tr>
<td>1200</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### C.4 MimeType

<table>
<thead>
<tr>
<th>MimeType</th>
<th>Type</th>
<th>Model/Format</th>
<th>X3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>model/vrml</td>
<td>VRML97</td>
<td>model/x3d+xml</td>
<td>X3D</td>
</tr>
<tr>
<td>application/x-3ds</td>
<td>3ds max</td>
<td>model/x3d+binary</td>
<td>X3D</td>
</tr>
<tr>
<td>application/dxf</td>
<td>AutoCad DXF</td>
<td>image/gif</td>
<td>*.gif images</td>
</tr>
<tr>
<td>application/x-autocad</td>
<td>AutoCad DXF</td>
<td>image/jpeg</td>
<td>*.jpeg, *.jpg images</td>
</tr>
<tr>
<td>application/x-dxf</td>
<td>AutoCad DXF</td>
<td>image/png</td>
<td>*.png images</td>
</tr>
<tr>
<td>application/acad</td>
<td>AutoCad DWG</td>
<td>image/tiff</td>
<td>*.tiff, *.tif images</td>
</tr>
<tr>
<td>application/x-shockwave-flash</td>
<td>Shockwave 3D</td>
<td>image/bmp</td>
<td>*.bmp images</td>
</tr>
<tr>
<td>model/x3d+vrml</td>
<td>X3D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The MIME types given in this table are defined by the Internet Assigned Numbers Authority (IANA), see http://www.iana.org/. Generally, the MIME format is standardized by the Internet Engineering Task Force (IETF), see http://www.ietf.org/. Unlike the other code lists the MIME types are not represented by numbers, but instead use their given identifier. This code list is not exhaustive. It contains a selection of frequently used MIME types.
### C.5 Vegetation

#### PlantClassType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>shrub</td>
</tr>
<tr>
<td>1010</td>
<td>low plants</td>
</tr>
<tr>
<td>1020</td>
<td>medium high plants</td>
</tr>
<tr>
<td>1030</td>
<td>high plants</td>
</tr>
<tr>
<td>1040</td>
<td>grasses</td>
</tr>
<tr>
<td>1050</td>
<td>ferns</td>
</tr>
<tr>
<td>1060</td>
<td>coniferous tree</td>
</tr>
<tr>
<td>1070</td>
<td>deciduous tree</td>
</tr>
<tr>
<td>1080</td>
<td>bushes</td>
</tr>
<tr>
<td>1090</td>
<td>aquatic plants</td>
</tr>
<tr>
<td>1100</td>
<td>climber</td>
</tr>
<tr>
<td>1110</td>
<td>unknown</td>
</tr>
</tbody>
</table>

#### PlantFunctionType

Code list identically specified as PlantClassType

#### PlantCoverClassType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>Lemnetea</td>
</tr>
<tr>
<td>1020</td>
<td>Asplenietea rupestris</td>
</tr>
<tr>
<td>1030</td>
<td>Adiantetea</td>
</tr>
<tr>
<td>1040</td>
<td>Thlaspietea rotundifolii</td>
</tr>
<tr>
<td>1050</td>
<td>Crithmo-Limonietea</td>
</tr>
<tr>
<td>1060</td>
<td>Ammophietea</td>
</tr>
<tr>
<td>1070</td>
<td>Cakiletea maritimae halophile</td>
</tr>
<tr>
<td>1080</td>
<td>Secalineta</td>
</tr>
<tr>
<td>1090</td>
<td>Chenopodietea</td>
</tr>
<tr>
<td>1100</td>
<td>Onopordetea</td>
</tr>
<tr>
<td>1110</td>
<td>Epilobietea angustifolii</td>
</tr>
<tr>
<td>1120</td>
<td>Bidentetea tripartiti</td>
</tr>
<tr>
<td>1130</td>
<td>Zoosteretea marinae halophile</td>
</tr>
<tr>
<td>1140</td>
<td>Ruppietea maritimae</td>
</tr>
<tr>
<td>1150</td>
<td>Potametea haftende</td>
</tr>
<tr>
<td>1160</td>
<td>Litorelletea</td>
</tr>
<tr>
<td>1170</td>
<td>Plantaginetea majoris</td>
</tr>
<tr>
<td>1180</td>
<td>Isoeto-Nanojuncetea</td>
</tr>
<tr>
<td>1190</td>
<td>Montino-Cardaminetea</td>
</tr>
<tr>
<td>1200</td>
<td>Corynephoretea</td>
</tr>
<tr>
<td>1210</td>
<td>Asteretea tripolium</td>
</tr>
<tr>
<td>1220</td>
<td>Salicornietea</td>
</tr>
<tr>
<td>1230</td>
<td>Juncetea maritimae</td>
</tr>
<tr>
<td>1240</td>
<td>Phragmitetea</td>
</tr>
<tr>
<td>1250</td>
<td>Spartinetea</td>
</tr>
<tr>
<td>1260</td>
<td>Sedo-Scleranthea</td>
</tr>
<tr>
<td>1270</td>
<td>Salicetea herbaceae</td>
</tr>
<tr>
<td>1280</td>
<td>Arrhenatheretea</td>
</tr>
<tr>
<td>1290</td>
<td>Molinio-Juncetea</td>
</tr>
<tr>
<td>1300</td>
<td>Scheuchzerio-Caricetea fuscae azidophile</td>
</tr>
<tr>
<td>1310</td>
<td>Festuco-Brometea</td>
</tr>
<tr>
<td>1320</td>
<td>Elyno-Seslerietea</td>
</tr>
<tr>
<td>1330</td>
<td>Cariceta curvulae azidophile</td>
</tr>
<tr>
<td>1340</td>
<td>Calluno-Ulicetea</td>
</tr>
<tr>
<td>1350</td>
<td>Oxyccoco-Sphagnetea</td>
</tr>
<tr>
<td>1360</td>
<td>Saliceta purpureae</td>
</tr>
<tr>
<td>1370</td>
<td>Betulo-Adenostyletea</td>
</tr>
<tr>
<td>1380</td>
<td>Alneta glutinosae</td>
</tr>
<tr>
<td>1390</td>
<td>Erico-Pinetea</td>
</tr>
<tr>
<td>1400</td>
<td>Vaccinio-Piceetea</td>
</tr>
<tr>
<td>1410</td>
<td>Querceta robori-petraeae</td>
</tr>
<tr>
<td>1420</td>
<td>Querco-Fagetea</td>
</tr>
<tr>
<td>1430</td>
<td>Crithmo-Staticetea</td>
</tr>
<tr>
<td>1440</td>
<td>Tuberarietea guttati</td>
</tr>
<tr>
<td>1450</td>
<td>Junceta maritimae</td>
</tr>
<tr>
<td>1460</td>
<td>Thero-Brachypodietea</td>
</tr>
<tr>
<td>1470</td>
<td>Ononido-Rosmarinetea</td>
</tr>
<tr>
<td>1480</td>
<td>Nerio-Tamaricetea</td>
</tr>
<tr>
<td>1490</td>
<td>Pegano-Salsoletea</td>
</tr>
<tr>
<td>1500</td>
<td>Cisto-Lavanduletea</td>
</tr>
<tr>
<td>1510</td>
<td>Querceta ilicis</td>
</tr>
<tr>
<td>1520</td>
<td>Populetea albae</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

#### PlantCoverFunctionType

Code list identically specified as PlantCoverClassType
<table>
<thead>
<tr>
<th>Code</th>
<th>Species Type</th>
<th>Code</th>
<th>Species Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1640</td>
<td>Abies alba</td>
<td>1790</td>
<td>Acer circinatum</td>
</tr>
<tr>
<td>1650</td>
<td>Abies cephalonica</td>
<td>1800</td>
<td>Acer Davidii</td>
</tr>
<tr>
<td>1660</td>
<td>Abies concolor</td>
<td>1810</td>
<td>Acer ginnala Maxim</td>
</tr>
<tr>
<td>1670</td>
<td>Abies grandis</td>
<td>1820</td>
<td>Acer grosserii</td>
</tr>
<tr>
<td>1680</td>
<td>Abies homolepsis</td>
<td>1830</td>
<td>Acer monspessulanum</td>
</tr>
<tr>
<td>1690</td>
<td>Abies koreana</td>
<td>1840</td>
<td>Acer negundo</td>
</tr>
<tr>
<td>1700</td>
<td>Abies lasiocarpa</td>
<td>1850</td>
<td>Acer palmatum</td>
</tr>
<tr>
<td>1710</td>
<td>Abies nordmanniana</td>
<td>1860</td>
<td>Acer platanoides</td>
</tr>
<tr>
<td>1720</td>
<td>Abies pinsapo</td>
<td>1870</td>
<td>Acer platanoides 'Crimson King'</td>
</tr>
<tr>
<td>1730</td>
<td>Abies procer a</td>
<td>1880</td>
<td>Acer pseudoplatanus</td>
</tr>
<tr>
<td>1740</td>
<td>Abies procer a 'Glauc a'</td>
<td>1890</td>
<td>Acer rubrum</td>
</tr>
<tr>
<td>1750</td>
<td>Abies veitchii</td>
<td>1900</td>
<td>Acer saccharinum</td>
</tr>
<tr>
<td>1760</td>
<td>Acer campéstre</td>
<td>1910</td>
<td>Acer saccharum Marsch</td>
</tr>
<tr>
<td>1770</td>
<td>Acer capillipes</td>
<td>1920</td>
<td>Acer tartaricum</td>
</tr>
<tr>
<td>1780</td>
<td>Acer cappadocicum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## C.6 Transportation

