

Open Geospatial Consortium

Submission Date: 2015-02-12

Approval Date: 2015-03-13

Publication Date: 2016-03-24

External identifier of this OGC® document: <http://www.opengis.net/doc/DP/gml-aviation-guidance/r1>

Internal reference number of this OGC® document: 12-028r1

Category: OGC® Discussion Paper

Editor: OGC Aviation Domain Working Group

Use of Geography Markup Language (GML) for Aviation Data

Copyright notice

Copyright © 2016 Open Geospatial Consortium

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Document type: OGC® Discussion Paper
Document subtype:
Document stage: Approved for Public Release
Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Contents

1. Scope	12
2. Conformance	12
3. References	12
4. Terms and Definitions	12
4.1. data set	13
4.2. right-handed CRS	13
4.3. left-handed CRS	13
5. Conventions.....	13
5.1. Abbreviated terms.....	13
6. Coordinate reference systems.....	15
6.1. Geographic vs geometric data	15
6.2. CRS and srsName	16
6.3. WGS-84	17
6.4. Use of global srsName.....	18
7. Positions	20
7.1. Background.....	20
7.2. GML encoding.....	20
8. Lines and Surfaces	21
8.1. Background.....	21
8.2. GML encoding.....	21
8.2.1. Straight lines.....	21
8.2.2. Parallels	22
8.2.3. Arc by edge	24
8.2.4. Arc by centre point.....	25
8.2.5. Circle by center point	30

8.2.6.	Perimeter encoding direction	31
8.2.7.	Corridor	32
8.2.8.	Other geometries	33
9.	Airspace aggregation.....	35
9.1.	Background.....	35
9.2.	GML encoding.....	35
9.2.1.	By reference	35
9.2.2.	By copying the geometry	38
9.2.3.	Combined method	40
10.	Point references and annotations.....	41
10.1.	Background	41
10.2.	GML encoding.....	42
10.2.1.	Using aixm:Point annotations.....	42
10.2.2.	Using xlink:href.....	43
10.2.3.	Summary.....	44
11.	Geographical border references	45
11.1.	Background	45
11.2.	GML encoding.....	46
11.2.1.	Using aixm:Curve annotations	46
11.2.2.	Using xlink:href.....	48
12.	AIXM GML Profile	54
12.1.	Introduction.....	54
12.2.	Purpose of the profile.....	54
12.3.	Scope of the profile	54
12.4.	Using the AIXM GML Profile.....	55
12.5.	XML Namespaces.....	55
12.6.	AIXM GML Profile Requirements.....	55

12.6.1.	Basic Types.....	55
12.6.2.	AIXM Types.....	57
12.6.3.	ISO 19107 and ISO 19136 (GML) Types - Overview.....	57
12.6.4.	ISO 19108 and ISO 19136 (GML) Types.....	61
12.6.5.	Use of xsi:type is forbidden.....	63
12.7.	AIXM GML Profile Definition.....	65
Annex A -	List of CRS used for aeronautical data.....	67
	Left-handed CRS.....	67
	Right-handed CRS.....	68
Annex B -	ArcByCenterPoint Interpretation Summary.....	69
	Introduction.....	69
	Angle Measuring Convention in GML.....	69
	General Direction of Increasing and Decreasing Angle Values.....	69
	Angle Measurement in Different Coordinate Systems.....	70
	Direction of an Arc.....	71
	Arc encoding examples in a left-handed CRS system.....	72
	Arc interpolation.....	76
Annex C -	Mapping for ArcByCenterPoint.....	77
Annex D -	Perimeter encoding direction considerations.....	80
Annex E –	Offset Curve and Airspace Corridor encoding issues.....	82
Annex F –	Considerations about interpolations.....	85
	Role of interpolation.....	85
	Why geodesic is better.....	85
	Practical considerations.....	86
	Interpolation and Densification Considerations.....	86
	Background.....	86
	Comparison of models.....	87

