



BSI Standards Publication

Geographic information – Geography Markup Language (GML)

Part 1: Fundamentals

National foreword

This British Standard is the UK implementation of EN ISO 19136-1:2020. It is identical to ISO 19136-1:2020. Together with BS EN ISO 19136-2:2018, it supersedes BS EN ISO 19136:2009, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/36, Geographic information.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2020
Published by BSI Standards Limited 2020

ISBN 978 0 539 00597 4

ICS 35.240.70

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2020.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

English Version

Geographic information - Geography Markup Language (GML) - Part 1: Fundamentals (ISO 19136-1:2020)

Information géographique - Langage de balisage en
géographie (GML) - Partie 1: Principes de base (ISO
19136-1:2020)

Geoinformation - Geography Markup Language (GML) -
Teil 1 (ISO 19136-1:2020)

This European Standard was approved by CEN on 14 April 2020.

CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration. Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CEN member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

This document (EN ISO 19136-1:2020) has been prepared by Technical Committee ISO/TC 211 "Geographic information/Geomatics" in collaboration with Technical Committee CEN/TC 287 "Geographic Information" the secretariat of which is held by BSI.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by October 2020, and conflicting national standards shall be withdrawn at the latest by October 2020.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN shall not be held responsible for identifying any or all such patent rights.

This document supersedes EN ISO 19136:2009.

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

Endorsement notice

The text of ISO 19136-1:2020 has been approved by CEN as EN ISO 19136-1:2020 without any modification.

Contents

	Page
Foreword	x
Introduction	xii
1 Scope	1
2 Normative references	1
3 Terms, definitions, symbols and abbreviated terms	2
3.1 Terms and definitions	2
3.2 Abbreviated terms	9
4 Conformance	10
4.1 Conformance requirements	10
4.2 Conformance classes related to GML application schemas	10
4.3 Conformance classes related to GML profiles	11
4.4 Conformance classes related to GML documents	12
4.5 Conformance classes related to software implementations	12
5 Conventions	13
5.1 XML namespaces	13
5.2 Versioning	13
5.3 Deprecated parts of previous versions of GML	13
5.4 UML notation	13
5.5 XML Schema	15
6 Overview of the GML schema	15
6.1 GML schema	15
6.2 GML application schemas	15
6.3 Relationship between the ISO 19100 series of International Standards, the GML schema and GML application schemas	16
6.4 Organization of this document	17
6.5 Deprecated and experimental schema components	18
7 GML schema — General rules and base schema components	19
7.1 GML model and syntax	19
7.1.1 GML instance documents	19
7.1.2 Lexical conventions	20
7.1.3 XML Schema definition of GML language	20
7.2 gmlBase schema components	21
7.2.1 Goals of base schema components	21
7.2.2 Base objects	21
7.2.3 GML properties	22
7.2.4 Standard properties of GML objects	27
7.2.5 Collections of GML objects	27
7.2.6 Metadata	28
8 GML schema — Xlinks and basic types	31
8.1 Xlinks — Object associations and remote properties	31
8.2 Basic types	33
8.2.1 Overview	33
8.2.2 Relationship with ISO 19103	33
8.2.3 Simple types	33
8.2.4 Lists	38
9 GML schema — Features	40
9.1 General concepts	40
9.2 Relationship with ISO 19109	40
9.3 Features	41
9.3.1 AbstractFeatureType	41
9.3.2 AbstractFeature	41

9.4	Standard feature properties	41
9.4.1	boundedBy, BoundingShapeType, EnvelopeWithTimePeriod, EnvelopeWithTimePeriodType	41
9.4.2	locationName, locationReference	42
9.4.3	FeaturePropertyType, FeatureArrayPropertyType	43
9.5	Geometry properties	43
9.6	Topology properties	44
9.7	Temporal properties	45
9.8	Defining application-specific feature types	46
9.9	Feature collections	47
9.9.1	GML feature collections	47
9.9.2	AbstractFeatureMemberType and derived property types	48
9.10	Spatial reference system used in a feature or feature collection	48
10	GML schema — Geometric primitives	49
10.1	General concepts	49
10.1.1	Overview	49
10.1.2	Relationship with ISO 19107	49
10.1.3	Abstract geometry	50
10.1.4	Coordinate geometry, vectors and envelopes	52
10.2	Abstract geometric primitives	54
10.2.1	AbstractGeometricPrimitiveType, AbstractGeometricPrimitive	54
10.2.2	GeometricPrimitivePropertyType	55
10.3	Geometric primitives (0-dimensional)	55
10.3.1	PointType, Point	55
10.3.2	PointPropertyType, pointProperty	55
10.3.3	PointArrayPropertyType, pointArrayProperty	56
10.4	Geometric primitives (1-dimensional)	56
10.4.1	AbstractCurveType, AbstractCurve	56
10.4.2	CurvePropertyType, curveProperty	56
10.4.3	CurveArrayPropertyType, curveArrayProperty	57
10.4.4	LineStringType, LineString	57
10.4.5	CurveType, Curve	57
10.4.6	OrientableCurveType, OrientableCurve, baseCurve	58
10.4.7	Curve segments	58
10.5	Geometric primitives (2-dimensional)	67
10.5.1	AbstractSurfaceType, AbstractSurface	67
10.5.2	SurfacePropertyType, surfaceProperty	68
10.5.3	SurfaceArrayPropertyType, surfaceArrayProperty	68
10.5.4	PolygonType, Polygon	68
10.5.5	exterior, interior	68
10.5.6	AbstractRingType, AbstractRing	69
10.5.7	AbstractRingPropertyType	69
10.5.8	LinearRingType, LinearRing	69
10.5.9	LinearRingPropertyType	70
10.5.10	SurfaceType, Surface	70
10.5.11	OrientableSurfaceType, OrientableSurface, baseSurface	70
10.5.12	Surface patches	72
10.6	Geometric primitives (3-dimensional)	76
10.6.1	AbstractSolidType, AbstractSolid	76
10.6.2	SolidPropertyType, solidProperty	76
10.6.3	SolidArrayPropertyType, solidArrayProperty	77
10.6.4	SolidType, Solid	77
10.6.5	ShellType, Shell	77
10.6.6	ShellPropertyType	78
11	GML schema — Geometric complex, geometric composites and geometric aggregates	78
11.1	Overview	78
11.2	Geometric complex and geometric composites	79

11.2.1	Geometric complex	79
11.2.2	Composite geometries	79
11.3	Geometric aggregates	81
11.3.1	Aggregates of unspecified dimensionality	81
11.3.2	0-Dimensional aggregates	82
11.3.3	1-Dimensional aggregates	83
11.3.4	2-Dimensional aggregates	84
11.3.5	3-Dimensional aggregates	85
12	GML schema — Coordinate reference systems schemas	85
12.1	Overview	85
12.1.1	General	85
12.1.2	Relationship with ISO 19111	86
12.1.3	Important XML elements	86
12.2	Reference systems	87
12.2.1	Overview	87
12.2.2	IdentifiedObjectType	88
12.2.3	Abstract coordinate reference system	88
12.3	Coordinate reference systems	89
12.3.1	Overview	89
12.3.2	Abstract coordinate reference systems	89
12.3.3	Concrete coordinate reference systems	90
12.4	Coordinate systems	96
12.4.1	Overview	96
12.4.2	Coordinate system axes	96
12.4.3	Abstract coordinate system	97
12.4.4	Concrete coordinate systems	98
12.5	Datums	103
12.5.1	Overview	103
12.5.2	Abstract datum	103
12.5.3	Geodetic datum	104
12.5.4	Other concrete datums	106
12.6	Coordinate operations	108
12.6.1	Overview	108
12.6.2	Abstract coordinate operations	109
12.6.3	Concrete coordinate operations	112
12.6.4	Parameter values and groups	115
12.6.5	Operation method	117
12.6.6	Operation parameters and groups	118
13	GML schema — Topology	120
13.1	General concepts	120
13.1.1	Overview	120
13.1.2	Relationship with ISO 19107	120
13.2	Abstract topology	121
13.3	Topological primitives	121
13.3.1	Abstract topological primitives	121
13.3.2	Topological primitives (0-dimensional)	121
13.3.3	Topological primitives (1-dimensional)	122
13.3.4	Topological primitives (2-dimensional)	123
13.3.5	Topological primitives (3-dimensional)	124
13.4	Topological collections	125
13.4.1	Topological collection (0-dimensional)	125
13.4.2	Topological collection (1-dimensional)	126
13.4.3	Topological collection (2-dimensional)	126
13.4.4	Topological collection (3-dimensional)	127
13.5	Topology complex	127
13.5.1	TopoComplexType, TopoComplex	127
13.5.2	Maximal, sub- and super-complexes	128

13.5.3	topoPrimitiveMember	128
13.5.4	topoPrimitiveMembers	128
13.5.5	TopoComplexPropertyType, topoComplexProperty	128
14	GML schema — Temporal information and dynamic features	129
14.1	General concepts	129
14.1.1	Overview	129
14.1.2	Relationship with ISO 19108	130
14.2	Temporal schema	130
14.2.1	Abstract temporal objects	130
14.2.2	Temporal geometry	132
14.3	Temporal topology schema	137
14.3.1	General	137
14.3.2	Temporal topology objects	137
14.4	Temporal reference systems	140
14.4.1	Overview	140
14.4.2	Basic temporal reference system, TimeReferenceSystem	140
14.4.3	TimeCoordinateSystem	141
14.4.4	Calendars and clocks	142
14.4.5	Ordinal temporal reference systems	144
14.5	Representing dynamic features	146
14.5.1	Overview	146
14.5.2	dataSource	146
14.5.3	Dynamic properties	147
14.5.4	DynamicFeature	147
14.5.5	DynamicFeatureCollection	147
14.5.6	AbstractTimeSlice	148
14.5.7	history	149
15	GML schema — Definitions and dictionaries	150
15.1	Overview	150
15.2	Dictionary schema	151
15.2.1	Definition, DefinitionType, remarks	151
15.2.2	Dictionary, DictionaryType	151
15.2.3	dictionaryEntry, DictionaryEntryType	152
15.2.4	Using definitions and dictionaries	152
16	GML schema — Units, measures and values	153
16.1	Introduction	153
16.2	Units schema	154
16.2.1	Overview	154
16.2.2	Using unit definitions	154
16.2.3	unitOfMeasure, UnitOfMeasureType	154
16.2.4	UnitDefinition, UnitDefinitionType	155
16.2.5	quantityType, quantityTypeReference	155
16.2.6	catalogSymbol	155
16.2.7	BaseUnit, BaseUnitType, unitsSystem	155
16.2.8	DerivedUnit, DerivedUnitType	156
16.2.9	derivationUnitTerms, DerivationUnitTermType	156
16.2.10	ConventionalUnit, ConventionalUnitType	156
16.2.11	conversionToPreferredUnit, roughConversionToPreferredUnit, ConversionToPreferredUnitType, FormulaType	157
16.2.12	Example of units dictionary <informative>	158
16.3	Measures schema	159
16.3.1	Overview	159
16.3.2	measure	159
16.3.3	Scalar measure types	159
16.3.4	angle	160
16.4	Value objects schema	160
16.4.1	Introduction	160

16.4.2	Value element hierarchy	160
16.4.3	Boolean, BooleanList	161
16.4.4	Category, CategoryList	161
16.4.5	Count, CountList	162
16.4.6	Quantity, QuantityList	162
16.4.7	AbstractValue, AbstractScalarValue, AbstractScalarValueList	163
16.4.8	Value	163
16.4.9	valueProperty, valueComponent, valueComponents	163
16.4.10	CompositeValue	164
16.4.11	ValueArray	165
16.4.12	Typed ValueExtents: CategoryExtent, CountExtent, QuantityExtent	166
16.4.13	BooleanPropertyType, CategoryPropertyType, CountPropertyType, QuantityPropertyType	167
17	GML schema — Directions	167
17.1	Direction schema	167
17.2	direction, DirectionPropertyType	167
17.3	DirectionVectorType	167
17.4	DirectionDescriptionType	168
18	GML schema — Observations	169
18.1	Observations	169
18.2	Observation schema	169
18.2.1	Overview	169
18.2.2	Observation	169
18.2.3	using	170
18.2.4	target	170
18.2.5	resultOf	171
18.2.6	DirectedObservation	171
18.2.7	DirectedObservationAtDistance	172
19	GML schema — Coverages	173
19.1	The coverage model and representations	173
19.1.1	General remarks	173
19.1.2	Formal description of a coverage	174
19.1.3	Coverage in GML	174
19.1.4	Relationship with ISO 19123	175
19.2	Grids schema	175
19.2.1	Overview	175
19.2.2	Grid	175
19.2.3	RectifiedGrid	176
19.3	Coverage schema	178
19.3.1	AbstractCoverageType, AbstractCoverage	178
19.3.2	DiscreteCoverageType, AbstractDiscreteCoverage	178
19.3.3	AbstractContinuousCoverageType, AbstractContinuousCoverage	178
19.3.4	domainSet, DomainSetType	179
19.3.5	rangeSet, RangeSetType	179
19.3.6	DataBlock	180
19.3.7	rangeParameters	180
19.3.8	tupleList	180
19.3.9	doubleOrNilReasonTupleList	181
19.3.10	File, FileType	181
19.3.11	coverageFunction, CoverageFunctionType	182
19.3.12	CoverageMappingRule	183
19.3.13	GridFunction, GridFunctionType	184
19.3.14	sequenceRule, SequenceRuleType, SequenceRuleEnumeration	184
19.3.15	Specific Coverage Types in GML	185
19.3.16	MultiPointCoverage	185
19.3.17	MultiCurveCoverage	186
19.3.18	MultiSurfaceCoverage	186