### AuxiliaryTrafficAreaFunctionType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>soft shoulder</td>
</tr>
<tr>
<td>1010</td>
<td>hard shoulder</td>
</tr>
<tr>
<td>1020</td>
<td>green area</td>
</tr>
<tr>
<td>1030</td>
<td>middle lane</td>
</tr>
<tr>
<td>1040</td>
<td>lay by</td>
</tr>
<tr>
<td>1100</td>
<td>parking bay</td>
</tr>
<tr>
<td>1200</td>
<td>ditch</td>
</tr>
<tr>
<td>1210</td>
<td>drainage</td>
</tr>
<tr>
<td>1220</td>
<td>kerbstone</td>
</tr>
<tr>
<td>1230</td>
<td>flower tub</td>
</tr>
<tr>
<td>1300</td>
<td>traffic island</td>
</tr>
<tr>
<td>1400</td>
<td>bank</td>
</tr>
<tr>
<td>1410</td>
<td>embankment, dike</td>
</tr>
<tr>
<td>1420</td>
<td>railroad embankment</td>
</tr>
<tr>
<td>1430</td>
<td>noise protection</td>
</tr>
<tr>
<td>1440</td>
<td>noise protection wall</td>
</tr>
<tr>
<td>1500</td>
<td>noise guard bar</td>
</tr>
<tr>
<td>1600</td>
<td>towpath</td>
</tr>
<tr>
<td>1700</td>
<td>others</td>
</tr>
</tbody>
</table>

### TrafficAreaFunctionType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>driving lane</td>
</tr>
<tr>
<td>2</td>
<td>footpath</td>
</tr>
<tr>
<td>3</td>
<td>cyclepath</td>
</tr>
<tr>
<td>4</td>
<td>combined foot-/cyclepath</td>
</tr>
<tr>
<td>5</td>
<td>square</td>
</tr>
<tr>
<td>6</td>
<td>car_park</td>
</tr>
<tr>
<td>7</td>
<td>parking_lay_by</td>
</tr>
<tr>
<td>8</td>
<td>rail</td>
</tr>
<tr>
<td>9</td>
<td>rail_road_combined</td>
</tr>
<tr>
<td>10</td>
<td>drainage</td>
</tr>
<tr>
<td>11</td>
<td>road marking</td>
</tr>
<tr>
<td>12</td>
<td>road_marking_direction</td>
</tr>
<tr>
<td>13</td>
<td>road_marking_lane</td>
</tr>
<tr>
<td>14</td>
<td>road_marking_restricted</td>
</tr>
<tr>
<td>15</td>
<td>road_marking_crosswalk</td>
</tr>
<tr>
<td>16</td>
<td>road_marking_stop</td>
</tr>
<tr>
<td>17</td>
<td>road_marking_other</td>
</tr>
<tr>
<td>18</td>
<td>overhead_wire (trolley)</td>
</tr>
<tr>
<td>19</td>
<td>train platform</td>
</tr>
<tr>
<td>20</td>
<td>crosswalk</td>
</tr>
<tr>
<td>21</td>
<td>barrier</td>
</tr>
<tr>
<td>22</td>
<td>stairs</td>
</tr>
<tr>
<td>23</td>
<td>escalator</td>
</tr>
<tr>
<td>24</td>
<td>filtering lane</td>
</tr>
<tr>
<td>25</td>
<td>airport_runway</td>
</tr>
<tr>
<td>26</td>
<td>airport_taxiway</td>
</tr>
<tr>
<td>27</td>
<td>airport apron</td>
</tr>
<tr>
<td>28</td>
<td>airport_heliport</td>
</tr>
<tr>
<td>29</td>
<td>airport_runway_marking</td>
</tr>
<tr>
<td>30</td>
<td>green spaces</td>
</tr>
<tr>
<td>31</td>
<td>recreation</td>
</tr>
<tr>
<td>32</td>
<td>bus_lay_by</td>
</tr>
<tr>
<td>33</td>
<td>motorway</td>
</tr>
<tr>
<td>34</td>
<td>motorway_entry</td>
</tr>
<tr>
<td>35</td>
<td>motorway_exit</td>
</tr>
<tr>
<td>36</td>
<td>motorway_emergency lane</td>
</tr>
<tr>
<td>37</td>
<td>private_area</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

### TrafficAreaUsageType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pedestrian</td>
</tr>
<tr>
<td>2</td>
<td>car</td>
</tr>
<tr>
<td>3</td>
<td>truck</td>
</tr>
<tr>
<td>4</td>
<td>bus, taxi</td>
</tr>
<tr>
<td>5</td>
<td>train</td>
</tr>
<tr>
<td>6</td>
<td>bicycle</td>
</tr>
<tr>
<td>7</td>
<td>motorcycle</td>
</tr>
<tr>
<td>8</td>
<td>tram, streetcar</td>
</tr>
<tr>
<td>9</td>
<td>boat, ferry, ship</td>
</tr>
<tr>
<td>10</td>
<td>teleferic</td>
</tr>
<tr>
<td>11</td>
<td>aeroplane</td>
</tr>
<tr>
<td>12</td>
<td>helicopter</td>
</tr>
<tr>
<td>13</td>
<td>taxi</td>
</tr>
<tr>
<td>14</td>
<td>horse</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>
**TrafficSurfaceMaterialType**

Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>asphalt</td>
</tr>
<tr>
<td>2</td>
<td>concrete</td>
</tr>
<tr>
<td>3</td>
<td>pavement</td>
</tr>
<tr>
<td>4</td>
<td>cobblestone</td>
</tr>
<tr>
<td>5</td>
<td>gravel</td>
</tr>
<tr>
<td>6</td>
<td>rail with bed</td>
</tr>
<tr>
<td>7</td>
<td>rail without bed</td>
</tr>
<tr>
<td>8</td>
<td>soil</td>
</tr>
<tr>
<td>9</td>
<td>sand</td>
</tr>
<tr>
<td>10</td>
<td>grass</td>
</tr>
<tr>
<td>11</td>
<td>wood</td>
</tr>
<tr>
<td>12</td>
<td>steel</td>
</tr>
<tr>
<td>13</td>
<td>marble</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

**TransportationComplexClass**

Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>private</td>
</tr>
<tr>
<td>1010</td>
<td>common</td>
</tr>
<tr>
<td>1020</td>
<td>civil</td>
</tr>
<tr>
<td>1030</td>
<td>military</td>
</tr>
<tr>
<td>1040</td>
<td>road traffic</td>
</tr>
<tr>
<td>1050</td>
<td>air traffic</td>
</tr>
<tr>
<td>1060</td>
<td>rail traffic</td>
</tr>
<tr>
<td>1070</td>
<td>waterway</td>
</tr>
<tr>
<td>1080</td>
<td>subway</td>
</tr>
<tr>
<td>1090</td>
<td>others</td>
</tr>
</tbody>
</table>