Mapping of GML to Simple Geometry	88
Flattening of geometry structure	88
Densification of curves.....	88
Loss of data structure	89
Annex G - GML Profile - XML Schema Implementation.....	90
AIXM Point – Documentation	90
AIXM ElevatedPoint – Documentation	91
AIXM Curve – Documentation.....	92
AIXM ElevatedCurve – Documentation.....	93
AIXM Surface – Documentation	95
AIXM ElevatedSurface – Documentation	96
DirectPosition / gml:pos – Documentation	98
GM_Object / gml:AbstractGeometry – Documentation	98
GM_Point / gml:Point – Documentation	100
GM_Envelope / gml:Envelope – Documentation	101
GM_PointRef / gml:pointProperty – Documentation	102
GM_Position / gml:geometricPositionGroup – Documentation	102
GM_PointArray / gml:posList – Documentation.....	103
gml:AbstractCurve – Documentation.....	104
GM_Curve / gml:Curve – Documentation.....	105
GM_CurveSegment / gml:AbstractCurveSegment – Documentation	106
gml:ArcByCenterPoint – Documentation	107
gml:CircleByCenterPoint – Documentation	109
GM_Arc / gml:Arc – Documentation	111
GM_Circle / gml:Circle – Documentation.....	112
GM_GeodesicString / gml:GeodesicString – Documentation	113
GM_Geodesic / gml:Geodesic – Documentation.....	114

GM_LineString / gml:LineStringSegment – Documentation	115
GM_Surface / gml:Surface – Documentation	116
GM_SurfacePatch / gml:AbstractSurfacePatch – Documentation.....	117
GM_Polygon / gml:PolygonPatch – Documentation.....	118
gml:AbstractRing – Documentation.....	119
GM_Ring / gml:Ring – Documentation.....	119
GM_OrientableCurve / gml:OrientableCurve – Documentation	121
GM_CompositeCurve / gml:CompositeCurve – Documentation	123
TM_GeometricPrimitive / gml:AbstractTimeGeometricPrimitive – Documentation	124
TM_Instant / gml:TimeInstant – Documentation.....	125
TM_Period / gml:TimePeriod – Documentation	126
GML Simple Types	127
Deprecated GML 3.2.1 items	127
Annex G - Bibliography	128

i. Abstract

Starting with version 5, the Aeronautical Information Exchange Model (AIXM) schema began using the OGC Geographical Markup Language (GML) version 3.2.1 for the encoding of positional and shape data of aeronautical information items, such as airspace, runway thresholds, nav aids, etc.

The ISO 19107 spatial schema, which is implemented in GML, is very complex. ISO 19107 contains an extensive list of geometries, geometric properties and operations – many of which are not necessary for aeronautical information applications. In addition, ISO 19107 contains an exhaustive 3D geometry model that is probably not needed in its entirety for AIXM either. Therefore, a profile of GML for AIXM needs to be defined.

The objective of this document is to identify the elements of the AIXM-GML profile and to provide guidelines for the use of GML constructs in AIXM data sets.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, GML, AIXM, aviation profile

iii. Preface

According to the ICAO¹ rules, Aeronautical Information (AI) is published by States using paper (and increasingly electronic) documents, such as Aeronautical Information Publications (AIP), charts, manuals. These data includes geographically related information items such as:

- Positions expressed in latitude/longitude, which according to ICAO Annex 15 shall use a WGS-84 datum;
- Shapes of airspace, expressed as a series of positions in combination with arcs of circles or as full circles. Sometimes, these shapes contain references to national borders, water courses, etc., which are not provided explicitly in the AIP;
- Shapes of obstacles, provided as point, line or polygon, again using a series of positions and arcs of circle.

Since 2003, the Aeronautical Information Exchange Model (AIXM) has been used for the provision of AI in digital format. Initially developed for the European AIS Database, AIXM was progressively adopted by other States world-wide. From the International Civil Aviation

¹ International Civil Aviation Organization. <http://www.icao.int/Pages/default.aspx>

Organisation (ICAO) perspective, AIXM is positioned as a means of achieving compliance with the standards and recommended practices for digital aeronautical data exchange, as stated in the ICAO Annex 15.

AIXM versions up to 4.5 used a custom XML encoding and did not use OGC standards. Beginning in 2008 with the publication of version 5, the AIXM specification has embraced GML as the data encoding format.

There are a number of specific requirements in the AI Domain that concern the provision of geographical and geometrical information. These requirements are discussed in this document and includes recommendations for data encoding in GML.

The first version of this document was developed by the OGC Aviation Domain Working Group and published as an OGC Discussion Paper (OGC document reference 12-028) in May 2012. This version is an update of the original version and based on the feedback from users and the results for OGC testbed activities provides further guidance and clarifications.