	19.3.19 MultiSolidCoverage.....	189
	19.3.20 GridCoverage.....	189
	19.3.21 RectifiedGridCoverage.....	190
20	Profiles.....	191
20.1	Profiles of GML and application schemas.....	191
20.2	Definition of profile.....	191
20.3	Relation to application schema.....	191
20.4	Rules for elements and types in a profile.....	192
20.5	Rules for referencing GML profiles from application schemas.....	192
20.6	Recommendations for application schemas using GML profiles.....	193
20.7	Summary of rules for GML profiles.....	193
21	Rules for GML application schemas.....	194
21.1	Instances of GML objects.....	194
21.1.1	GML documents.....	194
21.1.2	GML object elements in other XML documents.....	194
21.2	GML application schemas.....	194
21.2.1	General.....	194
21.2.2	Target namespace.....	196
21.2.3	Import GML schema.....	196
21.2.4	Object type derivation.....	196
21.2.5	Elements representing objects.....	196
21.2.6	Property type derivation.....	196
21.2.7	Elements representing properties.....	197
21.3	Schemas defining Features and Feature Collections.....	197
21.3.1	General.....	197
21.3.2	Import GML schema components.....	197
21.3.3	Elements representing features.....	198
21.3.4	Application features are features.....	198
21.4	Schemas defining spatial geometries.....	198
21.4.1	Import GML geometry schema components.....	198
21.4.2	User-defined geometry types and geometry property types.....	198
21.5	Schemas defining spatial topologies.....	199
21.5.1	Import GML topology schema components.....	199
21.5.2	User-defined topology types and topology property types.....	200
21.6	Schemas defining time.....	200
21.6.1	Import GML temporal schema components.....	200
21.6.2	User-defined temporal types and temporal property types.....	200
21.7	Schemas defining coordinate reference systems.....	201
21.7.1	General.....	201
21.7.2	Import GML coordinate reference system schema components.....	202
21.8	Schemas defining coverages.....	202
21.8.1	General.....	202
21.8.2	Import GML coverage schema components.....	202
21.8.3	User-defined coverage types.....	202
21.8.4	Range parameters shall be substitutable for AbstractValue.....	202
21.8.5	Coverage document.....	203
21.9	Schemas defining observations.....	203
21.9.1	General.....	203
21.9.2	Import GML observation schema components.....	203
21.9.3	User-defined observation types.....	204
21.9.4	Observation collections.....	204
21.9.5	Observations are features.....	204
21.9.6	Observation collection document.....	204
21.10	Schemas defining dictionaries and definitions.....	204
21.10.1	General.....	204
21.10.2	Import GML dictionary schema components.....	204
21.10.3	User-defined definition types.....	204

21.10.4	User-defined dictionary types.....	205
21.11	Schemas defining values.....	205
21.11.1	General.....	205
21.11.2	Import GML value objects schema components.....	205
21.11.3	Construction of new value types.....	205
21.12	GML profiles of the GML schema.....	205
Annex A	(normative) Abstract test suites for GML application schemas, GML profiles and GML documents	208
Annex B	(normative) Abstract test suite for software implementations.....	222
Annex C	(informative) GML schema.....	226
Annex D	(normative) Implemented profile of the ISO 19100 series of International Standards and extensions.....	228
Annex E	(normative) UML-to-GML application schema encoding rules.....	289
Annex F	(normative) GML-to-UML application schema encoding rules.....	308
Annex G	(informative) Guidelines for subsetting the GML schema.....	317
Annex H	(informative) Default styling.....	329
Annex I	(informative) Backwards compatibility with earlier versions of GML.....	339
Annex J	(informative) Modularization and dependencies.....	355
Bibliography		357

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*.

This first edition of ISO 19136-1 cancels and replaces ISO 19136:2007 which has been technically revised.

The main changes compared to the previous edition are as follows:

- The Geography Markup Language (GML) was originally developed within the Open Geospatial Consortium, Inc. (OGC). ISO 19136 was prepared by ISO/TC 211 jointly with the OGC. This edition of this document is a revision to GML 3.2.1 (ISO 19136:2007). It addresses the OGC Change Request 12-092 (gml:id attribute on LinearRing) by applying the following changes:
 - the XML attribute gml:id in gml:AbstractGMLType has been made optional;
 - the elements gml:AbstractRing and gml:Shell have been added to the substitutionGroups gml:AbstractCurve and gml:AbstractSurface respectively;
 - the types gml:AbstractRingType and gml:ShellType are now extended from base types gml:AbstractCurveType and gml:AbstractSurfaceType respectively;

These changes correct inconsistencies with ISO 19107 without breaking the validity of instance documents created using the GML 3.2.1 schema. i.e. all GML 3.2 instance documents that are valid against the GML 3.2.1 schema are also valid against the GML 3.2.2 schema.

The corrected GML 3.2 schema is available at <http://schemas.opengis.net/gml/3.2.1/>. Note that the use of “3.2.1” in the URL is unchanged since this version (3.2.2) replaces the GML 3.2.1 schema. Previous versions of the GML 3.2.1 schema are available at http://schemas.opengis.net/gml/gml-3_2_1.zip.

The change to the gml:id attribute reverts a change that has been made between GML 3.1.1 and GML 3.2.1. Reverting this change also addresses comments raised by several communities since the release of GML 3.2.1 / ISO 19136:2007.

As the correction relaxes a constraint in the XML schema, not all instance documents created based on the GML 3.2.2 schema will be valid against the GML 3.2.1 schema:

- all GML 3.2 instance documents that include a gml:id attribute on a ring or shell element are not valid against the GML 3.2.1 schema;
- all GML 3.2 instance documents that include a feature, a spatial object or a temporal object without a gml:id attribute are not valid against the GML 3.2.1 schema.

Local copies of the GML 3.2.1 schema documents have to be replaced by the GML 3.2.2 schema documents – or be replaced by links to <http://schemas.opengis.net/gml/3.2.1/gml.xsd>.

- URIs have been updated, mainly in examples, where OGC policies have changed since the release of GML 3.2.1 (location of the Xlink schema document, use of OGC HTTP URIs for coordinate reference systems).
- The reference to the normative schema documents in [Annex C](#) now refers to the OGC schema repository. Previously, copies of the GML schema were also published on ISO servers, but the schema documents were not always synchronized. Going forward, all references to the normative GML schema document should go to <http://schemas.opengis.net/gml/>.

A list of all parts in the ISO 19136 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

Geography Markup Language (GML) is an XML grammar written in XML Schema for the description of application schemas as well as the transport and storage of geographic information.

The key concepts used by GML to model the world are drawn from the ISO 19100 series of International Standards and the OpenGIS Abstract Specification.

A feature is an “abstraction of real world phenomena” (ISO 19101); it is a geographic feature if it is associated with a location relative to the Earth so a digital representation of the real world may be thought of as a set of features. The state of a feature is defined by a set of properties, where each property may be thought of as a {name, type, value} triple.

The number of properties a feature may have, together with their names and types, is determined by its type definition. Geographic features with geometry are those with properties that may be geometry-valued. A feature collection is a collection of features that may itself be regarded as a feature; as a consequence a feature collection has a feature type and thus may have distinct properties of its own, in addition to the features it contains.

Following ISO 19109, the feature types of an application or application domain is usually captured in an application schema. A GML application schema is specified in XML Schema and can be constructed in two different and alternative ways:

- by adhering to the rules specified in ISO 19109 for application schemas in UML, and conforming to both the constraints on such schemas and the rules for mapping them to GML application schemas specified in this document;
- by adhering to the rules for GML application schemas specified in this document for creating a GML application schema directly in XML Schema.

Both ways are supported by this document. To ensure proper use of the conceptual modelling framework of the ISO 19100 series of International Standards, all application schemas are expected to be modelled in accordance with the General Feature Model as specified in ISO 19109. Within the ISO 19100 series, UML is the preferred language by which to model conceptual schemas.

GML specifies XML encodings, conformant with ISO 19118, of several of the conceptual classes defined in the ISO 19100 series of International Standards and the OpenGIS Abstract Specification. These conceptual models include those defined in:

- ISO/TS 19103 — Conceptual schema language (units of measure, basic types);
- ISO 19107 — Spatial schema (geometry and topology objects);
- ISO 19108 — Temporal schema (temporal geometry and topology objects, temporal reference systems);
- ISO 19109 — Rules for application schemas (features);
- ISO 19111 — Spatial referencing by coordinates (coordinate reference systems);
- ISO 19123 — Schema for coverage geometry and functions.

The aim is to provide a standardized encoding (i.e. a standardized implementation in XML) of types specified in the conceptual models specified by the International Standards listed above. If every application schema were encoded independently and the encoding process included the types from, for example, ISO 19108, then, without unambiguous and completely fixed encoding rules, the XML encodings would be different. Also, since every implementation platform has specific strengths and weaknesses, it is helpful to standardize XML encodings for core geographic information concepts modelled in the ISO 19100 series of International Standards and commonly used in application schemas.

In many cases, the mapping from the conceptual classes is straightforward, while in some cases the mapping is more complex (a detailed description of the mapping is part of this document).

In addition, GML provides XML encodings for additional concepts not yet modelled in the ISO 19100 series of International Standards or the OpenGIS Abstract Specification, for example, dynamic features, simple observations or value objects.

Predefined types of geographic features in GML include coverages and simple observations.

A coverage is a subtype of feature that has a coverage function with a spatiotemporal domain and a value set range of homogeneous 1- to n -dimensional tuples. A coverage may represent one feature or a collection of features “to model and make visible spatial relationships between, and the spatial distribution of, Earth phenomena” (OGC Abstract Specification Topic 6^[18]) and a coverage “acts as a function to return values from its range for any direct position within its spatiotemporal domain” (ISO 19123).

An observation models the act of observing, often with a camera or some other procedure, a person or some form of instrument (Merriam-Webster Dictionary: “an act of recognizing and noting a fact or occurrence often involving measurement with instruments”). An observation is considered to be a GML feature with a time at which the observation took place, and with a value for the observation.

A reference system provides a scale of measurement for assigning values to a position, time or other descriptive quantity or quality.

A coordinate reference system consists of a set of coordinate system axes that is related to the Earth through a datum that defines the size and shape of the Earth.

A temporal reference system provides standard units for measuring time and describing temporal length or duration.

A reference system dictionary provides definitions of reference systems used in spatial or temporal geometries.

Spatial geometries are the values of spatial feature properties. They indicate the coordinate reference system in which their measurements have been made. The “parent” geometry element of a geometric complex or geometric aggregate makes this indication for its constituent geometries.

Temporal geometries are the values of temporal feature properties. Like their spatial counterparts, temporal geometries indicate the temporal reference system in which their measurements have been made.

Spatial or temporal topologies are used to express the different topological relationships between features.

A units of measure dictionary provides definitions of numerical measures of physical quantities, e.g. length, temperature and pressure, and of conversions between units.

NOTE This document makes reference to ISO 19107:2003 and ISO 19111:2007 (withdrawn standards, replaced by 2019 versions) because this edition of ISO 19136-1 is still an XML implementation of the previous edition of ISO 19107 and other standards.

Geographic information — Geography Markup Language (GML) —

Part 1: Fundamentals

1 Scope

The Geography Markup Language (GML) is an XML encoding in accordance with ISO 19118 for the transport and storage of geographic information modelled in accordance with the conceptual modelling framework used in the ISO 19100 series of International Standards and including both the spatial and non-spatial properties of geographic features.

This document defines the XML Schema syntax, mechanisms and conventions that:

- provide an open, vendor-neutral framework for the description of geospatial application schemas for the transport and storage of geographic information in XML;
- allow profiles that support proper subsets of GML framework descriptive capabilities;
- support the description of geospatial application schemas for specialized domains and information communities;
- enable the creation and maintenance of linked geographic application schemas and datasets;
- support the storage and transport of application schemas and datasets;
- increase the ability of organizations to share geographic application schemas and the information they describe.

Implementers can decide to store geographic application schemas and information in GML, or they can decide to convert from some other storage format on demand and use GML only for schema and data transport.