**TransportationComplexFunction**

Code list proposed by the SIG 3D

<table>
<thead>
<tr>
<th>Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>road</td>
</tr>
<tr>
<td>1010</td>
<td>freeway/motorway</td>
</tr>
<tr>
<td>1020</td>
<td>highway/national primary road</td>
</tr>
<tr>
<td>1030</td>
<td>land road</td>
</tr>
<tr>
<td>1040</td>
<td>district road</td>
</tr>
<tr>
<td>1050</td>
<td>main through-road</td>
</tr>
<tr>
<td>1060</td>
<td>main through-road</td>
</tr>
<tr>
<td>1100</td>
<td>freeway interchange/ highway junction</td>
</tr>
<tr>
<td>1110</td>
<td>junction</td>
</tr>
<tr>
<td>1200</td>
<td>road</td>
</tr>
<tr>
<td>1210</td>
<td>driveway</td>
</tr>
<tr>
<td>1220</td>
<td>footpath/footway</td>
</tr>
<tr>
<td>1230</td>
<td>hiking trail</td>
</tr>
<tr>
<td>1240</td>
<td>bikeway/cycle-path</td>
</tr>
<tr>
<td>1250</td>
<td>bridleway/bridlepath</td>
</tr>
<tr>
<td>1260</td>
<td>main agricultural road</td>
</tr>
<tr>
<td>1270</td>
<td>agricultural road</td>
</tr>
<tr>
<td>1280</td>
<td>bikeway/footway</td>
</tr>
<tr>
<td>1290</td>
<td>dead-end road</td>
</tr>
<tr>
<td>1300</td>
<td>dead-end road</td>
</tr>
<tr>
<td>1400</td>
<td>lane</td>
</tr>
<tr>
<td>1410</td>
<td>lane, one direction</td>
</tr>
<tr>
<td>1420</td>
<td>lane, both direction</td>
</tr>
<tr>
<td>1500</td>
<td>pedestrian zone</td>
</tr>
<tr>
<td>1600</td>
<td>place</td>
</tr>
<tr>
<td>1610</td>
<td>parking area</td>
</tr>
<tr>
<td>1620</td>
<td>marketplace</td>
</tr>
<tr>
<td>1700</td>
<td>service area</td>
</tr>
<tr>
<td>1800</td>
<td>rail transport</td>
</tr>
<tr>
<td>1805</td>
<td>rail</td>
</tr>
<tr>
<td>1810</td>
<td>urban/city train</td>
</tr>
<tr>
<td>1855</td>
<td>railway track</td>
</tr>
<tr>
<td>1860</td>
<td>magnetic levitation train</td>
</tr>
<tr>
<td>1900</td>
<td>railway station</td>
</tr>
<tr>
<td>1910</td>
<td>stop</td>
</tr>
<tr>
<td>1920</td>
<td>station</td>
</tr>
<tr>
<td>2000</td>
<td>power-wheel</td>
</tr>
<tr>
<td>2100</td>
<td>airport</td>
</tr>
<tr>
<td>2110</td>
<td>international airport</td>
</tr>
<tr>
<td>2120</td>
<td>regional airport</td>
</tr>
<tr>
<td>2130</td>
<td>landing place</td>
</tr>
<tr>
<td>2140</td>
<td>heliport</td>
</tr>
<tr>
<td>2150</td>
<td>landing place</td>
</tr>
<tr>
<td>2160</td>
<td>gliding airfield</td>
</tr>
<tr>
<td>2170</td>
<td>taxiway</td>
</tr>
<tr>
<td>2180</td>
<td>apron</td>
</tr>
<tr>
<td>2190</td>
<td>runway</td>
</tr>
<tr>
<td>2200</td>
<td>canal</td>
</tr>
<tr>
<td>2300</td>
<td>harbor</td>
</tr>
<tr>
<td>2310</td>
<td>pleasure craft harbour</td>
</tr>
<tr>
<td>2400</td>
<td>ferry</td>
</tr>
<tr>
<td>2410</td>
<td>car ferry</td>
</tr>
<tr>
<td>2420</td>
<td>train ferry</td>
</tr>
<tr>
<td>2430</td>
<td>ferry</td>
</tr>
<tr>
<td>2500</td>
<td>landing stage</td>
</tr>
<tr>
<td>2600</td>
<td>waterway I order</td>
</tr>
<tr>
<td>2610</td>
<td>navigable river</td>
</tr>
<tr>
<td>2620</td>
<td>inland navigation waterway 0</td>
</tr>
<tr>
<td>2621</td>
<td>inland navigation waterway 0</td>
</tr>
<tr>
<td>2622</td>
<td>inland navigation waterway I</td>
</tr>
<tr>
<td>2623</td>
<td>inland navigation waterway II</td>
</tr>
<tr>
<td>2624</td>
<td>inland navigation waterway III</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>1815</td>
<td>city railway</td>
</tr>
<tr>
<td>1820</td>
<td>tram</td>
</tr>
<tr>
<td>1825</td>
<td>subway</td>
</tr>
<tr>
<td>1830</td>
<td>funicular/mountain railway</td>
</tr>
<tr>
<td>1835</td>
<td>mountain railway</td>
</tr>
<tr>
<td>1840</td>
<td>chairlift</td>
</tr>
<tr>
<td>1845</td>
<td>ski-lift/ki tow lift</td>
</tr>
<tr>
<td>1850</td>
<td>suspension railway</td>
</tr>
</tbody>
</table>

**TransportationComplexUsage**

Code list identically specified as *TransportationComplexFunction*
### Water Body

#### WaterBodyClassType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>sea</td>
</tr>
<tr>
<td>1010</td>
<td>tidal waterbody</td>
</tr>
<tr>
<td>1020</td>
<td>watercourse</td>
</tr>
<tr>
<td>1030</td>
<td>river / stream</td>
</tr>
<tr>
<td>1040</td>
<td>ditch</td>
</tr>
<tr>
<td>1050</td>
<td>spring / water hole</td>
</tr>
<tr>
<td>1060</td>
<td>lake / pont</td>
</tr>
<tr>
<td>1070</td>
<td>bayou</td>
</tr>
<tr>
<td>1080</td>
<td>body of standing water</td>
</tr>
<tr>
<td>1090</td>
<td>waterfall</td>
</tr>
<tr>
<td>1100</td>
<td>rapids</td>
</tr>
<tr>
<td>1110</td>
<td>swamp</td>
</tr>
<tr>
<td>1120</td>
<td>sinkhole (karst)</td>
</tr>
<tr>
<td>1130</td>
<td>ephemeral watercourse</td>
</tr>
</tbody>
</table>

#### WaterBodyFunctionType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>nature-sanctuary</td>
</tr>
<tr>
<td>1010</td>
<td>protected waterbody</td>
</tr>
<tr>
<td>1020</td>
<td>reservoir</td>
</tr>
<tr>
<td>1030</td>
<td>retention waterbody</td>
</tr>
<tr>
<td>1040</td>
<td>flood plain waterbody</td>
</tr>
<tr>
<td>1050</td>
<td>waterway</td>
</tr>
<tr>
<td>1060</td>
<td>harbor waterbody</td>
</tr>
<tr>
<td>1070</td>
<td>sluice waterbody</td>
</tr>
<tr>
<td>1080</td>
<td>sewage system</td>
</tr>
</tbody>
</table>

#### WaterBodyUsageType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>sanctuary</td>
</tr>
<tr>
<td>1010</td>
<td>recreation / sports</td>
</tr>
<tr>
<td>1020</td>
<td>drinking water supply</td>
</tr>
<tr>
<td>1030</td>
<td>hydroelectric water supply</td>
</tr>
<tr>
<td>1040</td>
<td>ocean shipping</td>
</tr>
<tr>
<td>1050</td>
<td>inland shipping</td>
</tr>
<tr>
<td>1060</td>
<td>sewer</td>
</tr>
<tr>
<td>1070</td>
<td>port</td>
</tr>
<tr>
<td>1080</td>
<td>anchorage</td>
</tr>
<tr>
<td>1090</td>
<td>public use</td>
</tr>
<tr>
<td>1100</td>
<td>private use</td>
</tr>
</tbody>
</table>

#### WaterLevelType

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>MSL - Mean Sea Level</td>
</tr>
<tr>
<td>1010</td>
<td>LAT - Lowest Astronomical Tide</td>
</tr>
<tr>
<td>1020</td>
<td>National Water Level</td>
</tr>
<tr>
<td>1090</td>
<td>Hundred Year Flood</td>
</tr>
<tr>
<td>1100</td>
<td>highest known water level</td>
</tr>
<tr>
<td>1110</td>
<td>critical low-water level</td>
</tr>
<tr>
<td>1030</td>
<td>Mean High Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1040</td>
<td>Extreme High Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1050</td>
<td>Mean Low Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1060</td>
<td>Extreme Low Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1070</td>
<td>Mean Water Level (watercourse)</td>
</tr>
<tr>
<td>1080</td>
<td>critical high-water level</td>
</tr>
</tbody>
</table>
### Annex D

(informative)

#### Overview of employed GML3 geometry classes

<table>
<thead>
<tr>
<th>Abstract GML classes referenced in CityGML</th>
<th>GML subclass actually used in CityGML</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Geometry</td>
<td></td>
</tr>
<tr>
<td>_Solid</td>
<td>Solid (boundary is restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td></td>
<td>CompositeSolid</td>
</tr>
<tr>
<td>_Surface</td>
<td>Polygon (with holes, modelled by Rings. The boundary is restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td></td>
<td>OrientableSurface (base surface is restricted to a Polygon)</td>
</tr>
<tr>
<td></td>
<td>TexturedSurface (defined in CityGML’s TexturedSurface module, not in GML. This modelling approach has been marked deprecated. For restrictions see OrientableSurface)</td>
</tr>
<tr>
<td></td>
<td>CompositeSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td></td>
<td>TriangulatedSurface</td>
</tr>
<tr>
<td></td>
<td>Tin</td>
</tr>
<tr>
<td>_Curve</td>
<td>LineString</td>
</tr>
<tr>
<td></td>
<td>CompositeCurve (members are restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>_Coverage</td>
<td>RectifiedGridCoverage</td>
</tr>
<tr>
<td>AbstractGeometricAggregate</td>
<td>MultiSolid</td>
</tr>
<tr>
<td></td>
<td>MultiSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td></td>
<td>MultiCurve (members are restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td></td>
<td>MultiPoint</td>
</tr>
<tr>
<td></td>
<td>GeometricComplex (restricted to connected, linear networks)</td>
</tr>
</tbody>
</table>
Annex E
(informative)

Overview of the assignment of features to LODs

The following table lists all feature types of CityGML. For each feature type, all non-spatial and spatial properties are given, including their type. Each feature is assigned a range of LOD in which it may occur, and for each property the LOD in which it represents the feature is stated. Each feature type’s name is preceded with a prefix indicating the CityGML module defining the feature type. For a list of prefixes and associated CityGML modules, please refer to chapter 4.3 and chapter 7.