The objective is the publication of this document as an OGC Best Practice document by mid-2015

iv. Forward

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

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

v. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Avitech AG
- COCESNA
- Comsoft
- EgisAvia
- Erdas
- EUROCONTROL
- Galdos, Inc.
- IDS Ingegneria Dei Sistemi S.p.A.
- Interactive Instruments
- Luciad
- M-click
- Snowflake Software
- SOLITEC Software Solutions GesmbH
- Thales

vi. Submitters

All questions regarding this submission should be directed to the editor or the contributors:

Name	Role	Organization
Daniel Balog	Contributor	Luciad
Manfred Beckmann	Contributor	SOLITEC Software Solutions GesmbH
David Burggraf	Contributor	Galdos, Inc.
Eddie Curtis	Contributor	Snowflake Software
Warwick Dufour	Contributor	Avitech AG
Johannes Echterhoff	Editor	interactive instruments
Yves Ernotte	Contributor	Thales
Davide Castagni Fabbri	Contributor	IDS Ingegneria Dei Sistemi S.p.A
Benoit Geffroy	Contributor	EgisAvia
Francois Germain	Contributor	Thales
Volker Grabsch	Contributor	M-Click
Razvan Guleac	Contributor	EUROCONTROL
Robin Houtmeyers	Contributor	Luciad
Alain Kabamba	Contributor	Erdas
Michal Kadlec	Contributor	Avitech AG
Antonio Locandro	Contributor	COCESNA

Ian Painter	Contributor	Snowflake Software
Eduard Porosnicu	Editor	EUROCONTROL
Bert Robben	Contributor	Luciad
Timo Thomas	Contributor	Comsoft
Scott Wilson	Contributor	EUROCONTROL
Yvonne Zannoun	Contributor	Snowflake Software

1. Scope

The document provides guidelines for the use of GML and a GML profile description in the scope of aeronautical data encoding, in particular when using the Aeronautical Information Exchange Model (AIXM). In the future, the applicability of the guidelines contained in this document might be enlarged to cover other related domains, such as aeronautical weather data and flight data.

2. Conformance

This document is aimed at becoming an OGC Best Practice. Compliance with this best practice implies compliance with all the "shall" statements contained in this document.

For a particular AIXM data set, conformance with the provisions of this document shall be declared through the inclusion of the following annotation in the AIXM Schema file:

```
<annotation>
  <appinfo>
    <gml:gmlProfileSchema>http://www.aixm.aero/schema/GML_profile/gml321forAIXM.xsd
    </gml:gmlProfileSchema>
  </appinfo>
</annotation>
```

3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- ISO 19107:2003 – Geographic information — Spatial schema
- ISO 19136:2007 – Geography Markup Language
- OGC 07-092r3 - [Definition identifier URNs in OGC namespace](http://portal.opengeospatial.org/files/?artifact_id=30575)
- OGC 06-042 - [OpenGIS® Web Map Server Implementation Specification](http://portal.opengeospatial.org/files/?artifact_id=14416)
- ISO 19108:2002 - Geographic information - Temporal schema

4. Terms and Definitions

This document uses the standard terms defined in Sub-clause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2 - Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this OGC best practice.

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

4.1. data set

Means an identifiable collection of data.

4.2. right-handed CRS

The name derives from the right-hand rule (mathematics and physics). If the index finger of the right hand is pointed forward, the middle finger bent inward at a right angle to it, and the thumb placed at a right angle to both, the three fingers indicate the relative directions of the x-, y-, and z-axes in a right-handed system. The thumb indicates the x-axis, the index finger the y-axis and the middle finger the z-axis. Conversely, if the same is done with the left hand, a left-handed system results.

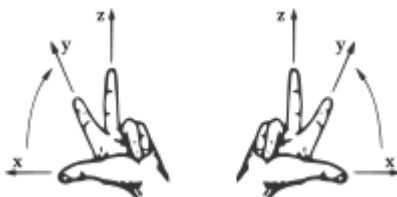


Figure 1 - Left-handed and right-handed systems

When applied to a geodetic Coordinate Reference System (CRS) for which the z axis points from the centre of the Earth outwards, this implies that right-handed systems will have longitude (East) as first axis and latitude (North) as second axis. This is not the usual aviation convention, as latitude is usually used in aviation as first axis and angles/bearings are measured clockwise towards East (the second axis).