NOTE If an ISO 19109 conformant application schema described in UML is used as the basis for the storage and transportation of geographic information, this document provides normative rules for the mapping of such an application schema to a GML application schema in XML Schema and, as such, to an XML encoding for data with a logical structure in accordance with the ISO 19109 conformant application schema.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601-1, *Date and time — Representations for information interchange — Part 1: Basic rules*

ISO/IEC 11404:2007, *Information technology — General-Purpose Datatypes (GPD)*

ISO 19108:2002, *Geographic information — Temporal schema*

ISO 19123:2005, *Geographic information — Schema for coverage geometry and functions*

ISO/IEC 19757-3, *Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron*

ISO 80000-3, *Quantities and units — Part 3: Space and time*

IETF RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax (August 1998)*

W3C XLink, XML Linking Language (XLink) Version 1.1, W3C Recommendation (6 May 2010)

W3C XML, Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation (26 November 2008)

W3C XML Namespaces, Namespaces in XML 1.0 (Third Edition), W3C Recommendation (8 December 2009)

W3C XML Schema Part 1, XML Schema Part 1: Structures, W3C Recommendation (28 October 2004)

W3C XML Schema Part 2, XML Schema Part 2: Datatypes, W3C Recommendation (28 October 2004)

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at <https://www.iso.org/obp>

— IEC Electropedia: available at <http://www.electropedia.org/>

3.1.1

application schema

conceptual *schema* ([3.1.52](#)) for data required by one or more applications

[SOURCE: ISO 19101-1:2014, 4.1.2]

3.1.2

association

<UML> semantic relationship that can occur between typed instances

[SOURCE: ISO 19103:2015, 4.4, modified — Note 1 to entry has been deleted.]

3.1.3

attribute

<XML> name-value pair contained in an *element* ([3.1.23](#))

Note 1 to entry: In this document an attribute is an XML attribute unless otherwise specified. The syntax of an XML attribute is “Attribute::= Name = AttValue”. An attribute typically acts as an XML element modifier (e.g. <Road gml:id = “r1” />; here gml:id is an attribute).

3.1.4

boundary

set that represents the limit of an entity

[SOURCE: ISO 19107:2019, 3.6, modified — Note 1 to entry has been deleted.]

3.1.5

child element

<XML> immediate descendant *element* of an *element* ([3.1.23](#))

3.1.6**closure**

union of the *interior* (3.1.39) and *boundary* (3.1.4) of a *topological object* (3.1.59) or *geometric object* (3.1.30)

[SOURCE: ISO 19107:2019, 3.8]

3.1.7**codelist**

value domain including a code for each permissible value

3.1.8**codespace**

rule or authority for a code, name, term or category

EXAMPLE Dictionaries, authorities, codelists, etc.

3.1.9**composite curve**

sequence of *curves* (3.1.17) such that each curve (except the first) starts at the end point of the previous curve in the sequence

Note 1 to entry: A composite curve, as a set of direct positions, has all the properties of a curve.

3.1.10**composite solid**

connected set of solids adjoining one another along shared *boundary* (3.1.4) surfaces

Note 1 to entry: A composite solid, as a set of direct positions, has all the properties of a solid.

3.1.11**composite surface**

connected set of *surfaces* (3.1.57) adjoining one another along shared *boundary* (3.1.4) curves

Note 1 to entry: A composite surface, as a set of direct positions, has all the properties of a surface.

3.1.12**coordinate**

one of a sequence of numbers designating the position of a *point* (3.1.47)

Note 1 to entry: In a spatial coordinate reference system, the coordinate numbers are qualified by units.

[SOURCE: ISO 19111:2019, 3.1.5]

3.1.13**coordinate reference system**

coordinate system (3.1.14) that is related to an object by a *datum* (3.1.19)

[SOURCE: ISO 19111:2019, 3.1.9, modified — Note 1 to entry has been deleted.]

3.1.14**coordinate system**

set of mathematical rules for specifying how *coordinates* (3.1.12) are to be assigned to *points* (3.1.47)

[SOURCE: ISO 19111:2019, 3.1.11]

3.1.15**coordinate tuple**

tuple (3.1.60) composed of *coordinates* (3.1.12)

[SOURCE: ISO 19111:2019, 3.1.13, modified — Note 1 to entry has been deleted.]

3.1.16

coverage

feature (3.1.26) that acts as a *function* (3.1.28) to return values from its *range* (3.1.50) for any *direct position* (3.1.20) within its spatial, temporal or spatiotemporal *domain* (3.1.21)

[SOURCE: ISO 19123:2005, 4.1.7, modified — Example and Note to entry have been deleted.]

3.1.17

curve

1-dimensional *geometric primitive* (3.1.31), representing the continuous image of a line

Note 1 to entry: The boundary of a curve is the set of points at either end of the curve. If the curve is a cycle, the two ends are identical, and the curve (if topologically closed) is considered to not have a boundary. The first point is called the start point, and the last is the end point. Connectivity of the curve is guaranteed by the “continuous image of a line” clause. A topological theorem states that a continuous image of a connected set is connected.

3.1.18

data type

specification of a value *domain* (3.1.21) with operations allowed on values in this domain

EXAMPLE Integer, Real, Boolean, String, Date (conversion of data into a series of codes).

Note 1 to entry: Data types include primitive predefined types and user-definable types. All instances of a data type lack identity.

[SOURCE: ISO 19103:2015, 4.14, modified — Supplementary information has been added to Example and Note 1 to entry.]

3.1.19

datum

parameter or set of parameters that realize the position of the origin, the scale, and the orientation of a *coordinate system* (3.1.14)

[SOURCE: ISO 19111:2019, 3.1.15]

3.1.20

direct position

position described by a single set of *coordinates* (3.1.12) within a *coordinate reference system* (3.1.13)

3.1.21

domain

well-defined set

Note 1 to entry: A mathematical *function* (3.1.28) may be defined on this set, i.e. in a function $f:A \rightarrow B$, A is the domain of the function f .

Note 2 to entry: A domain as in domain of discourse refers to a subject or area of interest.

[SOURCE: ISO 19109:2015, 4.8]

3.1.22

edge

<topology> 1-dimensional topological primitive

Note 1 to entry: The geometric realization of an edge is a curve. The boundary of an edge is the set of one or two nodes associated with the edge within a topological complex.

[SOURCE: ISO 19107:2019, 3.29]

3.1.23 element

<XML> basic information item of an XML document containing *child elements* (3.1.5), *attributes* (3.1.3) and character data

Note 1 to entry: The Extensible Markup Language (XML) 1.0 (Fifth Edition), Clause 3, uses the following definition: "Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its 'generic identifier' (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value."

3.1.24 exterior

<geometry, topology> difference between the universe and the *closure* (3.1.6)

Note 1 to entry: The concept of exterior is applicable to both topological object and geometric object.

[SOURCE: ISO 19107:2019, 3.37]

3.1.25 face

<topology> 2-dimensional topological primitive

Note 1 to entry: The geometric realization of a face is a surface. The boundary of a face is the set of directed edges within the same topology complex that are associated with the face via the boundary relations. These can be organized as rings.

[SOURCE: ISO 19107:2019, 3.38]

3.1.26 feature

abstraction of real world phenomena

Note 1 to entry: A feature may occur as a type or an instance. Feature type or feature instance will be used when only one is meant.

[SOURCE: ISO 19101-1:2014, 4.1.11]

3.1.27 feature association

relationship that links instances of one *feature* (3.1.26) type with instances of the same or a different feature type

[SOURCE: ISO 19110:2016, 3.3]

3.1.28 function

<mathematics, programming> rule that associates each *element* (3.1.23) from a *domain* (3.1.21) ("source domain," or "domain" of the function) to a unique element in another domain ("target domain," "co-domain," or "*range*" (3.1.50) of the function)

[SOURCE: ISO 19107:2019, 3.41]

3.1.29 geodetic datum

datum (3.1.19) describing the relationship of a 2- or 3-dimensional *coordinate system* (3.1.14) to the Earth

[SOURCE: ISO 19111:2007, 4.24]

3.1.30

geometric object

<geometry> *spatial object* ([3.1.56](#)) representing a *geometric set* ([3.1.32](#))

[SOURCE: ISO 19107:2019, 3.49, modified — Note 1 to entry has been deleted.]

3.1.31

geometric primitive

<geometry> *geometric object* ([3.1.30](#)) representing a single, connected, homogeneous (isotropic) element of space

[SOURCE: ISO 19107:2019, 3.50, modified — Note 1 to entry has been deleted.]

3.1.32

geometric set

set of *direct positions* ([3.1.20](#))

Note 1 to entry: This definition is taken from the definition in ISO 19107:2003 rather than ISO 19107:2019 since the 2019 version uses the term "point" with a broader meaning than in this document.

[SOURCE: ISO 19107:2003, 4.50]

3.1.33

geometry property

<GML> *property* ([3.1.49](#)) of a *GML feature* ([3.1.26](#)) that describes some aspect of the geometry of the feature

Note 1 to entry: The geometry property name is the role of the geometry in relation to the feature.

3.1.34

GML application schema

application schema ([3.1.1](#)) written in XML Schema in accordance with the rules specified in this document

3.1.35

GML document

XML document with a root element that is one of the elements *AbstractFeature*, *Dictionary* or *TopoComplex* specified in the *GML schema* ([3.1.37](#)) or any element of a substitution group of any of these elements

3.1.36

GML profile

subset of the *GML schema* ([3.1.37](#))

3.1.37

GML schema

schema ([3.1.52](#)) components in the *XML namespace* ([3.1.43](#)) "<http://www.opengis.net/gml/3.2>" as specified in this document

3.1.38

grid

network composed of two or more sets of *curves* ([3.1.17](#)) in which the members of each set intersect the members of the other sets in an algorithmic way

Note 1 to entry: The curves partition a space into grid cells.

[SOURCE: ISO 19123:2005, 4.1.23]

3.1.39

interior

set of all *direct positions* ([3.1.20](#)) that are on a *geometric object* ([3.1.30](#)) but which are not on its *boundary* ([3.1.4](#))

3.1.40**line string**

curve ([3.1.17](#)) composed of straight-line segments

3.1.41**measure**

<GML> value described using a numeric amount with a scale or using a scalar reference system

Note 1 to entry: When used as a noun, measure is a synonym for physical quantity.

3.1.42**measurand**

particular quantity subject to measurement

[SOURCE: VIM:1993, 2.6]

3.1.43**namespace**

<XML> collection of names, identified by a *URI* ([3.1.62](#)) reference, which are used in XML documents as element names and attribute names (W3C XML Namespaces)

3.1.44**node**

<topology> 0-dimensional topological primitive

Note 1 to entry: The boundary of a node is the empty set.

[SOURCE: ISO 19107:2019, 3.69]

3.1.45**observable type**

data type ([3.1.18](#)) to indicate the *physical quantity* ([3.1.46](#)) as a result of an observation

3.1.46**physical quantity**

quantity used for the quantitative description of physical phenomena

Note 1 to entry: In GML, a physical quantity is always a value described using a numeric amount with a scale or using a scalar reference system. Physical quantity is a synonym for measure when the latter is used as a noun.

3.1.47**point**

0-dimensional *geometric primitive* ([3.1.31](#)), representing a position

Note 1 to entry: The boundary of a point is the empty set.

3.1.48**polygon**

planar *surface* ([3.1.57](#)) defined by 1 *exterior* ([3.1.24](#)) *boundary* ([3.1.4](#)) and 0 or more *interior* ([3.1.39](#)) *boundaries* ([3.1.4](#))

3.1.49**property**

<GML> *child element* ([3.1.5](#)) of a GML object

Note 1 to entry: It corresponds to feature attribute and feature association role in ISO 19109. If a GML property of a feature has an xlink:href attribute that references a feature, the property represents a feature association role.

3.1.50

range

co-domain

<mathematics> acceptable target values of a *function* ([3.1.28](#))

[SOURCE: ISO 19107:2019, 3.80]

3.1.51

rectified grid

grid ([3.1.38](#)) for which there is an affine transformation between the grid coordinates and the *coordinates* ([3.1.12](#)) of an external *coordinate reference system* ([3.1.13](#))

[SOURCE: ISO 19123:2005, 4.1.32, modified — The NOTE has been deleted.]

3.1.52

schema

formal description of a model

Note 1 to entry: In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the relationship between the attributes and elements of an XML object.

EXAMPLE A document or a portion of a document.

[SOURCE: ISO 19101-1:2014, 4.1.34, modified — Note 1 to entry and EXAMPLE have been added.]

3.1.53

schema

<XML> collection of *schema* ([3.1.52](#)) components within the same target *namespace* ([3.1.43](#))

EXAMPLE Schema components of W3C XML Schema are types, elements, attributes, groups, etc.