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Property</th>
<th>Type</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>core:CityModelType</td>
<td>gml:FeaturePropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:appearanceMember</td>
<td>app:AppearancePropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfCityModel</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>core:AbstractCityObjectType</td>
<td>xs:dateTime</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>externalReference</td>
<td>core:ExternalReferenceType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>generalizesTo</td>
<td>core:GeneralizationRelationType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:appearance</td>
<td>app:AppearancePropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>gen: genericAttribute</td>
<td>gen:AbstractGenericAttributeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfCityObject</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>core:AbstractSiteType</td>
<td>xs:dateTime</td>
<td>1 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfSite</td>
<td>xs:anyType</td>
<td>1 – 4</td>
<td></td>
</tr>
<tr>
<td>core:AddressType</td>
<td>core:xalAddressPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>multiPoint</td>
<td>gml:MultiPointPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfAddress</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:AppearanceType</td>
<td>xs:string</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>surfaceDataMember</td>
<td>app:SurfaceDataPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfAppearance</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:AbstractSurfaceDataType</td>
<td>xs:dateTime</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>isFront</td>
<td>xs:Boolean</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfSurfaceData</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:AbstractTextureType</td>
<td>xs:dateTime</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>imageURI</td>
<td>xs:anyURI</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>mimeType</td>
<td>core:MimeTypeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>textureType</td>
<td>app:TextureTypeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>wrapMode</td>
<td>app:WrapModeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>borderColor</td>
<td>app:ColorPlusOpacity</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTexture</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:ParameterizedTextureType</td>
<td>xs:dateTime</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>app:TextureAssociationType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfParameterizedTexture</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:GeoreferencedTextureType</td>
<td>xs:dateTime</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>preferWorldFile</td>
<td>xs:boolean</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>referencePoint</td>
<td>gml:PointPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>orientation</td>
<td>core:TransformationMatrix2x2Type</td>
<td>0 – 4</td>
<td></td>
</tr>
</tbody>
</table>
target xs:anyURI 0 – 4
_GenericApplicationProperty OfGeoreferencedTexture xs:anyType 0 – 4

app:X3DMaterialType

ambientIntensity core:doubleBetween0and1 0 – 4
diffuseColor app:Color 0 – 4
emissiveColor app:Color 0 – 4
specularColor app:Color 0 – 4
shininess core:doubleBetween0and1 0 – 4
transparency core:doubleBetween0and1 0 – 4
isSmooth xs:boolean 0 – 4
target xs:anyURI 0 – 4
_GenericApplicationProperty OfX3DMaterial xs:anyType 0 – 4

bldg:AbstractBuildingType

class bldg:AbstractBuildingClassType 1 – 4
function bldg:AbstractBuildingFunctionType 1 – 4
usage bldg:AbstractBuildingUsageType 1 – 4
yearOfConstruction xs:gYear 1 – 4
yearOfDemolition xs:gYear 1 – 4
roofType bldg:AbstractBuildingRoofTypeType 1 – 4
measuredHeight gml:LengthType 1 – 4
storeysAboveGround xs:nonNegativeInteger 1 – 4
storeysBelowGround xs:nonNegativeInteger 1 – 4
storeyHeightsAboveGround gml:MeasureOrNullListType 1 – 4
storeyHeightsBelowGround gml:MeasureOrNullListType 1 – 4
lod1Solid gml:SolidPropertyType 1
lod1MultiSurface gml:MultiSurfacePropertyType 1
lod1TerrainIntersection gml:MultiSurfacePropertyType 1
lod2Solid gml:SolidPropertyType 2
lod2MultiSurface gml:MultiSurfacePropertyType 2
lod2MultiCurve gml:MultiSurfacePropertyType 2
lod2TerrainIntersection gml:MultiSurfacePropertyType 2
outerBuildingInstallation bldg:AbstractBuildingInstallationPropertyType 2 – 4
interiorBuildingInstallation bldg:AbstractBuildingInstallationPropertyType 4
boundedBy bldg:AbstractBuildingInstallationPropertyType 2 – 4
lod3Solid gml:SolidPropertyType 3
lod3MultiSurface gml:MultiSurfacePropertyType 3
lod3MultiCurve gml:MultiSurfacePropertyType 3
lod3TerrainIntersection gml:MultiSurfacePropertyType 3
lod4Solid gml:SolidPropertyType 4
lod4MultiSurface gml:MultiSurfacePropertyType 4
lod4MultiCurve gml:MultiSurfacePropertyType 4
lod4TerrainIntersection gml:MultiSurfacePropertyType 4
interiorRoom bldg:AbstractBuildingInstallationPropertyType 4
consistsOfBuildingPart bldg:AbstractBuildingInstallationPropertyType 1 – 4
address core:AbstractBuildingInstallationPropertyType 1 – 4
_GenericApplicationProperty OfAbstractBuilding xs:anyType 1 – 4

bldg:BuildingType

_GenericApplicationProperty xs:anyType 1 – 4

bldg:BuildingPartType

_GenericApplicationProperty xs:anyType 1 – 4

bldg:BuildingInstallationType

class bldg:AbstractBuildingClassType 1 – 4
function bldg:AbstractBuildingFunctionType 1 – 4
usage bldg:AbstractBuildingUsageType 1 – 4
lod2Geometry gml:GeometryPropertyType 2
lod3Geometry gml:GeometryPropertyType 3
lod4Geometry gml:GeometryPropertyType 4
_GenericApplicationProperty OfAbstractBuilding xs:anyType 2 – 4