4.3. left-handed CRS

See the explanations for right-handed CRS above. Left-handed CRS are the natural choice for the aeronautical data domain.

5. Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Abbreviated terms

Table 1 - abbreviations

AIP	Aeronautical Information Publications
AIXM	Aeronautical Information Exchange Model
CRS	Coordinate Reference System
ECEF	Earth Centered Earth Fixed

EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GML	Geographical Markup Language
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
OPADD	Operating Procedures for Aeronautical Dynamic Data
UCUM	Unified Code of Units of Measure
WFS	Web Feature Service

6. Coordinate reference systems

6.1. Geographic vs geometric data

As in many other classes of applications, aeronautical applications deal with entities which have a geographic extent. By definition, software systems supporting aeronautical applications are Geographic Information Systems (GIS). Within this context, distinguishing between the adjectives “geometric” and “geographic” is worthwhile.

Geometric entities are point sets in a metric space, namely a topological space endowed with a metric. A metric is nothing but a set of rules for measuring space properties such as distances, angles, volumes and so on. A well known example of metric space is n-dimensional Euclidean space E^n , namely the real vector space R^n endowed with the usual Euclidean metric (where the distance between any two points is given by the Pythagorean formula applied to the points’ coordinates).

Geographic entities are geometric entities belonging not to a generic, abstract metric space but to the Euclidean 3D space E^3 surrounding the Earth, where the metric can be operatively defined by means of the usual measuring processes, once a length unit of measure has been defined, such as “meters”. This metric is the Euclidean-Pythagorean one for Earth Centered Earth Fixed (ECEF) coordinates.

Given that a great part of the physical phenomena relevant for GIS applications is restricted to a thin layer of space surrounding the Earth surface, it is often useful to adopt a Coordinate Reference System (CRS) different from ECEF. Therefore, in many applications, including those in the aeronautical family, it is common to adopt a CRS in which the first two coordinates parameterize the Earth surface while the third coordinate parameterizes the orthogonal axis emanating from the surface. The third coordinate is called altitude or height, depending on the zero reference point (e.g. ellipsoid, geoid or terrain).

In many applications the horizontal aspect of geographic entities is far more relevant than the vertical aspect. Hence geographic entities are simply described as 2D geometric objects: the orthogonal projection of 3D entities onto the Earth surface.

However, there is a subtlety: 2D geographic entities exist in a curved world, the Earth’s surface. Curved means that the metric we adopt to measure distances, angles and areas on that surface cannot be mapped back to the Euclidean metric on R^2 (mathematicians say that “a curved surface is not isomorphic to the flat space E^2 ”) In other words, we cannot find any CRS parameterizing the surface where the distance between any couple of points can be calculated using the Pythagorean formula. This fact has important repercussions on the whole set of geometric concepts we use to describe reality, including the language we adopt (and hence on GML itself).

6.2. CRS and srsName

In GML, the geodetic datum is specified by reference to a Coordinate Reference System (CRS). A CRS relates a coordinate system to the Earth by a datum. A geodetic datum consists of an ellipsoid model and a prime meridian. The intersection of the equator and prime meridian is the origin of the CRS.