3.1.54

schema document

<XML> XML document containing *schema* ([3.1.52](#)) component definitions and declarations

Note 1 to entry: The W3C XML Schema provides an XML interchange format for schema information. A single schema document provides descriptions of components associated with a single XML namespace, but several documents may describe components in the same schema, i.e. the same target namespace.

3.1.55

semantic type

category of objects that share some common characteristics and are thus given an identifying type name in a particular domain of discourse

3.1.56

spatial object

<topology, geometry> object used for representing a spatial characteristic of a *feature* ([3.1.26](#))

[SOURCE: ISO 19107:2019, 3.87]

3.1.57

surface

2-dimensional *geometric primitive* ([3.1.31](#)), locally representing a continuous image of a region of a plane

Note 1 to entry: The boundary of a surface is the set of oriented, closed curves that delineate the limits of the surface. Surfaces that are isomorphic to a sphere, or to an *n*-torus (a topological sphere with *n* “handles”) have no boundary. Such surfaces are called cycles.

3.1.58

tag

<XML> markup in an XML document delimiting the content of an *element* ([3.1.23](#))

EXAMPLE <Road>

Note 1 to entry: A tag with no forward slash (e.g. <Road>) is called a start-tag (also opening tag), and one with a forward slash (e.g. </Road> is called an end-tag (also closing tag).

3.1.59

topological object

spatial object (3.1.56) representing spatial characteristics that are invariant under continuous transformations

[SOURCE: ISO 19107:2019, 3.99, modified — Note 1 to entry has been deleted.]

3.1.60

tuple

ordered list of values

Note 1 to entry: The number of values in a tuple is immutable.

3.1.61

UML application schema

application schema (3.1.1) written in UML in accordance with ISO 19109

3.1.62

Uniform Resource Identifier

URI

unique identifier for a resource, structured in conformance with IETF RFC 2396

Note 1 to entry: The general syntax is <scheme>::<scheme-specific-part>. The hierarchical syntax with a namespace is <scheme>://<authority><path>?<query> — see RFC 2396.

3.2 Abbreviated terms

CRS	Coordinate Reference System
CS	Coordinate System
CSV	Comma Separated Values
CT	Coordinate Transformation
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geography Markup Language
NOTE The abbreviated term GML was previously used also as Generalized Markup Language (which led to SGML, Standard Generalized Markup Language, ISO 8879).	
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
RDF	Resource Description Framework
RFC	Request for Comments

SMIL	Synchronized Multimedia Integration Language
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WFS	Web Feature Service
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language — Transformations
0D	zero-dimensional
1D	one-dimensional
2D	two-dimensional
3D	three-dimensional

4 Conformance

4.1 Conformance requirements

[Clauses 7](#) to [19](#) of this document specify XML Schema components, i.e. the GML schema, which shall be used in GML application schemas in accordance with [Clause 21](#). [Clause 20](#) specifies rules for the specification of a GML profile that may be defined for use in a GML application schema.

Few applications will require the full range of capabilities described by the GML schema. This clause, therefore, defines a set of conformance classes that will support applications whose requirements range from the minimum necessary to define simple feature types to full use of the GML schema.

Most of the schema components specified in this document implement concepts defined in the ISO 19100 series of International Standards. In these cases, the conformance classes defined in this document are based on the conformance classes defined in the corresponding standard.

Any GML application schema, GML profile or software implementation claiming conformance with one of the conformance classes shall pass all test cases of the corresponding abstract test suite.

Any software implementation claiming conformance to this document shall document the GML profile supported by the implementation. The GML profile shall pass all mandatory test cases of the abstract test suite corresponding to GML profiles.

4.2 Conformance classes related to GML application schemas

GML application schemas claiming conformance to this document shall conform to the rules specified in [Clauses 7](#) to [21](#) and pass all relevant test cases of the abstract test suite in [A.1](#).

Depending on the characteristics of a GML application schema, 12 conformance classes are distinguished. [Table 1](#) lists these classes and the corresponding subclause of the abstract test suite.

Table 1 — Conformance classes related to GML application schemas

Conformance class	Subclause of the abstract test suite
All GML application schemas	A.1.1
GML application schemas converted from an ISO 19109 application schema in UML	A.1.2
GML application schemas to be converted to an ISO 19109 application schema in UML	A.1.3
GML application schemas defining features and feature collections	A.1.4
GML application schemas defining spatial geometries	A.1.5
GML application schemas defining spatial topologies	A.1.6
GML application schemas defining time	A.1.7
GML application schemas defining coordinate reference systems	A.1.8
GML application schemas defining coverages	A.1.9
GML application schemas defining observations	A.1.10
GML application schemas defining dictionaries and definitions	A.1.11
GML application schemas defining values	A.1.12

4.3 Conformance classes related to GML profiles

The requirements of an application schema determine the XML Schema components from the GML schema that shall be included in a GML profile. GML profiles claiming conformance to this document shall satisfy the requirements of the abstract test suite in [A.2](#).

Depending on the contents and requirements concerning a specific GML profile, 31 conformance classes are distinguished. [Table 2](#) lists these classes and the corresponding subclause of the abstract test suite.

Table 2 — Conformance classes related to GML profiles

Conformance class	Subclause of the abstract test suite
All GML profiles	A.2.1
Geometric primitives (spatial) — 0-dimensional	A.2.2.1.1
Geometric primitives (spatial) — 0/1-dimensional	A.2.2.1.2
Geometric primitives (spatial) — 0/1/2-dimensional	A.2.2.1.3
Geometric primitives (spatial) — 0/1/2/3-dimensional	A.2.2.1.4
Geometric complexes (spatial) — 0/1-dimensional	A.2.3.1.1
Geometric complexes (spatial) — 0/1/2-dimensional	A.2.3.1.2
Geometric complexes (spatial) — 0/1/2/3-dimensional	A.2.3.1.3
Topologic complexes (spatial) — 0/1-dimensional	A.2.4.1.1
Topologic complexes (spatial) — 0/1/2-dimensional	A.2.4.1.2
Topologic complexes (spatial) — 0/1/2/3-dimensional	A.2.4.1.3
Topologic complexes with geometric realization (spatial) — 1-dimensional	A.2.5.1.1
Topologic complexes with geometric realization (spatial) — 2-dimensional	A.2.5.1.2
Topologic complexes with geometric realization (spatial) — 3-dimensional	A.2.5.1.3
Coordinate reference systems	A.2.6
Coordinate operations between two coordinate reference systems	A.2.7
Temporal geometry — 0-dimensional	A.2.8.1
Temporal geometry — 0/1-dimensional	A.2.8.2
Temporal topology	A.2.9

Table 2 (continued)

Conformance class	Subclause of the abstract test suite
Temporal reference systems	A.2.10
Dynamic features	A.2.11
Dictionaries	A.2.12
Units dictionaries	A.2.13
Observations	A.2.14
Abstract coverage	A.2.15.1
Discrete point coverage	A.2.15.2
Discrete curve coverage	A.2.15.3
Discrete surface coverage	A.2.15.4
Discrete solid coverage	A.2.15.5
Grid coverage	A.2.15.6
Continuous coverage	A.2.15.7

Curve implementations, for those GML profiles including 1-dimensional spatial geometry objects, shall always include a “linear” interpolation technique. Surface implementations, for those GML profiles including 2-dimensional spatial geometry objects, shall always include a “planar” interpolation technique. Additional curve and surface interpolation mechanisms are optional but, if implemented, they shall follow the definition included in this document.

NOTE 1 Compare these conformance classes with ISO 19107:2003, Clause 2, ISO 19108:2002, 2.2, and ISO 19123:2005, Clause 2.

NOTE 2 A GML profile conforming to the three conformance classes “Geometric primitives (spatial) — 0-dimensional”, “Geometric primitives (spatial) — 0/1-dimensional”, and “Geometric primitives (spatial) — 0/1/2-dimensional” (in addition to conformance class “All GML profiles”) conforms to the spatial profile defined in ISO 19137:2007 and the respective conformance tests in ISO 19137:2007, B.1, B.2 and B.3.

4.4 Conformance classes related to GML documents

GML documents claiming conformance to this document shall conform to the rules specified in [Clauses 7](#) to [21](#) and pass all relevant test cases of the abstract test suite in [A.3](#).

4.5 Conformance classes related to software implementations

Software implementations reading or writing GML or GML application schemas claiming conformance to this document shall pass all of the corresponding abstract test suites described in the abstract test suite in [Annex B](#).

Depending on the capabilities of the implementation, 11 conformance classes are distinguished. [Table 3](#) lists these classes and the corresponding subclause of the abstract test suite.

Table 3 — Conformance classes related to implementations

Conformance class	Subclause of the abstract test suite
All software implementations	B.1
Support for remote simple Xlinks	B.2.1
Support for extended Xlinks	B.2.2
Support for nillable properties	B.2.3
Support for units of measurement	B.2.4
Support for ownership semantics of properties	B.2.5

Table 3 (continued)

Conformance class	Subclause of the abstract test suite
Metadata properties	B.2.6
Support for GML profiles in instance validation	B.2.7
Writing GML	B.3
Reading GML	B.4
Writing GML application schemas	B.5
Reading GML application schemas	B.6

5 Conventions

5.1 XML namespaces

All components of the GML schema are defined in the namespace with the identifier "<http://www.opengis.net/gml/3.2>", for which the prefix **gml** or the default namespace is used within this document.

All components described by the W3C Xlink Recommendation are defined in the namespace with the identifier "<http://www.w3.org/1999/xlink>", for which the prefix **xlink** is used within this document.

NOTE The schema components in both namespaces are documented in XML Schema documents in [Annex C](#).

5.2 Versioning

Each schema document specifying components of the GML schema shall carry a version attribute as defined in the XML Schema Recommendation. The format of the version attribute string is x.y.z where x denotes the major version number, y denotes a minor version number and z denotes a bug fix release for that document. The version described by this document is 3.2.2.

NOTE GML 3.2 is the first version of GML as an ISO standard. Previous versions of GML have been developed and published by the Open Geospatial Consortium (OGC).

5.3 Deprecated parts of previous versions of GML

The verb "**deprecate**" provides notice that the referenced portion of this document is being retained for backwards compatibility with earlier versions but may be removed from a future version without further notice.

Sections of this document that describe or refer to deprecated GML components are written in *italics*.

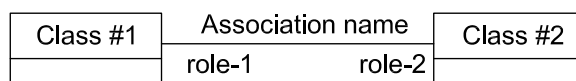
NOTE 1 This document is published by the Open Geospatial Consortium as GML 3.2.2, the previous adopted version of GML in the Open Geospatial Consortium was 3.1.1.

NOTE 2 All schema components that were part of version 2.1 of GML but were deprecated in version 3.0 of GML have been removed and are not supported by this document.

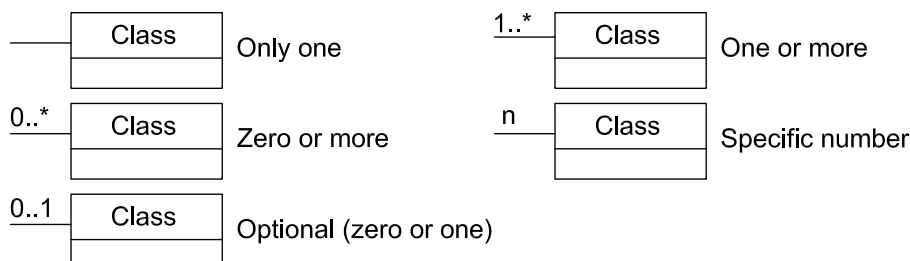
5.4 UML notation

Many diagrams that appear in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in [Figure 1](#).

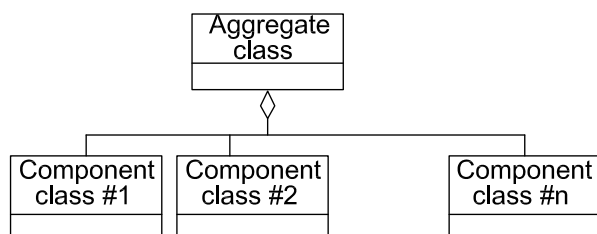
Association between classes



Association Cardinality



Aggregation between classes



Class Inheritance (subtyping of classes)

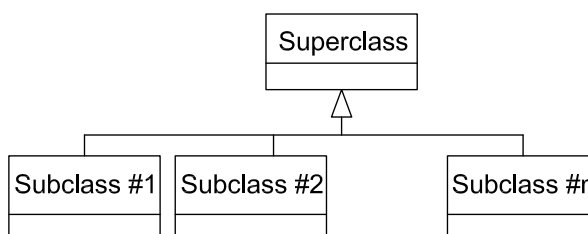


Figure 1 — UML notation

In this document, the following stereotypes of UML class are used:

- `<<DataType>>` is a set of properties that lack identity (independent existence and the possibility of side effects). A `DataType` is a class with no operations whose primary purpose is to hold the information.
- `<<Union>>` is a set of properties. The semantics is that only one of the properties may be present at any time.
- `<<FeatureType>>` is a feature as defined in ISO 19109.
- `<<CodeList>>` is a flexible enumeration that uses string values for expressing a list of potential values.
- `<<Enumeration>>` is a fixed list of valid identifiers of named literal values. Attributes of an enumerated type may only take values from this list.
- `<<Abstract>>` is an abstract object type (the stereotype is used in addition to formatting the class name in italics).
- `<<Type>>` is a set of abstract attributes and associations. Abstract means that their specification does not imply that they have to be concretely implemented as instance variables.