bldg:BuildingInstallationProperty

_GenericApplicationProperty xs:anyType 1 – 4

bldg:BuildingPartProperty

_GenericApplicationProperty xs:anyType 1 – 4

Copyright © 2008 Open Geospatial Consortium, Inc. All Rights Reserved.
<table>
<thead>
<tr>
<th>bldg:IntBuildingInstallationType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>bldg:IntBuildingInstallationClassType 4</td>
</tr>
<tr>
<td>function</td>
<td>bldg:IntBuildingInstallationFunctionType 4</td>
</tr>
<tr>
<td>usage</td>
<td>bldg:IntBuildingInstallationUsageType 4</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 4</td>
</tr>
<tr>
<td>OfIntBuildingInstallation</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:AbstractBoundarySurfaceType</th>
<th>2 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType 2</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType 3</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType 4</td>
</tr>
<tr>
<td>opening</td>
<td>bldg:OpeningPropertyType 3 – 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 2 – 4</td>
</tr>
<tr>
<td>OfBoundarySurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:RoofSurfaceType</th>
<th>2 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 2 – 4</td>
</tr>
<tr>
<td>OfRoofSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:WallSurfaceType</th>
<th>2 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 2 – 4</td>
</tr>
<tr>
<td>OfWallSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:GroundSurfaceType</th>
<th>2 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 2 – 4</td>
</tr>
<tr>
<td>OfGroundSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:ClosureSurfaceType</th>
<th>2 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 2 – 4</td>
</tr>
<tr>
<td>OfClosureSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:FloorSurfaceType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 4</td>
</tr>
<tr>
<td>OfFloorSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:InteriorWallSurfaceType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 4</td>
</tr>
<tr>
<td>OfInteriorWallSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:CeilingSurfaceType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 4</td>
</tr>
<tr>
<td>OfCeilingSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:AbstractOpeningType</th>
<th>3 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType 3</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 3 – 4</td>
</tr>
<tr>
<td>OfOpening</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:WindowType</th>
<th>3 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 3 – 4</td>
</tr>
<tr>
<td>OfWindow</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:DoorType</th>
<th>3 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>core:AddressPropertyType 3 – 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 3 – 4</td>
</tr>
<tr>
<td>OfCeilingSurface</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:RoomType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>bldg:RoomClassType 4</td>
</tr>
<tr>
<td>function</td>
<td>bldg:RoomFunctionType 4</td>
</tr>
<tr>
<td>usage</td>
<td>bldg:RoomUsageType 4</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType 4</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType 4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>bldg:BoundarySurfacePropertyType 4</td>
</tr>
<tr>
<td>interiorFurniture</td>
<td>bldg:InteriorFurniturePropertyType 4</td>
</tr>
<tr>
<td>roomInstallation</td>
<td>bldg:IntBuildingInstallationPropertyType 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType 4</td>
</tr>
<tr>
<td>OfRoom</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bldg:BuildingFurnitureType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>bldg:BuildingFurnitureClassType 4</td>
</tr>
<tr>
<td>function</td>
<td>bldg:BuildingFurnitureFunctionType 4</td>
</tr>
<tr>
<td>usage</td>
<td>bldg:BuildingFurnitureUsageType 4</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType 4</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>bldg:ImplicitRepresentationPropertyType 4</td>
</tr>
<tr>
<td>Property</td>
<td>Datatype</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>OfBuildingFurniture</td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>frn:FurnitureFunctionType</td>
</tr>
<tr>
<td>class</td>
<td>frn:FurnitureClassType</td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>grp:CityObjectGroupType</td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>xs:string</td>
</tr>
<tr>
<td>class</td>
<td>xs:string</td>
</tr>
<tr>
<td>usage</td>
<td>xs:string</td>
</tr>
<tr>
<td>groupMember</td>
<td>grp:CityObjectGroupMemberType</td>
</tr>
<tr>
<td>parent</td>
<td>grp:CityObjectGroupParentType</td>
</tr>
<tr>
<td>geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>OfCityFurniture</td>
<td></td>
</tr>
<tr>
<td>lod0Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod0TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod0ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>gen:GenericCityObjectType</td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>xs:string</td>
</tr>
<tr>
<td>class</td>
<td>xs:string</td>
</tr>
<tr>
<td>usage</td>
<td>xs:string</td>
</tr>
<tr>
<td>lod0Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod0TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod0ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>luse:LandUseType</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>luse:LandUseClassType</td>
</tr>
<tr>
<td>function</td>
<td>luse:LandUseFunctionType</td>
</tr>
<tr>
<td>usage</td>
<td>luse:LandUseUsageType</td>
</tr>
<tr>
<td>lod0MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>OfReliefFeature</td>
<td></td>
</tr>
<tr>
<td>lod</td>
<td>core:integerBetween0and4</td>
</tr>
<tr>
<td>reliefComponent</td>
<td>dem:ReliefComponentPropertyType</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>OfReliefFeature</td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Property</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>dem:AbstractReliefComponent</td>
<td>lod</td>
</tr>
<tr>
<td></td>
<td>extent</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfReliefComponent</td>
</tr>
<tr>
<td>dem:TINReliefType</td>
<td>tin</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfTINRelief</td>
</tr>
<tr>
<td>dem:RasterReliefType</td>
<td>grid</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfRasterRelief</td>
</tr>
<tr>
<td>dem:MassPointReliefType</td>
<td>reliefPoints</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfMassPointRelief</td>
</tr>
<tr>
<td>dem:BreakLineReliefType</td>
<td>ridgeOrValleyLines</td>
</tr>
<tr>
<td></td>
<td>breaklines</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfBreakLineRelief</td>
</tr>
<tr>
<td>tran:AbstractTransportationObject</td>
<td>_GenericApplicationPropertyOfTransportationObject</td>
</tr>
<tr>
<td>tran:TransportationComplexType</td>
<td>function</td>
</tr>
<tr>
<td></td>
<td>usage</td>
</tr>
<tr>
<td></td>
<td>trafficArea</td>
</tr>
<tr>
<td></td>
<td>auxiliaryTrafficArea</td>
</tr>
<tr>
<td></td>
<td>lod0Network</td>
</tr>
<tr>
<td></td>
<td>lod1MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod2MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfTransportationComplex</td>
</tr>
<tr>
<td>tran:TrafficAreaType</td>
<td>usage</td>
</tr>
<tr>
<td></td>
<td>function</td>
</tr>
<tr>
<td></td>
<td>surfaceMaterial</td>
</tr>
<tr>
<td></td>
<td>lod1MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod2MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfTrafficArea</td>
</tr>
<tr>
<td>tran:AuxiliaryTrafficAreaType</td>
<td>function</td>
</tr>
<tr>
<td></td>
<td>surfaceMaterial</td>
</tr>
<tr>
<td></td>
<td>lod1MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod2MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSurface</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfAuxiliaryTrafficArea</td>
</tr>
<tr>
<td>tran:TrackType</td>
<td>_GenericApplicationPropertyOfTrack</td>
</tr>
<tr>
<td>tran:RoadType</td>
<td>_GenericApplicationPropertyOfRoad</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tran:RailwayType</td>
<td>_GenericApplicationProperty OfRailway xs:anyType 1–4</td>
</tr>
<tr>
<td>tran:SquareType</td>
<td>_GenericApplicationProperty OfSquare xs:anyType 1–4</td>
</tr>
<tr>
<td>veg:AbstractVegetationObjectType</td>
<td>_GenericApplicationProperty OfVegetationObject xs:anyType 1–4</td>
</tr>
<tr>
<td>veg:PlantCoverType</td>
<td>class veg:PlantCoverClassType 1–4</td>
</tr>
<tr>
<td></td>
<td>function veg:PlantCoverFunctionType 1–4</td>
</tr>
<tr>
<td></td>
<td>averageHeight gml:LengthType 1–4</td>
</tr>
<tr>
<td></td>
<td>lod1MultiSurface gml:MultiSurfacePropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod2MultiSurface gml:MultiSurfacePropertyType 2</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSurface gml:MultiSurfacePropertyType 3</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface gml:MultiSurfacePropertyType 4</td>
</tr>
<tr>
<td></td>
<td>lod1MultiSolid gml:MultiSolidPropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod2MultiSolid gml:MultiSolidPropertyType 2</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSolid gml:MultiSolidPropertyType 3</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfPlantCover xs:anyType 1–4</td>
</tr>
<tr>
<td>veg:SolitaryVegetationObjectType</td>
<td>class veg:PlantClassType 1–4</td>
</tr>
<tr>
<td></td>
<td>function veg:PlantFunctionType 1–4</td>
</tr>
<tr>
<td></td>
<td>height gml:LengthType 1–4</td>
</tr>
<tr>
<td></td>
<td>trunkDiameter gml:LengthType 1–4</td>
</tr>
<tr>
<td></td>
<td>crownDiameter gml:LengthType 1–4</td>
</tr>
<tr>
<td></td>
<td>lod1Geometry gml:GeometryPropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod2Geometry gml:GeometryPropertyType 2</td>
</tr>
<tr>
<td></td>
<td>lod3Geometry gml:GeometryPropertyType 3</td>
</tr>
<tr>
<td></td>
<td>lod4Geometry gml:GeometryPropertyType 4</td>
</tr>
<tr>
<td></td>
<td>lod1ImplicitRepresentation core:ImplicitRepresentationPropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod2ImplicitRepresentation core:ImplicitRepresentationPropertyType 2</td>
</tr>
<tr>
<td></td>
<td>lod3ImplicitRepresentation core:ImplicitRepresentationPropertyType 3</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation core:ImplicitRepresentationPropertyType 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfSolitaryVegetationObject xs:anyType 1–4</td>
</tr>
<tr>
<td>wtr:AbstractWaterObjectType</td>
<td>_GenericApplicationProperty OfWaterObject xs:anyType 0–4</td>
</tr>
<tr>
<td>wtr:WaterBodyType</td>
<td>class wtr:WaterBodyClassType 0–4</td>
</tr>
<tr>
<td></td>
<td>function wtr:WaterBodyFunctionType 0–4</td>
</tr>
<tr>
<td></td>
<td>usage wtr:WaterBodyUsageType 0–4</td>
</tr>
<tr>
<td></td>
<td>lod0MultiCurve gml:MultiCurvePropertyType 0</td>
</tr>
<tr>
<td></td>
<td>lod1MultiCurve gml:MultiCurvePropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod0MultiSurface gml:MultiSurfacePropertyType 0</td>
</tr>
<tr>
<td></td>
<td>lod1MultiSurface gml:MultiSurfacePropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod1Solid gml:SolidPropertyType 1</td>
</tr>
<tr>
<td></td>
<td>lod2Solid gml:SolidPropertyType 2</td>
</tr>
<tr>
<td></td>
<td>lod3Solid gml:SolidPropertyType 3</td>
</tr>
<tr>
<td></td>
<td>lod4Solid gml:SolidPropertyType 4</td>
</tr>
<tr>
<td></td>
<td>boundedBy wtr:BoundedByWaterSurface PropertyType 2–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfWaterBody xs:anyType 0–4</td>
</tr>
<tr>
<td>wtr:AbstractWaterBoundarySurfaceType</td>
<td>lod2Surface gml:SurfacePropertyType 2</td>
</tr>
<tr>
<td></td>
<td>lod3Surface gml:SurfacePropertyType 3</td>
</tr>
<tr>
<td></td>
<td>lod4Surface gml:SurfacePropertyType 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterBoundarySurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>wtr:WaterSurfaceType</td>
<td></td>
</tr>
<tr>
<td>waterLevel</td>
<td>WaterLevelType</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>wtr:WaterGroundSurfaceType</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterGroundSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>wtr:WaterClosureSurfaceType</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterClosureSurface</td>
<td>xs:anyType</td>
</tr>
</tbody>
</table>
Annex F
(informative)