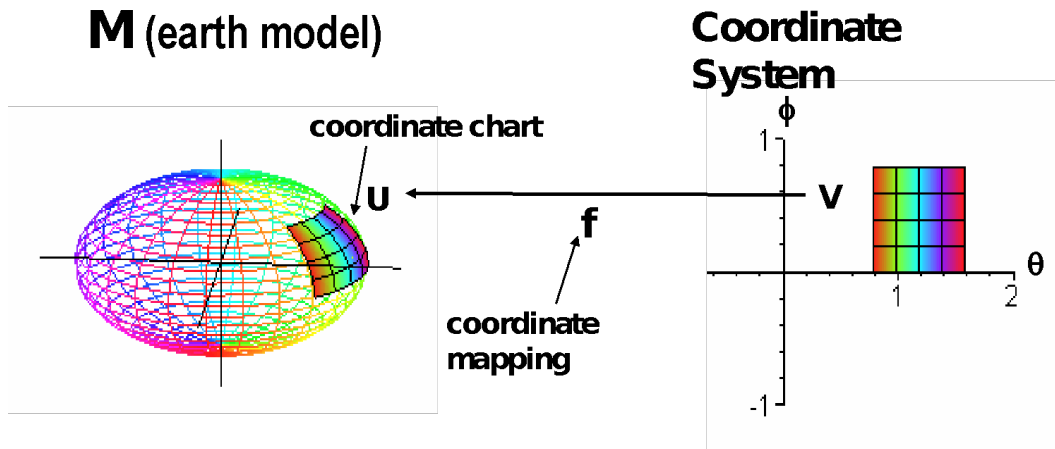


Figure 2 - ECEF coordinate system

A geodetic CRS (e.g. EPSG 4326) relates a (lat/long) ellipsoidal coordinate system to the Earth.

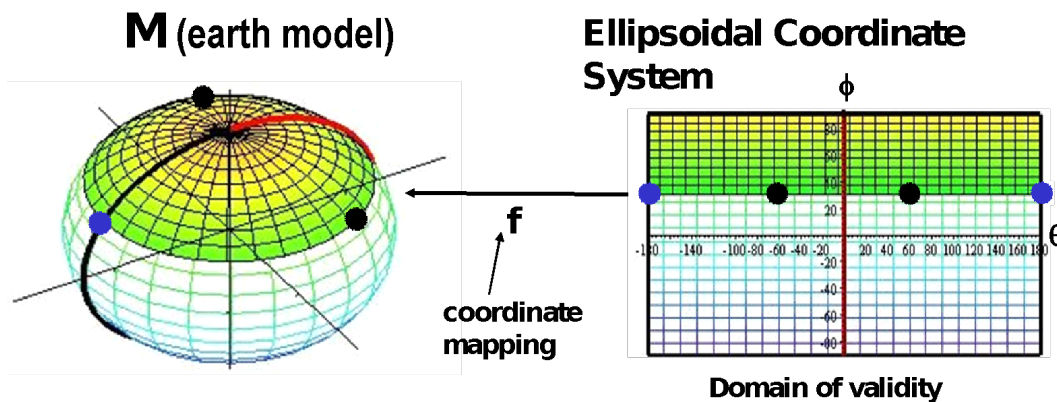


Figure 3 - Ellipsoidal coordinate system

The *CRS reference is critical for the correct encoding and processing* of the geographical data contained in AIXM/GML files. The CRS indicates not only the geodetic reference datum, but also the order of the coordinate axes (latitude/longitude or longitude/latitude) and has important implications for the convention used for measuring angles (from the North, clockwise, for example). This will be discussed in more detail later in this document.

There are two main CRS definition authorities that are relevant for the AI domain: Oil and Gas Producers (OGP), formerly known as the European Petroleum Survey Group (EPSG) and the

OGC. The two sets of CRS definitions have many common points. For example, the OGC:CRS84 is a variant of EPSG:4326 (differing only in its coordinate order: longitude/latitude) and is defined in the ISO 19128 Geographic information — Web Map Server standard. The EPSG CRS database is available at <http://www.epsg.org> - European Petroleum Survey Group Geodesy Parameters [EPSG CRS].

Recommendations for the use of CRS references in GML data sets are provided in the OGC Recommendation Paper “URNs of definitions in ogc namespace” [OGC 07-092r3]. The CRS of an AIXM geometry is identified by a URN (e.g. urn:ogc:def:crs:EPSG::4326) and defined in its srsName attribute or derived from the larger context that the geometry is part of.

When applied to the encoding of a surface in AIXM, this will give the following GML element:

```
<aixm:Surface gml:id="S01" srsName="urn:ogc:def:crs:EPSG::4326">
```

Specifying the srsDimension attribute is not required. This is because it is implicit in the srsName. Specifying the srsDimension could lead to discrepancies, such as using srsName="epsg:4326" and srsDimension="3". One could think that this is a good way to describe 3D WGS84 coordinates. However, this assumption is wrong and an appropriate 3D srsName should be used in that case, such as EPSG::4979.

6.3. WGS-84

According to ICAO Annex 15, all “*published aeronautical geographical coordinates (indicating latitude and longitude) shall be expressed in terms of the WGS-84 geodetic reference datum*”.