In this document, the following standard data types are used:

- `CharacterString` — A sequence of characters (in general this data type is mapped to “string” in XML Schema).
- `Integer` — An integer number (in general this data type is mapped to “integer” in XML Schema).
- `Real` — A floating point number (in general this data type is mapped to “double” in XML Schema).

- Boolean — A value specifying TRUE or FALSE (in general this data type is mapped to “boolean” in XML Schema).

5.5 XML Schema

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

6 Overview of the GML schema

6.1 GML schema

GML specifies XML encodings of a number of the conceptual classes defined in the ISO 19100 series of International Standards and the OpenGIS Abstract Specification in conformance with these standards and specifications.

The relevant conceptual models include those defined in:

- ISO 19103 — Conceptual schema language (units of measure, basic types);
- ISO 19107 — Spatial schema (spatial geometry and topology);
- ISO 19108 — Temporal schema (temporal geometry and topology, temporal reference systems);
- ISO 19109 — Rules for application schemas (features);
- ISO 19111 — Spatial referencing by coordinates (coordinate reference systems);
- ISO 19123 — Schema for coverage geometry and functions (coverages, grids).

In many cases, the mapping from the conceptual classes to XML is straightforward, while in some cases the mapping is more complex. For both cases, the mapping is documented in detail in [Annex D](#).

In addition, GML provides XML encodings for additional concepts not yet modelled in the ISO 19100 series of International Standards or the OpenGIS Abstract Specification. Examples include moving objects, simple observations or value objects. Additional conceptual classes corresponding to these extensions are also specified in [Annex D](#).

The GML schema comprises the components (e.g. XML elements, attributes, simple types, complex types, attribute groups, groups) that are described in this document. The XML encoding conforms to ISO 19118.

6.2 GML application schemas

Designers of GML application schemas may extend or restrict the types defined in the GML schema to define appropriate types for an application domain. Non-abstract elements, attributes and types from the GML schema may be used directly in an application schema, if no changes are required.

Following ISO 19109, the feature types of an application or application domain are specified in an application schema. A GML application schema shall be specified in XML Schema and import the GML schema. It may be constructed in one of two different ways:

- By adhering to the rules for GML application schemas specified in [Clause 21](#) for creating a GML application schema directly in XML Schema.

- By adhering to the rules specified in ISO 19109 for application schemas in UML, and conforming to both the constraints on such schemas and the rules for mapping them to GML application schemas specified in [Annex E](#) of this document. The mapping from an ISO 19109 conformant Application Schema in UML to the corresponding GML application schema is based on a set of encoding rules. These encoding rules conform with the rules for GML application schemas and ISO 19118.

Both ways are valid approaches to construct GML application schemas. All application schemas shall be modelled in accordance with the General Feature Model specified in ISO 19109. Within the ISO 19100 series of International Standards, UML is the preferred language to describe conceptual schemas.

The second approach is recommended in general to ensure proper use of the conceptual modelling framework of the ISO 19100 series of International Standards. However, the following reasons are examples where it may be justified to apply the first approach:

- Additional capabilities of the GML schema may be required in addition to the capabilities that are accessible by using the encoding rules specified in [Annex E](#).
- Only an XML representation may be required and the application schema may be relatively simple, so the use of a conceptual schema language may be considered an unjustified overhead.
- The application may need a more optimized or compact XML encoding than the one that is the result of the encoding rules specified in [Annex E](#).

NOTE [Annex F](#) provides rules for mapping a GML application schema to an ISO 19109 conformant Application Schema in UML.

In both cases, GML application schemas conformant with this document shall use all of the applicable GML schema components, either directly or by specialization, and are valid in accordance with the rules for XML Schema. How the GML application schemas were produced is not relevant for conformance to the requirements of this document.

6.3 Relationship between the ISO 19100 series of International Standards, the GML schema and GML application schemas

The approach taken by this document is shown in [Figure 2](#). The two main aspects are:

- Clear documentation of the conceptual model of GML: The profile of the ISO 19100 series of International Standards that is implemented by GML is documented as well as the extensions to this profile.
- Support for application schema development either in UML or XML Schema: In order to achieve this two-way mapping between UML (i.e. ISO 19109 conformant application schemas in UML) and XML Schema (i.e. GML application schemas in XML Schema) the constructs used in both representations have been limited. While this reduces the expressiveness of the schema descriptions to some extent, this also reduces their complexity and may make them easier to implement.

NOTE While the mapping from UML to XML Schema is discussed in ISO 19118:2011, Annex A, the reverse mapping is not discussed in any other standard in the ISO 19100 series of International Standards.

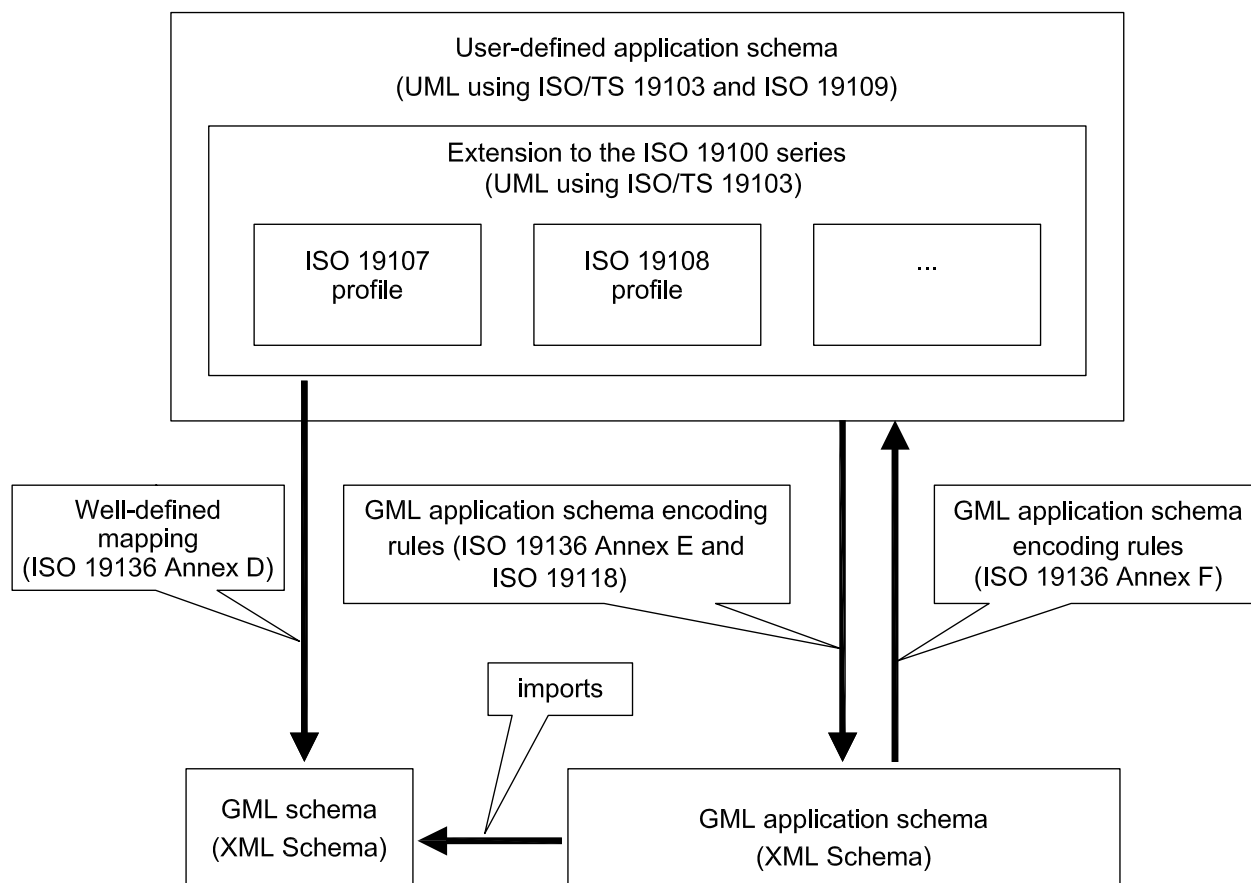


Figure 2 — Relationship between the ISO 19100 series of International Standards and ISO 19136/GML

6.4 Organization of this document

GML defines the various entities, e.g. features, geometries, topologies, through a hierarchy of GML object types. The mapping between GML object types and classes in the conceptual model of the ISO 19100 series of International Standards and the OGC Abstract Specification is shown in [Table D.2](#). The normative GML schema is organized around these object types.

[Subclause 7.2](#) describes basic schema components of GML. It defines the root object, `gml:AbstractObject`, and the root of the GML class hierarchy, `gml:AbstractGML`.

[Subclause 8.1](#) describes the Xlink schema. This schema is an OGC implementation of the XLink specification using XML Schema. It may be replaced in some future release by an equivalent schema from the W3C.

NOTE 1 Within this document an XML Schema description is provided for xlink components. This is provided for convenience in the context of an XML Schema-based environment. The normative definitions are given a non-XML Schema form in the XLink Recommendation.

[Subclause 8.2](#) defines the GML representation of some basic data types that are used in the GML schema. Most of these types are simple types or simple content types.

[Clause 9](#) describes the feature schema components which defines `gml:AbstractFeature` and some derived components.

[Clause 10](#), [10.5.10](#) and [Clause 11](#) describe the geometry schema components that define `gml:AbstractGeometry`, `gml:AbstractGeometricPrimitive`, `gml:AbstractGeometricAggregate`, `gml:GeometricComplex` and some derived components.

[Clause 12](#) describes the coordinate reference system schema components that define the subtypes of `gml:IdentifiedObject`, `gml:AbstractCRS`, `gml:AbstractCoordinateReferenceSystem`, and the elements and types required to construct specific coordinate reference systems.

[Clause 13](#) describes the schema components for topology which define `gml:AbstractTopology`, `gml:AbstractTopoPrimitive`, `gml:TopoComplex` and some derived components.

[Clause 14](#) describes the schema components for temporal constructs defining `gml:AbstractTimeObject`, `gml:AbstractTimePrimitive`, `gml:AbstractTimeGeometricPrimitive`, `gml:AbstractTimeTopologyPrimitive`, `gml:AbstractTimeComplex` and derived components as well as `gml:DynamicFeature` and derived components.

[Clause 15](#) describes the schema components for definitions and dictionaries including `gml:Definition` and `gml:Dictionary`.

[Clause 16](#) describes the schema components for the construction of units of measure (`gml:UnitDefinition` and derived components), measures and value objects (`gml:AbstractValue`, `gml:AbstractScalarValue`, `gml:AbstractScalarValueList` and derived components).

[Clause 17](#) describes the schema components for the description of direction.

[Clause 18](#) describes the schema components for simple observations (`gml:Observation` and derived components).

[Clause 19](#) describes the schema components for grids and coverages. This describes `gml:Grid`, `gml:AbstractCoverage`, `gml:AbstractDiscreteCoverage`, `gml:AbstractContinuousCoverage` and derived components.

These clauses describe the normative GML schema and explain their contents, structure and dependencies.

The representation of the GML schema presented in this document uses the XML interchange format provided by W3C XML Schema. The descriptions of the set of components are factored into schema documents, where each document gathers together components that correspond broadly to the classification shown in [Figure 2](#). However, while the XML representation of each GML schema component in this document is normative, the packaging into schema documents is not. [Clause 20](#) (profiles) and [Annex G](#) (subsetting) describe principles and methods for alternative packaging of the XML representation of GML schema components.

All components defined or declared in this document use the same target namespace of <http://www.opengis.net/gml/3.2>.

NOTE 2 XML namespaces provide a mechanism for avoiding ambiguity arising from name clashes within XML documents. All components described in a single schema document are in a single target namespace, but more than one schema document can describe components in a namespace. Within the XML development community there are precedents for assigning either one or several namespaces to a set of schema components for a single application. The use of a single namespace for GML schema components is consistent with the non-normative factoring of the XML representation of GML components between schema documents.

UML uses packages to collect related components. Furthermore, within the ISO 19100 series of International Standards, prefixes following the pattern "AA_" are used to distinguish classes from different packages in a way that resembles XML namespaces. However, for the reasons given above, packaging of GML components is non-normative and all GML components are in a single namespace, so no correspondence between the two letter prefixes in the ISO 19100 series of International Standards and XML namespaces in GML is possible.