Example CityGML datasets

F.1 Example of a CityGML dataset for a building in LOD1 and LOD2

Fig. 50: Visualisation of the following CityGML dataset containing buildings in LOD1 and LOD2 (source: IGG Uni Bonn).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
xmlns:tex="http://www.opengis.net/citygml/texturedsurface/1.0"
xmlns:grp="http://www.opengis.net/citygml/cityobjectgroup/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xAL="urn:oasis:names:tc:ciq:xsd:schema:xAL:2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/citygml/building/1.0 http://www.citygml.org/citygml/building/1.0/building.xsd
http://www.opengis.net/citygml/texturedsurface/1.0 http://www.citygml.org/citygml/texturedsurface/1.0/texturedSurface.xsd
http://www.opengis.net/citygml/cityobjectgroup/1.0 http://www.citygml.org/citygml/cityobjectgroup/1.0/cityObjectGroup.xsd">
<description>
Simple example for an XML dataset according to CityGML, the GML application schema of the SIG 3D. This dataset contains four parts with different complexities, which have been truncated here (the full version can be obtained from www.citygml.org):
1.) Simple building in LOD2 with one textured and one colored surface
2.) Simple building in LOD1 as blocks model without balcony, and the same building with gabled roof and balcony in LOD2.
3.) House with gabled roof and garage, represented by two BuildingParts. The common wall surface of the building and the garage is defined only once and is in the boundary of one solid, and re-used by the second solid.
4.) Building group consisting of two buildings that have been defined previously.
The coordinate reference system is given in DHDN / Gauss-Krueger 3 degree (2nd zone) + normal heights above sea level (DHHN92). This system is referred to by srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783". Please note that the coordinates actually used in this dataset have been trimmed for clarity reasons and thus do not match this CRS.  
</description>
<name>3D city model of Samplecity</name>
<boundedBy>
<pos srsDimension="3">0.0 0.0 0.0</pos>
<pos srsDimension="3">33.0 34.0 2.5</pos>
</Envelope>
</boundedBy>
<objectMember>
<bldg:Building gml:id="Build0815">
<externalReference>
<informationSystem>http://www.adv-online.de</informationSystem>
</externalReference>
</bldg:Building>
</objectMember>
</CityModel>
```
<gml:CompositeSurface>
  <gml:surfaceMember>
    <tex:TexturedSurface orientation="+">
      <!-- front surface -->
      <gml:baseSurface>
        <gml:Polygon>
          <gml:LinearRing>
            <gml:posList srsDimension="3">
              1.0 1.0 0.0 3.0 1.0 1.5 2.0 1.0 2.5 1.0 1.0 1.5
            </gml:posList>
          </gml:LinearRing>
        </gml:Polygon>
      </gml:baseSurface>
      <tex:appearance>
        <tex:material>
          <tex:ambientIntensity>0.4</tex:ambientIntensity>
        </tex:material>
        <tex:diffuseColor>0 0 1</tex:diffuseColor>
        <!-- defines blue color -->
        <tex:textureMap>FrontTexture096454.jpg</tex:textureMap>
        <tex:textureCoordinates>0.05 0.07 0.95 0.07 0.95 0.5 0.5 0.05 0.05 0.07</tex:textureCoordinates>
        <tex:textureType>specific</tex:textureType>
      </tex:appearance>
    </tex:TexturedSurface>
  </gml:surfaceMember>
  <gml:surfaceMember>
    <tex:TexturedSurface orientation="+">
      <!-- back surface -->
      <gml:baseSurface>
        <gml:Polygon>
          <gml:LinearRing>
            <gml:posList srsDimension="3">
              1.0 4.0 0.0 3.0 4.0 0.0 3.0 4.0 1.5 2.0 4.0 2.5 1.0 4.0 1.5
            </gml:posList>
          </gml:LinearRing>
        </gml:Polygon>
      </gml:baseSurface>
      <tex:appearance>
        <tex:material>
          <tex:ambientIntensity>0.4</tex:ambientIntensity>
        </tex:material>
        <tex:diffuseColor>0 0 1</tex:diffuseColor>
        <!-- defines blue color -->
        <tex:textureMap>FrontTexture096454.jpg</tex:textureMap>
        <tex:textureCoordinates>0.05 0.07 0.95 0.07 0.95 0.5 0.5 0.05 0.05 0.07</tex:textureCoordinates>
        <tex:textureType>specific</tex:textureType>
      </tex:appearance>
    </tex:TexturedSurface>
  </gml:surfaceMember>
</gml:CompositeSurface>

Both values are defined in external code lists.

The function is residential building (1000) and the roof type is 'gabled roof' (1030).

Both values are defined in external code lists.

...
The nice balcony to the south

The geometry of the balcony is defined by an aggregation of 3D surfaces.

1st roof surface. This polygon will be referenced below.

simple building with gabled roof

1st roof surface.
This may be a database, which contains all roof surfaces of a city covered with solar panels. The roof surface, shares surface with garage geometry, represented by two BuildingParts. The common wall surface of the building and the garage is shared by both solids realizing a topological connection between both parts.
<Address>
  <xAL:AddressDetails>
    <xAL:CountryName>Germany</xAL:CountryName>
    <xAL:LocalityType>Town</xAL:LocalityType>
  </xAL:AddressDetails>
</Address>
Listing 1: Excerpt from the CityGML dataset for buildings in LOD1 and 2 visualised in Fig. 50.
F.2 Example of a CityGML dataset for a building in LOD3

Fig. 51: Visualisation of buildings in LOD3, automatically generated from IFC building objects. Please note the coherent semantic and geometric decomposition (source: Research Center Karlsruhe).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/citygml/building/1.0 http://www.citygml.org/citygml/building/1.0/building.xsd">

<description>This file contains four buildings which are automatically converted from IFC models. This listing only shows an excerpt. The full dataset can be downloaded from http://www.citygml.org (example dataset for “four buildings in LOD3”).</description>

<name>IFC_Building_Variant</name>
<boundedBy>
<pos srsDimension="3">5429999.751795 3449999.751795 0.0</pos>
<pos srsDimension="3">5430023.2 3450021.2 20.0</pos>
</Envelope>
</boundedBy>

<cityObjectMember>
<bldg:Building gml:id="GEB_TH_IFC_Building_Variant_GEB_75">
<description>Building in LOD3</description>
<name>Building - ADT - 2006</name>
<externalReference>
<externalObject>
<uri>urn:ifc:oid:0deJpNQ05BvwV03c405oVp</uri>
</externalObject>
</externalReference>
<bldg:boundedBy>
<bldg:Envelope srsDimension="3">5429999.751795 3449999.751795 0.0</bldg:Envelope>
</bldg:boundedBy>
</bldg:Building>

<bldg:Building gml:id="GEB_TH_IFC_Building_Variant_GEB_75">
<description>Building in LOD3</description>
<name>Building - ADT - 2006</name>
<externalReference>
<externalObject>
<uri>urn:ifc:oid:0deJpNQ05BvwV03c405oVp</uri>
</externalObject>
</externalReference>
<bldg:boundedBy>
<bldg:Envelope srsDimension="3">5429999.751795 3449999.751795 0.0</bldg:Envelope>
</bldg:boundedBy>
</bldg:Building>

<bldg:Building gml:id="GEB_TH_IFC_Building_Variant_GEB_75">
<description>Building in LOD3</description>
<name>Building - ADT - 2006</name>
<externalReference>
<externalObject>
<uri>urn:ifc:oid:0deJpNQ05BvwV03c405oVp</uri>
</externalObject>
</externalReference>
<bldg:boundedBy>
<bldg:Envelope srsDimension="3">5429999.751795 3449999.751795 0.0</bldg:Envelope>
</bldg:boundedBy>
</bldg:Building>

<bldg:RoofSurface gml:id="GEB_TH_IFC_Building_Variant_DACH_136">
<externalObject>
<uri>urn:ifc:oid:3CPSkwS7f9QRfhfr5gf7dq</uri>
</externalObject>
</bldg:RoofSurface>
</cityObjectMember>
</CityModel>
```
Listing 2: Excerpt from the CityGML dataset for the buildings in LOD3 visualised in Fig. 51.
F.3 Example of a CityGML dataset illustrating the appearance model

The following CityGML dataset contains a simple building given in geometric representations for LOD1 and LOD2. Furthermore, two separate appearance themes are defined – a summer theme and a winter theme – describing different visual appearances for the building and the surrounding terrain. Each LOD has an individual appearance for these specific themes.

Several concepts of CityGML’s appearance model are used in this dataset. Regarding LOD1, an XI3Material object defines the material of the whole building which is applied to all of its surfaces. In addition, a GeoreferencedTexture is assigned both to the terrain and the roof surface of the building. In LOD2, the vertical surfaces of the building are texturised individually using ParameterizedTexture objects whereas the roof surfaces and the terrain again are described by a GeoreferencedTexture. The modelling approach results in four possible visualizations of the dataset that are represented in Fig. 52 and Fig. 53.

![CityGML dataset example](image)

Fig. 52: Visualisation of a simple building in LOD1 using CityGML’s appearance model. Two themes are defined for the building and the surrounding terrain: (a) theme showing the building in summer time and (b) showing the building in winter time (image: Hasso-Plattner-Institute).

![CityGML dataset example](image)

Fig. 53: Visualisation of a simple building in LOD2 using CityGML’s appearance model. Two themes are defined for the building and the surrounding terrain: (a) theme showing the building in summer time and (b) showing the building in winter time (image: Hasso-Plattner-Institute).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
  xmlns:app="http://www.opengis.net/citygml/appearance/1.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/1.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/1.0 http://www.citygml.org/citygml/building/1.0/building.xsd"
  xsi:schemaLocation="http://www.opengis.net/citygml/appearance/1.0 http://www.citygml.org/citygml/appearance/1.0/appearance.xsd"
  xsi:schemaLocation="http://www.opengis.net/citygml/appearance/1.0 http://www.citygml.org/citygml/appearance/1.0/appearance.xsd"
>`
Simple example for an XML dataset according to CityGML, the GML application schema of the SIG 3D. This dataset contains one simple building in LOD1 and LOD2 and the surrounding terrain as well as two separate appearance themes:
1.) Simple building in LOD1
2.) Simple building in LOD2
3.) Digital terrain given by a TIN.
4.) Appearance theme "summer".
5.) Appearance theme "winter".

Please note, that appearances are explicitly linked to GML geometry objects using URIs. Since Texture objects are modelled as features (with a unique id) they can be (and in fact are) reused. This is realized using XLinks. The coordinate reference system is given in DHDN / Gauss-Krueger 3 degree (2nd zone) + normal heights above sea level (DHHN92). This system is referred to by srsName="urn:ogc:def:crs,crs:EPSG:6.12:31466,crs:EPSG:6.12:5783". Please note, that the coordinates actually used in this dataset have been trimmed for clarity reasons and thus do not match this CRS.