The Coordinate Reference System (CRS) reference is critical for the correct encoding and processing of AIXM/GML geometries. This is because a CRS not only indicates the geodetic datum and ellipsoid for which point coordinates are expressed but also the order of the coordinate axes in which coordinate values are provided, *e.g. latitude before longitude – which is an important convention for the aviation domain*.

Due to the way that angle directions are traditionally measured in the AI domain (North corresponds to 0°, East to 90°, etc.), the use of the OGC:CRS 84 is not straight-forward for AIXM 5.1/GML data sets that contain arcs of circle defined by start angle/end angle measured from the North. This will be explained in section "[Measuring angles in GML](#)" of this document. Therefore, *the EPSG:4326 CRS is the typical choice for AIXM 5.1 data sets* that use the WGS-84 reference datum. However, this does not exclude the use of other CRS when appropriate.

When encoding aeronautical data that complies with the WGS-84 ICAO Standard, the following Coordinate Reference System (CRS) *shall* be used in AIXM 5.1:

- EPSG:4326 - for data conforming to the usual aviation practice (latitude first, longitude second, angles/bearings measured from the North with positive values clockwise);
- OGC:CRS84 - for data conforming to the more “mathematical” practice (longitude as first axis, latitude as second axis, angles measured from the East with positive values counter-clockwise).

When encoding aeronautical data that does not comply with the WGS-84 ICAO Standard an appropriate CRS *shall* be used. A list of CRS that are likely to be used in aeronautical data sets is provided in Annex A of this document. The use of other CRS falls outside the scope of this document.

Geographic coordinates(latitude and longitude) *shall* be expressed in decimal degrees, not in degrees minutes seconds.

6.4. Use of global srsName

For all geometries (Surface, ElevatedPoint, etc.) a CRS *shall* be specified. The CRS is either defined directly on the geometry element using the srsName attribute or is derived from the larger context the geometry is part of.

Unless overruled by the presence of a local srsName for convenience in constructing feature and feature collection instances, **the value of the srsName attribute on the gml:Envelope shall be inherited by all directly expressed geometries in all properties of the feature or members of the collection.** As indicated in Figure 4, the gml:Envelope is a child element of the gml:boundedBy property of the feature. If a geometry uses the same coordinate reference system as given on the gml:boundedBy property of its parent feature that geometry does not require a srsName attribute,

Inheritance of the coordinate reference system continues to any depth of nesting. However, if overruled by a local srsName declaration, then the new coordinate reference system is inherited by all its children in turn.

Notwithstanding this rule, ***all the geometries used in a feature or feature collection may carry srsName attributes.*** This is in order to indicate a local reference system, even if they are the same as the parent. A geometry without a srsName derives its CRS from its closest ancestor that has a srsName. Due to the way that AIXM is modelled, this can only be one of the following:

- aixm:Surface
- aixm:Curve
- aixm:Point

If no such ancestor is found, that geometry without srsName derives its CRS from the srsName of the gml:Envelope in the boundedBy element of the feature or feature collection in which the geometry is contained. Geometries for which no CRS can be derived are invalid.