6.5 Deprecated and experimental schema components

Experimental, informative schema components dealing with rules for a default styling of GML objects are described in [Annex H](#).

Deprecated global schema components (elements, attributes, types) are included in [Annex I Documents](#)

7 GML schema — General rules and base schema components

7.1 GML model and syntax

7.1.1 GML instance documents

GML uses an explicit syntax to instantiate a GML application schema conformant with the General Feature Model defined in ISO 19109 in an XML document.

A feature is encoded as an XML element with the name of the feature type. Other identifiable objects are encoded as XML elements with the name of the object type.

Each feature attribute and feature association role is a property of a feature. Feature properties are encoded in an XML element.

NOTE 1 The term "attribute" in XML refers to a specific syntactic component in XML documents, so to avoid confusion when describing the XML encoding, GML follows RDF (W3C, 1999) terminology and uses the term property rather than attribute or association role. The General Feature Model (ISO 19109) also uses the term "property" as a generalization for "attribute", "association role" or "operation".

Furthermore, the property semantics, which is indicated by the name of the element representing the property, is distinguished from the property value, which is given by the content of the property element. A property element may contain its value as content encoded inline, or reference its value with a simple XLink. The value of a property may be simple, or it may be a feature or other complex object. When recorded inline, the value of a simple property is recorded as a literal value with no embedded markup (text), while if the value is complex it appears as a subtree using XML markup (i.e. an XML element with sub-structure).

NOTE 2 The GML model has a straightforward representation using the UML profile used in the ISO 19100 series of International Standards (defined in ISO 19103). This is described in detail in [Annex D](#) and [Annex E](#), but can be summarized approximately and briefly as follows.

Features are represented

- in UML by objects, where the name of the feature type is used as the name of the object class;
- in GML instances by XML elements, where the name of the feature type is used as the name of the element.

Feature properties are represented

- in UML by association roles with feature type classes, and attributes of feature type classes, where the property semantics are given by the association role name or attribute name;
- in GML instances by sub-elements (known as property elements) of feature elements, where the property semantics are given by the property element name.

The property value has a type indicated

- in UML by the class of the association target, or by the data type of the attribute;
- in GML, in the case of properties with complex values, by the name of the object element contained within the property element and in case of a property with simple value by the type of the literal value containing no embedded XML markup.

The result is a layered XML document, in which XML elements corresponding to features, objects or values occur interleaved with XML elements corresponding to the properties that relate them. The

function of a feature, object or value in context can always be determined by inspecting the name of the property element which directly contains it, or which carries the reference to it.

NOTE 3 This encoding pattern is sometimes referred to as the “object-property model” and has been the basis of the GML encoding model since the first version was adopted by OGC. While in some cases this encoding pattern adds extra levels of elements in instance documents it also provides significant benefits: It helps to make a GML instance document understandable on its own, provides a predictable structure and avoids too heavy reliance on XML Schema as it is expected that GML instance documents can outlive the common use of W3C XML Schema language.

7.1.2 Lexical conventions

There are several lexical conventions used in the GML schema for the names of elements and complex types to assist in human comprehension of GML instances and schemas:

- objects are instantiated as XML elements with a conceptually meaningful name in UpperCamelCase;
- properties are instantiated as XML elements whose name is in lowerCamelCase;
- abstract elements have a prefix “Abstract” (objects) or “abstract” (properties) prepended to their name;
- the names of XML Schema complex types are in UpperCamelCase ending in the word “Type”;
- abstract XML Schema complex types have the word “Abstract” prepended.

It is strongly recommended to follow these conventions also in GML application schemas. The rules are only applicable in languages that distinguish between upper and lower case.

NOTE UpperCamelCase is a naming convention in which a name is formed of multiple words that are joined together as a single word with the first letter of each of the multiple words capitalized within the new word that forms the name. lowerCamelCase is a variation in which the first letter of the new word is lower case, allowing it to be easily distinguished from an UpperCamelCase name.

7.1.3 XML Schema definition of GML language

The GML schema consists of W3C XML Schema components that define types and declare

- XML elements to encode GML objects with identity,
- XML elements to encode GML properties of those objects, and
- XML attributes qualifying those properties.

A GML object is an XML element of a type derived directly or indirectly from `gml:AbstractGMLType`. From this derivation, a GML object may have a `gml:id` attribute.

A GML property shall not be derived from `gml:AbstractGMLType`, shall not have a `gml:id` attribute, or any other attribute of XML type ID.

An element is a GML property if and only if it is a child element of a GML object.

A GML object shall not appear as the immediate child of a GML object.

Consequently, no element may be both a GML object and a GML property.

All XML attributes declared in the GML schema are defined without namespace, the only exception is the `gml:id` XML attribute.

The use of additional XML attributes in a GML application schema is discouraged.

7.2 gmlBase schema components

7.2.1 Goals of base schema components

The gmlBase schema components establish the GML model and syntax, in particular

- a root XML type from which XML types for all GML objects should be derived,
- a pattern and components for GML properties,
- patterns for collections and arrays, and components for generic collections and arrays,
- components for associating metadata with GML objects,
- components for constructing definitions and dictionaries.

NOTE The corresponding schema document in [Annex C](#) is identified by the following location-independent name (using URN syntax):

- urn:ogc:specification:gml:schema-xsd:gmlBase:3.2.1

7.2.2 Base objects

7.2.2.1 AbstractObject

An abstract convenience element `gml:AbstractObject` is declared as follows:

```
<element name="AbstractObject" abstract="true"/>
```

This element has no type defined, and is therefore implicitly (in accordance with the rules of W3C XML Schema) an XML Schema anyType. It is used as the head of an XML Schema substitution group which unifies complex content and certain simple content elements used for datatypes in GML, including the `gml:AbstractGML` substitution group.

NOTE `gml:AbstractObject` is defined primarily to act as a variable in certain aggregate patterns where it is necessary to allow either elements in the `gml:AbstractGML` substitution group, or certain complex content or simple content elements to be valid in an instance.

A GML dataset (also called a data instance or data document) is represented by an object element. This object may in turn be a collection of GML objects.

7.2.2.2 AbstractGML, AbstractGMLType

The most basic components for representations of identifiable objects are described in the schema as follows:

```
<element name="AbstractGML" type="gml:AbstractGMLType" abstract="true" substitutionGroup="
gml:AbstractObject"/>
```

```
<complexType name="AbstractGMLType" abstract="true">
  <sequence>
    <group ref="gml:StandardObjectProperties"/>
  </sequence>
  <attribute ref="gml:id"/>
</complexType>
```

```
<group name="StandardObjectProperties">
  <sequence>
    <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
    <element ref="gml:description" minOccurs="0"/>
    <element ref="gml:descriptionReference" minOccurs="0"/>
    <element ref="gml:identifier" minOccurs="0"/>
    <element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</group>
```

```
</sequence>
</group>
```

The abstract element `gml:AbstractGML` is “any GML object having identity”. It acts as the head of an XML Schema substitution group, which may include any element which is a GML feature, or other object, with identity. This is used as a variable in content models in GML core and application schemas. It is effectively an abstract superclass for all GML objects.

The pairing of `gml:AbstractGML` and `gml:AbstractGMLType` shows a basic pattern used in the GML schema, whereby each GML object type is represented by a global element declaration, which has an associated XML Schema type definition. The name of an element representing a GML object indicates the conceptual meaning of the object. Generic element names in GML include `gml:AbstractObject`, `gml:AbstractGML`, `gml:AbstractFeature`, `gml:AbstractValue`, `gml:AbstractCoverage`, `gml:AbstractTopology` and `gml:AbstractCRS`. These other generic elements representing objects are defined elsewhere in this document.

The child XML elements and XML attributes of a GML object are properties of that object. Thus an object represented by an `gml:AbstractGML` element has five non-deprecated properties: `gml:identifier`, `gml:description`, `gml:descriptionReference`, `gml:name` and `gml:id`. These are described in [7.2.4](#).

NOTE The group `gml:StandardObjectProperties` is provided for convenience in the construction of application schema, particularly when it is desired to define types derived by restriction from `gml:AbstractGMLType` and `gml:AbstractFeatureType`. Derivation by restriction requires that all components that are used unchanged are copied down into the new type definition. As an alternative to including element declarations for all the standard object properties, a one line reference to `gml:StandardObjectProperties` can be used instead:

```
<group ref="gml:StandardObjectProperties"/>
```

7.2.3 GML properties

7.2.3.1 General

The term “property” is used to refer to a GML property, which is any characteristic of a GML object. An element in a GML document or data stream is a GML property if and only if it is a child element of a GML object element. The meaning of each property shall be indicated by the name of the element that instantiates it.

GML objects may have an unlimited number of properties, in addition to those inherited from `gml:AbstractGMLType`. A property may be defined to have either simple or complex content. A property with simple content has an XML Schema simple content type, as illustrated by the case of the standard property elements `gml:description` and `gml:name`. A property with complex content has an XML Schema complex content type.

Property elements may use two modes:

- **inline:** the value of the property is represented directly, as the content of the property element. This method is used by the standard property `gml:name` and may be used for `gml:description` (see [7.2.4.2](#)).
- **by reference:** the value of the property is available elsewhere, and is identified by the value of an `xlink:href` attribute on the property element. This alternative method shall be used for the standard property `gml:descriptionReference` (see [7.2.4.3](#)).

EXAMPLE See [8.1](#) for examples on the use of `xlink` references.

NOTE The roles of feature associations as defined in ISO 19109 (General Feature Model) and OpenGIS Abstract Specification Topic 8 can be represented in several ways in a GML application schema:

- By implementing only one role of the association as navigable, i.e. representing it in the XML encoding. This is the usual representation in the GML schema itself with some exceptions, for example, the boundary and co-boundary association roles between the topology objects.

- By specifying individual properties in the feature types participating in the association. However in this case, the consistency constraints implied by the association cannot be enforced by XML Schema validation. This encoding style is, for example, used for the boundary and co-boundary association roles between the topology objects and in [Annex E](#). See also [7.2.3.9](#).
- By creating an association object as a GML object. This also allows n-ary associations and associations with properties to be modelled.
- By using extended Xlinks. This encoding is similar to the “association object” representation.

7.2.3.2 AssociationAttributeGroup

XLink components are the standard method to support hypertext referencing in XML. An XML Schema attribute group, `gml:AssociationAttributeGroup`, is provided to support the use of Xlinks as the method for indicating the value of a property by reference in a uniform manner in GML. This attribute group is defined as follows:

```
<attributeGroup name="AssociationAttributeGroup">
  <attributeGroup ref="xlink:simpleLink"/>
  <attribute name="nilReason" type="gml:nilReasonType"/>
  <attribute ref="gml:remoteSchema"/>
</attributeGroup>
```

with the following definitions from Xlink (see [8.1](#)):

```
<attributeGroup name="simpleLink">
  <attribute name="type" type="string" fixed="simple" form="qualified"/>
  <attribute ref="xlink:href"/>
  <attribute ref="xlink:role"/>
  <attribute ref="xlink:arcrole"/>
  <attribute ref="xlink:title"/>
  <attribute ref="xlink:show"/>
  <attribute ref="xlink:actuate"/>
</attributeGroup>
```

The value of a GML property that carries an `xlink:href` attribute is the resource returned by traversing the link.

The `nilReason` attribute may be used in a property element that is nillable to indicate a reason for a nil value.

NOTE All components in the attribute group are optional.