<gml:boundedBy>
    <gml:lowerCorner>-6.0 -7.0 0.0</gml:lowerCorner>
    <gml:upperCorner>17.0 13.0 5.0</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="Build0815">
    <bldg:yearOfConstruction>2007</bldg:yearOfConstruction>
    <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
    <bldg:lod1Solid>
      <gml:Solid>
        <gml:exterior>
          <gml:CompositeSurface gml:id="lod1Surface">
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList srsDimension="3">0.0 0.0 0.0 10.0 0.0 0.0 10.0 0.0 4.0 0.0 0.0 4.0 0.0 0.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="lod1RoofPoly1">
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList srsDimension="3">0.0 0.0 4.0 10.0 0.0 4.0 10.0 5.0 4.0 0.0 5.0 4.0 0.0 0.0 4.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:surfaceMember>
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </bldg:lod1Solid>
    <bldg:lod2Solid>
      <gml:Solid>
        <gml:exterior>
          <gml:CompositeSurface gml:id="fLeft">
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing gml:id="fLeftExt1">
                    <gml:posList srsDimension="3">0.0 0.0 0.0 5.0 0.0 0.0 5.0 0.0 3.0 0.0 0.0 3.0 0.0 0.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:surfaceMember>
            <gml:surfaceMember>
              <gml:Polygon gml:id="fLeftExt2">
                <gml:exterior>
                  <gml:LinearRing gml:id="fLeftExt2">
                    <gml:posList srsDimension="3">5.0 0.0 0.0 10.0 0.0 0.0 10.0 0.0 3.0 5.0 0.0 3.0 5.0 0.0 0.0</gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:surfaceMember>
          </gml:CompositeSurface>
        </gml:exterior>
      </gml:Solid>
    </bldg:lod2Solid>
  </bldg:Building>
</cityObjectMember>
<gml:LinearRing>
  <gml:exterior>
    <gml:LinearRing id="fFrontExt">
      <gml:posList srsDimension="3">
        10.0 0.0 0.0 10.0 5.0 0.0 10.0 5.0 3.0 10.0 2.5 5.0 10.0 0.0 3.0 10.0 2.5 5.0 10.0 0.0 0.0</gml:posList>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>

<gml:Polygon id="fRight">
  <gml:exterior>
    <gml:LinearRing id="fRightExt">
      <gml:posList srsDimension="3">
        10.0 5.0 0.0 0.0 5.0 0.0 0.0 5.0 3.0 10.0 5.0 3.0 10.0 5.0 0.0</gml:posList>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>

<gml:Polygon id="fBack">
  <gml:exterior>
    <gml:LinearRing id="fBackExt">
      <gml:posList srsDimension="3">
        0.0 5.0 0.0 0.0 0.0 5.0 0.0 0.0 3.0 2.5 5.0 0.0 5.0 3.0 0.0</gml:posList>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>

<gml:Polygon id="lod2RoofPoly1">
  <gml:exterior>
    <gml:LinearRing>
      <gml:posList srsDimension="3">
        0.0 0.0 3.0 10.0 0.0 3.0 10.0 2.5 5.0 0.0 2.5 5.0 0.0 3.0 0.0</gml:posList>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>

<gml:Polygon id="lod2RoofPoly2">
  <gml:exterior>
    <gml:LinearRing>
      <gml:posList srsDimension="3">
        10.0 5.0 3.0 0.0 5.0 3.0 0.0 2.5 5.0 10.0 2.5 5.0 10.0 5.0 3.0</gml:posList>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>
Listing 3: Excerpt from the CityGML dataset illustrating CityGML’s appearance model. The dataset is visualised in Fig. 52 and Fig. 53.

The following three raster images (Fig. 54 - Fig. 56) are referenced in the dataset by ParameterizedTexture objects to texturize the vertical boundary surfaces of the building in LOD2. The image facade.png (cf. Fig. 54) is assigned to the side surfaces using the texture wrapping mode wrap and is applied both within the summer and the winter theme.

![fig:facade.png](image)

Fig. 54: Image facade.png used in the dataset to texturize the side surfaces of the building in LOD2 (cf. Fig. 53 a. and b.) (image: Hasso-Plattner-Institute).

Fig. 55 shows the texture atlas front_back_summer.png combining the textures for the front surface and the back surface of the building in LOD2 within the summer theme. Only a portion of this image is assigned to the specific surfaces. The relevant parts are defined using a TextCoordList object.

![fig:front_back_summer.png](image)

Fig. 55: Texture atlas front_back_summer.png containing the textures for the front surface and the back surface of the building in LOD2 within the summer theme (cf. Fig. 53 a.) (image: Hasso-Plattner-Institute).

Identically to front_back_summer.png the texture atlas front_back_winter.png contains the textures for the front surface and the back surface of the building in LOD2 within the winter theme.

![fig:front_back_winter.png](image)
The raster images shown in Fig. 57 and Fig. 58 are assigned to the terrain and the roof surfaces of the building in LOD1 as well as in LOD2. In the dataset this is implemented by a GeoreferencedTexture object linking to the according GML geometry objects. Whereas the image ground_summer.png (cf. Fig. 57) represents the texture for the summer theme, ground_winter.png (cf. Fig. 58) is used within the winter theme.
F.4 Example of a CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes

The following example demonstrates the use of texture coordinates for complex surfaces with holes. Additionally, the concept of overwriting (cf. chapter 9.1) is exemplified. The example shows a textured road in LOD1, which consists of a roundabout and two entering streets (Fig. 59). It uses two app:ParameterizedTexture objects, a regular road piece (rd, Fig. 60) and a dirt track becoming a paved road (dt, Fig. 61). All geometry is contained in a single gml:MultiSurface (road) being the target of rd. The roundabout is modelled as a polygon with holes (roundaboutPoly). Both rings raEx and raIn need to receive texture coordinates. For efficient texturing, texture coordinates and wrap mode are chosen such that the regular road piece is draped onto the polygonal ring as desired at the price of some distortion. A distortion-free alternative, e.g. for higher LODs, requires an additional road piece texture for the roundabout (Fig. 62) or a texture for the complete roundabout (Fig. 63). In both cases, texture space is wasted. Wasted areas are marked red.

Fig. 59: A rendering of the textured geometry (images for this example: Hasso-Plattner-Institute).

Fig. 60: The road piece texture.

Fig. 61: The dirt track texture.

Fig. 62: A distortion-free road piece for the roundabout. Red areas never become visible and are wasted.
The complete roundabout in a distortion-free texture.

Red areas never become visible and are wasted.

The entering dirt track requires a different texture. Even though its geometry (dirtPoly) is contained in the already textured road, the existing texture is overwritten and replaced by using dirtPoly as target of dt. Overwriting only occurs when assigning a new texture to a surface geometry object contained in an already textured aggregated geometry object. Assigning two textures to the same surface geometry object directly is not allowed. The same applies to materials.

```xml
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
  xmlns:app="http://www.opengis.net/citygml/appearance/1.0"
  xmlns:tran="http://www.opengis.net/citygml/transportation/1.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xsi:schemaLocation="http://www.opengis.net/citygml/appearance/1.0 http://www.citygml.org/citygml/appearance/1.0/appearance.xsd"
  xsi:schemaLocation="http://www.opengis.net/citygml/transportation/1.0 http://www.citygml.org/citygml/transportation/1.0/transportation.xsd">
  <gml:boundedBy>
    <gml:Envelope>
      <gml:lowerCorner>-45.0 -20.0 0.0</gml:lowerCorner>
      <gml:upperCorner>45.0 20.0 10.0</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <tran:Road>
      <app:appearance>
        <app:Appearance>
          <app:theme>visual</app:theme>
          <app:surfaceDataMember>
            <app:ParameterizedTexture gml:id="rd">
              <app:imageURI>rd.png</app:imageURI>
              <app:wrapMode>mirror</app:wrapMode>
              <app:target uri="#road">
                <app:TexCoordList>
                  <app:textureCoordinates ring="#raEx">0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1</app:textureCoordinates>
                  <app:textureCoordinates ring="#raIn">0 1 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 0 0</app:textureCoordinates>
                  <app:textureCoordinates ring="#roadEx">0 2 5 0 2 5 0 1 0 1 0 0</app:textureCoordinates>
                  <app:textureCoordinates ring="#dirtEx">0 2 5 0 2 5 0 1 0 1 0 0</app:textureCoordinates>
                </app:TexCoordList>
              </app:target>
            </app:ParameterizedTexture>
          </app:surfaceDataMember>
          <app:surfaceDataMember>
            <app:ParameterizedTexture gml:id="dt">
              <app:imageURI>dt.png</app:imageURI>
              <app:wrapMode>mirror</app:wrapMode>
              <app:target uri="#dirtPoly">
                <app:TexCoordList>
                  <app:textureCoordinates ring="#dirtEx">0 0 1 0 1 0 0 1 0 1 0 0</app:textureCoordinates>
                </app:TexCoordList>
              </app:target>
            </app:ParameterizedTexture>
          </app:surfaceDataMember>
        </app:Appearance>
        <app:lod1MultiSurface>
          <gml:MultiSurface gml:id="road">
            <gml:surfaceMember>
              <gml:Polygon gml:id="roundaboutPoly">
                <gml:exterior>
                </gml:exterior>
              </gml:Polygon>
            </gml:surfaceMember>
          </gml:MultiSurface>
        </app:lod1MultiSurface>
      </app:appearance>
    </tran:Road>
  </cityObjectMember>
</CityModel>
```
Listing 4: CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes. The dataset is visualised in Fig. 59.
Annex G
(informative)

Example ADE for Noise Immission Simulation

This annex illustrates the usage of CityGML within an environmental simulation application. The definition of the corresponding Application Domain Extension (ADE) is included as an example.

The Environmental Noise Directive of the European Union 2002/49/EG obligates the EU member states to calculate every 5 years the noise levels at a height of 4m on buildings and to document the results in noise maps. The noise maps serve as information for the European Union and the citizens affected by noise (Fig. 64). These noise maps are generated on the basis of acoustic models and noise propagation calculations, not on the basis of measurements (Fig. 65). For the noise propagation calculations, a great number of thematic data and 3D geodata is necessary for each EU member state. Because of the large spatial extent of the noise calculation, the provision of statewide and ubiquitous 3D geodata on buildings, roads, railways and terrain for a multitude of users is necessary, in part with high requirements on resolution.