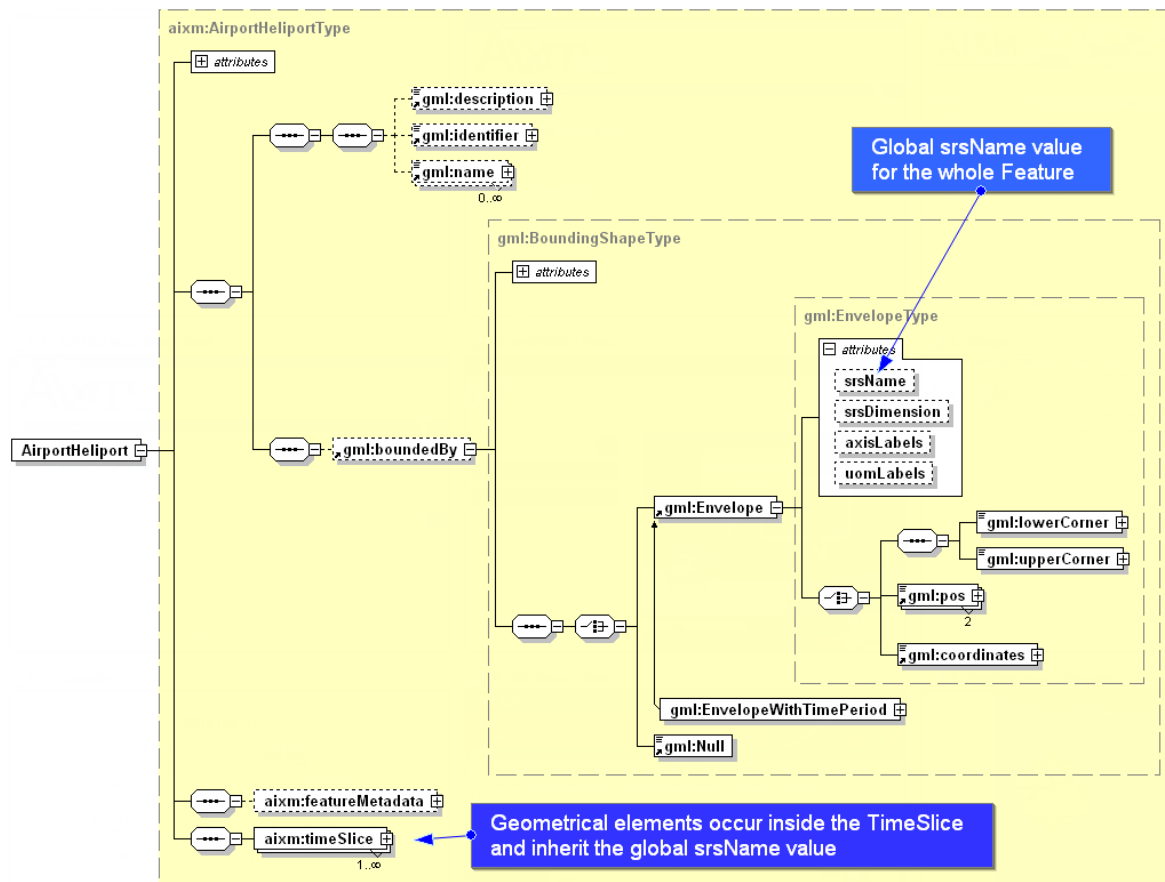


Figure 4 - Global srsName specified using gml:Envelope

7. Positions

7.1. Background

Simple positions are used in order to indicate the geographical location of an airport reference point, navaid, waypoint, runway threshold, etc. In AI publications, simple positions are expressed as a pair of latitude/longitude coordinates. Usually the information about the geodetic reference datum is specified once for the whole AIP and not provided for each individual position.

7.2. GML encoding

In AIXM 5.1, simple positions are encoded using the `aixm:Point` or `aixm:ElevatedPoint` elements, which are extensions of the `gml:Point`:

```
...
<aixm:ElevatedPoint srsName="urn:ogc:def:crs:EPSG::4326" gml:id="ID55">
  <gml:pos>52.2889 -32.0350</gml:pos>
</aixm:ElevatedPoint>
...
```

Note that the EPSG:4326 CRS has latitude as the primary axis, which indicates that the order of the values in the `gml:pos` element is ***first latitude, then longitude***. This convention is very important for the correct interpretation of the data and it is not the same for all CRS! For example, the OGC:CRS84 has longitude as the primary axis. Therefore in a GML data set that use the OGC:CRS84 the first value in the `gml:pos` element will indicate the longitude!

The convention “first latitude, then longitude” is widely observed in the AI domain. Not surprisingly, the OGC WMS standard has a note that reads: “*users in the international aviation and marine sectors may expect latitude to be before longitude, and a different coordinate display may have safety implications, especially in an emergency response situation*” and “*developers of user interfaces for WMSs are cautioned that all references to latitude and longitude, for example user input of bounding box or readout of cursor coordinates, should show latitude before longitude*”.

For GML encoding, the latitude/longitude data needs to be expressed in numerical (degrees with decimals) format. If this is not the case with the originator data, a data format conversion should take place. This conversion should be done with a sufficient number of decimal values in order to preserve the original precision of the data. In order to avoid the introduction of small imprecision due to such format conversions, data originators shall be asked to send the data in the raw numerical format (degrees with decimals) that is typically used for survey and geodetic calculations.