7.2.3.3 abstractAssociationRole, AssociationRoleType

To support the encoding of properties that may have complex content, a basic pattern for property elements is provided in the GML schema as follows:

```
<element name="abstractAssociationRole" type="gml:AssociationRoleType"
abstract="true"/>

<complexType name="AssociationRoleType">
  <sequence minOccurs="0">
    <any namespace="##any"/>
  </sequence>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Applying this pattern shall restrict the multiplicity of objects in a property element using this content model to exactly one. An instance of this type shall contain an element representing an object, or serve as a pointer to a remote object.

Applying the pattern to define an application schema specific property type allows to restrict

- the inline object to specified object types,
- the encoding to “by reference only” (see [7.2.3.7](#)),

— the encoding to "inline only" (see [7.2.3.8](#)).

NOTE 1 The declaration of `gml:abstractAssociationRole` and its accompanying type definition is provided for convenience, to act as a template or pattern for the construction of property elements in application schemas. There is no requirement for specific properties to use XML Schema type derivation from `gml:AssociationType` to create properties in a conformant GML application schema. This contrasts with the requirement that the content model for all identifiable objects has to derive from `gml:AbstractGMLType`, and for all features from `gml:AbstractFeatureType`.

NOTE 2 While `gml:abstractAssociationRole` is abstract, its type `gml:AssociationRoleType` is not, because the same type is used by the instantiable `gml:member` property (see [7.2.3.10](#)). Note also that this property has been deprecated.

7.2.3.4 Inline or by reference?

The `any` element in the content model for properties is optional. In combination with the component cardinalities in `gml:AssociationAttributeGroup` this means that an element of this type may have a content element or `xlink` attributes. GML property elements which follow this pattern may be used to attach values either inline or by-reference.

EXAMPLE A utility property provided for features is "centerOf". This may be used to indicate a spatial location inline as follows:

```
<gml:centerOf>
  <gml:Point gml:id="point96" srsName="http://www.opengis.net/def/crs/EPSSG/0/4326">
    <gml:pos>-31.936 15.834</gml:pos>
  </gml:Point>
</gml:centerOf>
```

which uses the `gml:Point` object as defined in the GML geometry schemas (described in [10.2](#)). The same property element may be used to indicate a location by reference as follows:

```
<gml:centerOf xlink:href="http://my.big.org/locations/point53"/>
```

where "<http://my.big.org/location/point53>" identifies a point (a `gml:Point` element) supplied by the service indicated.

However, a property element following this pattern may have no content or attributes, or it may have both content and attributes, and still be XML Schema-valid. It is not possible to constrain the co-occurrence of content or attributes, so it is not possible to use W3C XML Schema to restrict a property to be either inline or by-reference only.

If both a link and content are present in an instance of a property element, then the object found by traversing the `xlink:href` link shall be the normative value of the property. The object included as content shall be used by the data recipient only if the remote instance cannot be resolved; this may be considered to be a "cached" version of the object.

NOTE Most GML-Object-valued properties in the GML schema can be encoded either inline or by-reference. However, using a GML profile (see [Clause 20](#)) it is possible to restrict the usage to "inline only" or "by-reference only".

7.2.3.5 Ownership of property values

Encoding a GML property inline vs. by-reference shall not imply anything about the "ownership" of the contained or referenced GML object, i.e. the encoding style shall not imply any "deep-copy" or "deep-delete" semantics. To express ownership over the contained or referenced GML object, the `gml:OwnershipAttributeGroup` attribute group may be added to object-valued property elements. If the attribute group is not part of the content model of such a property element, then the value may not be "owned".

The attribute group is defined as follows:

```
<attributeGroup name="OwnershipAttributeGroup">
  <attribute name="owns" type="boolean" default="false"/>
</attributeGroup>
```

When the value of the `owns` attribute is “true”, the existence of inline or referenced object(s) depends upon the existence of the parent object.

EXAMPLE If a property “hasOwner” is represented in an instance document as

```
<Parcel gml:id="p123">
  <hasOwner xlink:href="urn:x-abc:id:o123"/>
</Parcel>
```

then the referenced object, e.g. a person, is not “owned” by the parcel feature, i.e. the person feature will not be deleted, if the parcel is deleted. However, if a property is encoded with an attribute `owns="true"`, for example

```
<Car gml:id="c123">
  <hasParts owns="true" xlink:href="urn:x-abc:id:x123"/>
  <!-- ... -->
</Car>
```

then the referenced object is “owned” by the car feature, i.e. the part will be deleted, if the car is deleted.

7.2.3.6 abstractStrictAssociationRole

The constraint that the value of a property may be either embedded inline or specified by an xlink reference may be described precisely using the auxiliary constraint language Schematron (ISO/IEC 19757-3). The abstract, global elements `gml:abstractAssociationRole` and `gml:abstractStrictAssociationRole` both use `gml:AssociationRoleType`, but the following schema fragments shows how an element declaration may be accompanied by a Schematron constraint to limit the property to act in either inline or by-reference mode, but not both.

```
<element name="abstractAssociationRole" type="gml:AssociationRoleType"
abstract="true"/>

<element name="abstractStrictAssociationRole" type="gml:AssociationRoleType"
abstract="true"/>

<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" xmlns:gml="http://www.
opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink" xml:lang="en">
  <sch:title>Schematron constraints for GML / ISO 19136</sch:title>
  <sch:ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>
  <sch:ns prefix="gml" uri="http://www.opengis.net/gml/3.2"/>
  <sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <sch:pattern>
    <sch:rule context="gml:abstractStrictAssociationRole">
      <sch:assert test="not(@xlink:href and (*|text()))">Property element may not
carry both a reference to an object and contain an object.</sch:assert>
      <sch:assert test="@xlink:href | (*|text())">Property element shall either
carry a reference to an object or contain an object.</sch:assert>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

NOTE Some XML validators will process the Schematron constraints automatically. Otherwise, the Schematron code can be seen as a formal description of a constraint. It is included here primarily as an illustration of how this can be used for specific purposes by application schema developers.

7.2.3.7 abstractReference, ReferenceType

In order to support the encoding of properties whose value is provided remotely by-reference, the following components are provided:

```
<element name="abstractReference" type="gml:ReferenceType" abstract="true"/>

<complexType name="ReferenceType">
  <sequence/>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
```

```
<attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

The element `gml:abstractReference` is abstract, and thus may be used as the head of a substitution group of more specific elements providing a value by-reference.

NOTE While `gml:abstractReference` is abstract, its type `gml:ReferenceType` is not, because the type is intended to be used in application schemas directly, if a property element is intended to always use a “by-reference only” encoding.

7.2.3.8 abstractInlineProperty, InlinePropertyType

In order to support the encoding of properties whose value is provided inline, the following components are provided:

```
<element name="abstractInlineProperty" type="gml:InlinePropertyType" abstract="true"/>

<complexType name="InlinePropertyType">
  <sequence>
    <any namespace="##any"/>
  </sequence>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The element `gml:abstractInlineProperty` is abstract, and thus may be used as the head of a substitution group of more specific elements providing a value inline.

7.2.3.9 Properties representing the same relationship

If the value of an object property is another object and that object contains also a property for the association between the two objects, then this name of the reverse property may be encoded in a `gml:reversePropertyName` element in an `appinfo` annotation of the property element to document the constraint between the two properties. The value of the element shall contain the qualified name of the property element.

```
<element name="reversePropertyName" type="string"/>
```

EXAMPLE

```
<element name="owner" type="ex:PersonPropertyType" minOccurs="0">
  <annotation>
    <appinfo>
      <gml:reversePropertyName>ex:owns</gml:reversePropertyName>
    </appinfo>
  </annotation>
</element>
...
<complexType name="PersonPropertyType">
  <sequence minOccurs="0">
    <element ref="ex:Person"/>
  </sequence>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

7.2.3.10 Properties of value objects

Value objects, see [16.4](#), are special objects in the sense that in the case of a single property that can be represented by a single literal value, the value appears as the direct content of object element without an extra element for the property.

EXAMPLE `<gml:Integer>5</gml:Integer>` is used instead of, for example, `<gml:Integer> <gml:value>5</gml:value> </gml:Integer>`.

7.2.4 Standard properties of GML objects

7.2.4.1 Derivation from AbstractGMLType

XML Schema types for all GML objects derive directly or indirectly from `gml:AbstractGMLType`. This means that all GML objects inherit certain standard properties that are included in the content model of `gml:AbstractGMLType`.

7.2.4.2 description

The value of this property is a text description of the object. `gml:description` uses `gml:StringOrRefType` (see 0) as its content model, i.e. it should contain a simple text string content.

```
<element name="description" type="gml:StringOrRefType"/>
```

NOTE The use of `gml:description` to reference an external description has been deprecated and replaced by the `gml:descriptionReference` property (see [7.2.4.3](#)).

7.2.4.3 descriptionReference

The value of this property is a remote text description of the object. The `xlink:href` attribute of the `gml:descriptionReference` property references the external description.

```
<element name="descriptionReference" type="gml:ReferenceType"/>
```

7.2.4.4 name, identifier

The `gml:name` property provides a label or identifier for the object, commonly a descriptive name.

An object may have several names, typically assigned by different authorities. `gml:name` uses the `gml:CodeType` content model. The authority for a name is indicated by the value of its (optional) `codeSpace` attribute. The name may or may not be unique, as determined by the rules of the organization responsible for the `codeSpace`. In common usage there will be one name per authority, so a processing application may select the name from the `codeSpace` that it prefers.

```
<element name="name" type="gml:CodeType"/>
```

Often, a special identifier is assigned to an object by the authority that maintains the feature with the intention that it is used in references to the object. For such cases, the `codeSpace` shall be provided. That identifier is usually unique either globally or within an application domain. `gml:identifier` is a predefined property for such identifiers.

EXAMPLE UUIDs and URNs are commonly used globally unique identifiers.

```
<element name="identifier" type="gml:CodeWithAuthorityType"/>
```

7.2.4.5 id

The attribute `gml:id` supports provision of a handle for the XML element representing a GML object. Its use is recommended for all GML objects. In particular, all GML objects that are intended to be referenced should carry an attribute `gml:id`. For some GML object types, the attribute `gml:id` is mandatory.

```
<attribute name="id" type="ID"/>
```

It is of XML type **ID**, so is constrained to be unique in the XML document within which it occurs. An external identifier for the XML element representing the GML object in the form of a URI may be constructed using standard methods (IETF RFC 2396). This is done by concatenating the URI for the document, a fragment separator "#", and the value of the attribute of XML type ID.

7.2.5 Collections of GML objects

7.2.5.1 AbstractMemberType and derived property types

To create a collection of GML objects that are not all features, a property type shall be derived by extension from `gml:AbstractMemberType`.

```
<complexType name="AbstractMemberType" abstract="true">
  <sequence/>
  <attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The derived property type shall follow one of the patterns specified in [7.2.3](#) and may set the multiplicity of the objects in the collection as required for its intended use.

This abstract property type is intended to be used only in object types where software shall be able to identify that an instance of such an object type is to be interpreted as a collection of objects.

EXAMPLE See `gml:DictionaryEntryType` in [15.2.3](#) for such a property type.

By default, this abstract property type does not imply any ownership of the objects in the collection. The `owns` attribute of `gml:OwnershipAttributeGroup` may be used on a property element instance to assert ownership of an object in the collection. A collection shall not own an object already owned by another object.

7.2.5.2 GML object collections, `AggregationAttributeGroup`

A GML object collection is any `gml:AbstractObject` with a property element in its content model whose content model is derived by extension from `gml:AbstractMemberType`.

EXAMPLE `gml:Dictionary` is a GML object collection, because the content model of property `gml:dictionaryEntry` specified in [15.2.3](#) is derived by extension from `gml:AbstractMemberType`.

In addition, the complex type describing the content model of the GML object collection may also include a reference to the attribute group `gml:AggregationAttributeGroup` to provide additional information about the semantics of the object collection. This information may be used by applications to group GML objects, and optionally to order and index them.

```
<attributeGroup name="AggregationAttributeGroup">
  <attribute name="aggregationType" type="gml:AggregationType"/>
</attributeGroup>
```

The allowed values for the `aggregationType` attribute are defined by `gml:AggregationType`. See ISO/IEC 11404:2007, 8.4 for the meaning of the values in the enumeration.

```
<simpleType name="AggregationType" final="#all">
  <restriction base="string">
    <enumeration value="set"/>
    <enumeration value="bag"/>
    <enumeration value="sequence"/>
    <enumeration value="array"/>
    <enumeration value="record"/>
    <enumeration value="table"/>
  </restriction>
</simpleType>
```

If a collection of aggregation type “array” is implemented in an application schema, then the array type in the application schema needs to model the additional information to cope with indexing.

If a collection of aggregation type “table” is implemented in an application schema, then the table type in the application schema needs to model the additional information to add the required information about the fields and their structure.

7.2.6 Metadata

To associate metadata described by any XML Schema with a GML object, a property element shall be defined whose content model is derived by extension from `gml:AbstractMetadataPropertyType`.

The value of such a property shall be metadata. The content model of such a property type, i.e. the metadata application schema shall be specified by the GML application schema.

```
<complexType name="AbstractMetadataPropertyType" abstract="true">
  <sequence/>
```

```
<attributeGroup ref="gml:OwnershipAttributeGroup"/>
</complexType>
```

The property type derived from `gml:AbstractMetadataPropertyType` shall follow one of the patterns specified for GML property types in [7.2.3](#).

By default, this abstract property type does not imply any ownership of the metadata. The `owns` attribute of `gml:OwnershipAttributeGroup` may be used on a metadata property element instance to assert ownership of the metadata.

If metadata following the conceptual model of ISO 19115 is to be encoded in a GML document, the corresponding Implementation Specification specified in ISO/TS 19139 shall be used to encode the metadata information.

EXAMPLE 1 Assume that a feature type "Road" can be associated with two metadata elements, a data quality property "horizontalAbsoluteAccuracy" and a generic ISO/TS 19139 "metadata" property.