The calculation of noise levels from a road requires information about the traffic flow, the heavy vehicle percentage, the speed limits, the road surface types and the road gradient. Furthermore, the noise level depends on the distance between point of emission and reception (immission) as well as on reflection (e.g. on building facades) or shielding effects (e.g. noise barriers). The noise level is calculated separately for the day (06.00-18.00), the evening (18.00-22.00) and the night (22.00-06.00). As the noise level is calculated at a height of 4m and as the influence of vertical reflecting surfaces is considered (e.g. noise barriers and buildings), a multitude of geodata in the third dimension is necessary. In addition to all 3D geodata specific thematic data are required. For example the following data are necessary for the noise calculation of roads: Digital Terrain Model with 10m grid, 3D building models with their thematic attributes (e.g. reflection, inhabitants), 3D road data with their thematic attributes (e.g. traffic flow, heavy vehicle percentage, speed limit, type of road surface, road gradient, width of a road), 3D noise barriers and their thematic attributes (e.g. reflection).

Fig. 64: Noise map generated from the 3D CityGML geodata and used for the reporting demanded of the EU Environmental Noise Directive (dark colours show higher noise immission) (source: State Agency for nature, environment and consumer protection NRW).
In the state of North Rhine-Westphalia, special conditions have to be considered: high population and transportation route density and therefore the highest amount of noise calculation areas and objects in Germany. The aim is to provide a sustainable, efficient and variable access to the required 3D geodata for the 5 years iteration period and the different noise calculation authorities.

In order to provide this considerable amount of statewide 3D geodata, the responsible partners, in particular the State Ministry of Environment, Nature Conservation, Agriculture and Consumer Protection of North Rhine-Westphalia, the State Agency for nature, environment and consumer protection of North Rhine-Westphalia and the Surveying and Mapping Agency of North Rhine-Westphalia, have decided to use the Spatial Data Infrastructure in North Rhine-Westphalia (GDI NRW) and to extend it with statewide Web Services for 2.5D and 3D geoinformation. Therefore, new OGC Web Services for building models, terrain, road and railway data were implemented (e.g. Web Feature Service for 3D block models in LOD1 and for 3D road and railway data, Web Coverage Service for DTM).
CityGML is used together with GeoTIFF as the only exchange formats between web services and noise calculation software (Fig. 66 - Fig. 68). For the special requirements of the noise directive, a CityGML noise application schema has been developed by the Institute of Geodesy and Geoinformation University of Bonn and the Special Interest Group SIG 3D of GDI NRW. It is based on the ADE mechanism (see chapter 6.12 and 10.11). This mechanism allows the supplementation of existing classes and objects in CityGML (e.g. buildings) by thematic attributes. The quantity as well as the type of these attributes is selectable. The CityGML schema can also be complemented by new classes. Hence, the noise application schema contains new objects (e.g. segmentation of roads according to noise requirements - NoiseRoadSegment, Fig. 69) as well as noise attributes attached to existing objects (e.g. reflection of buildings, Fig. 70). These additional noise attributes are derived from regulations issued by the Federal Government of Germany realising the obligations of the Environmental Noise Directive of the European Union (cf. BImSchV 2006, VBUS 2006, VBUSch 2006).

The interoperability techniques of this project demonstrate a remarkable innovation, as for the first time statewide 3D geodata are provided via common standards and web services. Therefore, the Spatial Data Infrastructure for noise calculation in North Rhine-Westphalia provides an application example for the INSPIRE directive of the European Union 2007/2/EC (Infrastructure for Spatial Information in Europe).

Fig. 67: 3D geodata in CityGML for the calculation of the noise map in Fig. 64: Derived contour lines for the generation of CityGML breaklines, 3D block model in CityGML, 3D road and railway data in CityGML, state road data for higher-level roads in CityGML (source: Surveying and Mapping Agency NRW, State Road Enterprise NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

Fig. 68: Extract from Fig. 67 shows the integration of 3D block models in the DTM by appropriate CityGML modelling (lowest point of ALK building polygon is taken as measure to generate the building bottom side) (source: Surveying and Mapping Agency NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).
G.1 CityGML Noise ADE

In this section the data models for the CityGML Noise ADE are given as UML diagrams and XML schema. As the semantics of the specific attributes and object types result from the German regulations for noise immersion computations, they are not explained in detail here (see BimSchV 2006, VBUS 2006, VBUSch 2006). The purpose of this section is to provide an example how CityGML can be extended using the ADE mechanism.

![CityGML noise application schema – transportation model](image)

Fig. 69: CityGML noise application schema – transportation model (light yellow=CityGML Transportation module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Transportation module. The prefix noise is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).
Fig. 70: CityGML noise application schema – building model (light yellow=CityGML Building module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Building module. The prefix noise is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).
Header of the Noise ADE Schema definition file

```
<xs:import namespace="http://www.opengis.net/gml" schemaLocation="/3.1.1/base/gml.xsd"/>
<xs:import namespace="http://www.opengis.net/citygml/cityfurniture/1.0" schemaLocation="/citygml/cityfurniture/1.0/cityfurniture.xsd"/>
<xs:import namespace="http://www.opengis.net/citygml/building/1.0" schemaLocation="/citygml/building/1.0/building.xsd"/>
<xs:import namespace="http://www.opengis.net/citygml/transportation/1.0" schemaLocation="/citygml/transportation/1.0/transportation.xsd"/>
<xs:import namespace="http://www.citygml.org/ade/noise_de" schemaLocation="/citygml/ade/noise_de/noise_de.xsd"/>
```

**NoiseCityFurnitureSegmentTypeType, NoiseCityFurnitureSegment**

```
<xs:element name="NoiseCityFurnitureSegmentProperty type="NoiseCityFurnitureSegmentPropertyType"

substitutionGroups="fn_GenericApplicationPropertyOfCityFurniture"/>
```

---

Fig. 71: CityGML noise application schema – city furniture model (light yellow=CityGML CityFurniture module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML CityFurniture module. The prefix *noise* is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).
Application specific attributes for _AbstractBuilding

- buildingReflection
- buildingReflectionCorrection
- buildingLDenEq
- buildingLDenMin
- buildingLDenMax
- buildingLDenMin
- buildingLNightEq
- buildingLNightMin
- buildingLNightMax
- buildingReflection
- brakePortionDay
- brakePortionEvening
- brakePortionNight
- lengthDay
- lengthDay
- lengthEvening
- lengthEvening
- lengthNight
- lengthNight
- speedDay
- speedDay
- speedEvening
- speedEvening
- speedNight
- speedNight
- additionalCorrectionTrain
- trainTypeCorrection
- trainType
G.2 Example dataset

The following dataset illustrates a CityGML instance document which uses the application noise schema. It contains two CityObject features: a road object and a building object. The dataset references the XML Schema definition file of the CityGML Noise ADE which explicitly imports the XML Schema definitions of the CityGML modules extended by the Noise ADE (CityGML Core, Building, Transportation, and CityFurniture module). Thus, all classes defined by the employed CityGML modules can be used in the instance document. Furthermore, the application specific additions such as new object types (e.g. NoiseRoadSegment) and additional thematic attributes (e.g. the attributes defined for _AbstractBuilding) are available. These additional elements are distinguished from standard CityGML elements by the namespace prefix noise which refers to the noise schema definition.

<?xml version="1.0" encoding="ISO-8859-1"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0/
xmlns:trans="http://www.opengis.net/citygml/transportation/1.0"
xmlns:bldg="http://www.opengis.net/citygml/building/1.0"
xmlns:noise="http://www.opengis.net/ade/noise_de"
xsi:schemaLocation="http://www.citygml.org/ade/noise_de http://www.citygml.org/ade/noise_de/CityGML-NoiseADE-0-5-0.xsd">
  <gml:boundedBy>
      <gml:pos srsDimension="3">5616000.0 2540097.5 54.5</gml:pos>
      <gml:pos srsDimension="3">5673522.3 2576495.6 172.9</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <trans:Road gml:id="CR_0815">
      <gml:name>B1</gml:name>
        <gml:pos srsDimension="3">5616000.0 2573988.4 158.0</gml:pos>
        <gml:pos srsDimension="3">5671805.5 2574049.8 158.2</gml:pos>
      </gml:Envelopes>
      <gml:boundedBy>
        <gml:Envelope>
          <gml:pos srsDimension="3">5616000.0 2573988.4 158.0</gml:pos>
          <gml:pos srsDimension="3">5671805.5 2574049.8 158.2</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <trans:function>B1303</trans:function>
    </trans:Road>
    <noise:NoiseRoadSegment gml:id="CNRS_0815">
      <gml:boundedBy>
          <gml:pos srsDimension="3">5616868.0 2573988.4 158.0</gml:pos>
          <gml:pos srsDimension="3">5671805.5 2574049.8 158.2</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <noise:streetCenterline>
        <noise:streetCenterlineDimension srsDimension="3">5616868.0 2573988.4 158.0</noise:streetCenterlineDimension>
      </noise:streetCenterline>
    </noise:NoiseRoadSegment>
  </cityObjectMember>
</CityModel>
Listing 5: Excerpt from a CityGML dataset implementing the illustrated CityGML noise application schema.
Bibliography