This may be mapped in the application schema as follows by bundling the metadata properties in a complex property:

```
<complexType name="RoadType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <!-- ... -->
        <element name="roadMetadata" type="ex:RoadMetadataPropertyType"/>
        <!-- ... -->
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<complexType name="RoadMetadataPropertyType">
  <complexContent>
    <extension base="gml:AbstractMetadataPropertyType">
      <sequence>
        <element ref="ex:RoadMetadata"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<complexType name="RoadMetadataType">
  <complexContent>
    <extension base="gmx:AbstractObjectMetadata_Type">
      <sequence>
        <element name="horizontalAbsoluteAccuracy" minOccurs="0"
          type="gmd:DQ_AbsoluteExternalPositionalAccuracy_PropertyType"/>
        <element name="metadata" minOccurs="0" type="gmd:MD_Metadata_
PropertyType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<element name="RoadMetadata" type="myAs:RoadMetadataType" substitutionGroup="gmx:AbstractO
bjectMetadata"/>
```

Then, an instance of a Road feature could look like:

```
<ex:Road>
  <!-- ... -->
  <ex:roadMetadata>
    <ex:RoadMetadata>
      <ex:horizontalAbsoluteAccuracy>
        <gmd:DQ_AbsoluteExternalPositionalAccuracy>
          <!-- The DQ_Element subelements are not detailed -->
        </gmd:DQ_AbsoluteExternalPositionalAccuracy>
      </ex:horizontalAbsoluteAccuracy>
      <ex:metadata>
        <gmd:MD_Metadata>
```

```

        <!-- a full set of ISO/TS 19139 metadata elements -->
        </gmd:MD_Metadata>
    </ex:metadata>
</ex:RoadMetadata>
</ex:roadMetadata>
<!-- ... -->
</myAs:Road>

```

An alternative encoding representing the metadata properties as separate properties of the feature would be:

```

<complexType name="RoadType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>
        <!-- ... -->
        <element name="horizontalAbsoluteAccuracy" minOccurs="0">
          <complexType>
            <complexContent>
              <extension base="gml:AbstractMetadataPropertyType">
                <sequence>
                  <element ref="gmd:DQ_AbsoluteExternalPositionalAccuracy"/>
                </sequence>
              </extension>
            </complexContent>
          </complexType>
        </element>
        <!-- ... -->
        <element name="metadata" minOccurs="0">
          <complexType>
            <complexContent>
              <extension base="gml:AbstractMetadataPropertyType">
                <sequence>
                  <element ref="gmd:MD_Metadata"/>
                </sequence>
              </extension>
            </complexContent>
          </complexType>
        </element>
        <!-- ... -->
      </sequence>
    </extension>
  </complexContent>
</complexType>

```

The instance example would then look like this:

```

<ex:Road>
  <!-- ... -->
  <ex:horizontalAbsoluteAccuracy>
    <gmd:DQ_AbsoluteExternalPositionalAccuracy>
      <!-- The DQ_Element subelements are not detailed here -->
    </gmd:DQ_AbsoluteExternalPositionalAccuracy>
  </ex:horizontalAbsoluteAccuracy>
  <!-- ... -->
  <ex:metadata>
    <gmd:MD_Metadata>
      <!-- a full set of ISO/TS 19139 metadata elements -->
    </gmd:MD_Metadata>
  </ex:metadata>
  <!-- ... -->
</ex:Road>

```

EXAMPLE 2 Assume that a dataset shall be enabled to contain Dublin Core metadata elements. This may be mapped in the application schema as follows:

```

<import namespace="http://www.purl.org/dc/terms/" schemaLocation="http://schemas.opengis.
net/csw/2.0.0/rec-dcterms.xsd"/>

<complexType name="DatasetType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      <sequence>

```

```

        <!-- ... -->
        <element name="generalMetadata" type="ex:DublinCoreMetadataPropertyType"/>
        <!-- ... -->
    </sequence>
</extension>
</complexContent>
</complexType>

<complexType name="GeneralMetadataPropertyType">
    <complexContent>
        <extension base="gml:AbstractMetadataPropertyType">
            <sequence>
                <element ref="ex:DublinCoreMetadata"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<element name="DublinCoreMetadata">
    <complexType name="DublinCoreMetadataType">
        <sequence>
            <group ref="dct:DCMI-terms" xmlns:dct="http://www.purl.org/dc/terms/" />
        </sequence>
    </complexType>
</element>

```

An instance example could look like this:

```

<ex:Dataset>
  <!-- ... -->
  <ex:generalMetadata>
    <ex:DublinCoreMetadata>
      <dc:title>Vector Smart Map Level 0</dc:title>
      <dct:abstract>Vector Map: a general purpose database design to support GIS
applications</dct:abstract>
      <dc:publisher>US National Geospatial-Intelligence Agency</dc:publisher>
      <dc:format>VPF</dc:format>
      <dc:coverage>world</dc:coverage>
      <dc:language>en</dc:language>
    <!-- ... -->
    </ex:DublinCoreMetadata>
  </ex:generalMetadata>
  <!-- ... -->
</ex:Dataset>

```

8 GML schema — Xlinks and basic types

8.1 Xlinks — Object associations and remote properties

The normative Xlink specification is available from W3C.

NOTE A schema document `xlinks.xsd` is provided as part of the GML schema documents in [Annex C](#).

Xlink components are used in GML to implement associations between objects by reference. GML property elements (see [7.2.3](#)) may carry Xlink attributes, which support the encoding of an association relationship by reference, the name of the property element denoting the target role in the association. The most important Xlink component is:

`xlink:href` identifier of the resource which is the target of the association, given as a URI

The appearance of an `xlink:href` on a GML property indicates that the value of the property shall be found by traversing the link, that is the value is pointed to by the value of the `xlink:href` attribute. Following the terminology of Xlink, GML properties with `xlink:href` attributes are sometimes referred to as remote properties.

The other Xlink components are used to indicate additional semantics of the relationship. The most useful of these are

<code>xlink:role</code>	description of the nature of the target resource, given as a URI
<code>xlink:arcrole</code>	description of the role or purpose of the target resource in relation to the present resource, given as a URI
<code>xlink:title</code>	description of the association or the target resource, given as text

For complete definitions of these and other Xlink components, including their use in extended Xlink association maps, refer to the Xlink specification.

A URI reference [URI] is defined as an optional choice between an absolute or relative URI, followed by fragment identifier that consists of a crosshatch ("#") and additional reference information. For GML object properties and remote associations, this additional reference information shall be one of the following:

- a shorthand (formerly called "barename") XPointer [XPointer Framework] consisting of the value of the `gml:id` attribute of a GML object, or
- an `element()` scheme based XPointer [XPointer `element()`], or
- an `xpointer()` scheme based XPointer [XPointer `xpointer()`] containing an XPath [XPath] expression that selects a GML object, optionally preceded by one or more `xmlns()` scheme based XPointer(s) [XPointer `xmlns()`] that define the namespace prefixes used in the XPath expression.

A URI that does not contain an absolute or relative URI, but that consists entirely of a fragment identifier, refers to a GML object elsewhere in the same GML document.

Absolute and relative URIs may include a query component that consists of a question mark ("?") followed by a query to be interpreted by the resource. For GML object properties and remote associations, any such query shall be a request to a service that returns a GML object. The URI containing such a query may or may not make use of a fragment identifier, depending on the request syntax defined by the service.

In the GML schema, simple Xlinks are used exclusively to denote association roles of GML objects and to denote remotely referenced property values.

EXAMPLE 1 A reference to an object element in the same GML document can be encoded as:

```
<myProperty xlink:href="#o1"/>
```

EXAMPLE 2 A reference to an object element in a remote XML document using the `gml:id` value of that object can be encoded as:

```
<myProperty xlink:href="http://my.big.org/test.xml#o1"/>
```

EXAMPLE 3 A reference to an object element in a remote XML document (or GML object repository) using the `gml:identifier` property value of that object can be encoded as:

```
<myProperty xlink:href="http://my.big.org/test.xml#element (//gml:GeodeticCRS[.//gml:identifier[@codeSpace="http://www.opengis.net/def/crs/EPSG/0/"]="4326"])" />
```

EXAMPLE 4 A reference to an object element with a uniform resource name can be encoded as follows (note that a URN resolver is required to resolve the URN and access the referenced object):

```
<myProperty xlink:href="urn:ogc:def:crs:EPSG::4326"/>
```

The IDREF data type and the unique, key, and keyref elements defined in the XML and XML Schema specifications provide alternative identification and linking mechanisms to the ID data type and Xlink reference for use within a single XML document. Although these XML components may be used in XML Schemas, they have no normative role in GML, and shall not be used to denote association roles of GML objects or remotely referenced property values.

8.2 Basic types

8.2.1 Overview

W3C XML Schema provides a set of built-in “simple” types which define methods for representing values as literals without internal markup. These are described in W3C XML Schema Part 2:2001. Because GML is an XML encoding in which instances are described using XML Schema, these simple types shall be used as far as possible and practical for the representation of data types. W3C XML Schema also provides methods for defining

- new simple types by restriction and combination of the built-in types, and
- complex types, with simple content, but which also have XML attributes.

In many places where a suitable built-in simple type is not available, simple content types derived using the XML Schema mechanisms are used for the representation of data types in GML.

A set of these simple content types that are required by several GML components are defined in the basicTypes schema, as well as some elements based on them. These are primarily based around components needed to record amounts, counts, flags and terms, together with support for exceptions or null values.

NOTE The basic types and elements are described in the basicTypes schema document in [Annex C](#). The schema is identified by the following location-independent name (using URN syntax):

- urn:ogc:specification:gml:schema-xsd:basicTypes:3.2.1

8.2.2 Relationship with ISO 19103

ISO 19103 defines basic types for the conceptual schemas in the ISO 19100 series of International Standards. GML implements a subset of these basic types as described in [D.2.2](#).

NOTE Some of the ISO 19103 basic types are specified in other schema documents of the GML schema: units of measure are specified in [Clause 16](#) and vector in [10.1.4.5](#).

8.2.3 Simple types

8.2.3.1 NilReasonType

`gml:NilReasonType` defines a content model that allows recording of an explanation for a void value or other exception.

```
<simpleType name="NilReasonEnumeration">
  <union>
    <simpleType>
      <restriction base="string">
        <enumeration value="inapplicable"/>
        <enumeration value="missing"/>
        <enumeration value="template"/>
        <enumeration value="unknown"/>
        <enumeration value="withheld"/>
      </restriction>
    </simpleType>
    <simpleType>
      <restriction base="string">
        <pattern value="other:\w{2,}"/>
      </restriction>
    </simpleType>
  </union>
</simpleType>

<simpleType name="NilReasonType">
  <union memberTypes="gml:NilReasonEnumeration anyURI"/>
</simpleType>
```

`gml:NilReasonType` is a union of the following enumerated values:

- **"inapplicable"**: there is no value
- **"missing"**: the correct value is not readily available to the sender of this data. Furthermore, a correct value may not exist
- **"template"**: the value will be available later
- **"unknown"**: the correct value is not known to, and not computable by, the sender of this data. However, a correct value probably exists
- **"withheld"**: the value is not divulged
- **"other:"+text**: other brief explanation, where **text** is a string of two or more characters with no included spaces

and

- **anyURI** which should refer to a resource which describes the reason for the exception

A particular community may choose to assign more detailed semantics to the standard values provided. Alternatively, the URI method enables a specific or more complete explanation for the absence of a value to be provided elsewhere and indicated by-reference in an instance document.

`gml:NilReasonType` is used as a member of a union in a number of simple content types defined below (see [8.2.3.4](#), [8.2.4.1](#), [8.2.4.2](#), [8.2.4.3](#)) where it is necessary to permit a value from the `NilReasonType` union as an alternative to the primary type.

8.2.3.2 Elements declared to be “nillable”

The XML Schema attribute `nillable` may be included in any element declaration within a schema.

NOTE 1 By default the schema attribute `nillable` has a value of “false”.

EXAMPLE 1 The following element declarations illustrate the use of the `nillable` attribute:

```
<element name="amount" type="double" nillable="true"/>
<element ref="my:amount" nillable="true"/>
```

By declaring an element as nillable (`nillable="true"`), an instance of that element may omit its content in cases where an empty value would normally not be schema valid by supplying an attribute `nil` from the XML Schema Instance namespace with the value “true”.

EXAMPLE 2 Elements that have been declared with this `nillable="true"` in the schema can appear in instance documents as follows:

```
<my:amount>34.567</my:amount>
<my:amount xsi:nil="true"/>
```

Declaring an element to be nil is an implementation of the “Void” data type of ISO/IEC 11404, i.e. represents “an object whose presence is syntactically or semantically required, but carries no information in a given instance” [ISO/IEC 11404].

NOTE 2 This is different to an element declaration with the cardinality attribute set to make the element optional, such as:

```
<element name="amount" type="double" minOccurs="0"/>
```

which allows the element to be omitted in the instance entirely.

In some situations where it is required to declare an element in an application schema nillable, it may be convenient to also add an attribute of type `gml:NilReasonType`.

EXAMPLE 3 The application schema components