8. Lines and Surfaces

8.1. Background

Certain features in the AI domain have polygonal shapes (such as Airspace) or linear shapes (such as a power line Vertical Structure). They are typically published (for example, in Aeronautical Information Publications – AIP) as a series of latitude/longitude positions, such as in the following example:

EAP 25 (The Castle)
 521108.00N 0051230.00E;
 521222.00N 0051715.00E;
 521121.00N 0051756.00E;
 521009.00N 0051756.00E;
 (then along the parallel to) 521009.00N 0051311.00E;
 to point of origin.

Usually, the interpolation method used for the curve between the consecutive points it is not indicated in the AI source documents, but it is generally assumed that:

- If two consecutive points have the same latitude value, then the line connecting the two points is a parallel on the surface of the Earth; this may be explicitly stated using words such as “along the parallel to”;
- Otherwise, it is considered a “straight line on the map”.

In addition, the source map used when the airspace was designed is typically unknown.

Arcs of circle are also used in the definition of airspace borders, such as in the following examples:

EHR 4A (VLIEHORS) TSA
 531012.59N 0044621.14E; *along clockwise arc (radius 8 NM, centre 531500.00N 0045700.00E) to 530701.98N 0045602.41E;*
 531100.00N 0045124.00E; *to point of origin.*

EHR 4B (VLIEHORS)
 530943.06N 0050658.79E; 530240.00N 0051500.00E; 525809.00N 0050622.00E; 530701.98N 0045602.41E; *along anti-clockwise arc (radius 8 NM, centre 531500.00N 0045700.00E) to point of origin.*

Arcs may also be used in the definition of approach/departure trajectories. However, specific “path and terminator” codes are used to encode such arcs and the use of GML in this case is limited to providing a curve for printing the procedure on a map. GML is not used to encode the real flight trajectory of an aircraft, as stored in the Flight Management System (FMS).

8.2. GML encoding

8.2.1. Straight lines

The gml:GeodesicString *shall* be used as default encoding for straight lines. The reasons are:

- A geodesic interpolation is the mathematical generalization of the notion of “straight line” on the surface of the Earth;
- The result of the interpolation on the surface of the Earth does not depend on the CRS used; by contrast, a linear interpolation would result on different curves on the surface of the Earth, depending on the CRS used (in fact, this will be exploited in order to encode lines along a parallel, see further down).

Surfaces are encoded in GML using `gml:PolygonPatch` elements. The pairs of lat/long coordinates can be encoded as either a sequence of `gml:pos` or, more compact, using a `gml:posList` element:

```
...
  <aixm:Surface gml:id="S01" srsName="urn:ogc:def:crs:EPSG::4326">
    <gml:patches>
      <gml:PolygonPatch>
        <gml:exterior>
          <gml:Ring>
            <gml:curveMember>
              <gml:Curve gml:id="C001">
                <gml:segments>
                  <gml:GeodesicString>
                    <gml:posList>52.18556 5.20833 52.20611 5.2875 52.18917
                    5.29889 52.16917 5.29889 52.18556 5.20833</gml:posList>
                  </gml:GeodesicString>
                </gml:segments>
              </gml:Curve>
            </gml:curveMember>
          </gml:Ring>
        </gml:exterior>
      </gml:PolygonPatch>
    </gml:patches>
  </aixm:Surface>
...

```

Note that the first latitude/longitude pair in this `posList` example is equal to the last one. As stated in section 10.5.11.1 of the GML Standard: “*Every `gml:curveMember` references or contains one curve, i.e. any element which is substitutable for `gml:AbstractCurve`. In the context of a ring, the curves describe the boundary of the surface. The sequence of curves shall be contiguous and connected in a cycle*”. In the special case that there is only one curve member in a `gml:Ring`, this means that the curve member itself needs to form a cycle, thus the need for the last position to be equal with the first one. Note that in GML 3.3 there are new compact encodings for geometry primitives, such as `SimplePolygon` that do not require the repeated last coordinate. However, these are not available yet in AIXM 5.1, which uses GML 3.2.1.

Also note that the same separator (space) is used both between the latitude and longitude values (coordinate separator) and also between the latitude/longitude groups (tuple separator).

8.2.2. Parallels

In the AI domain, if an Airspace border has two consecutive points at the same geographical latitude, it is assumed that the line between the two points is “along the parallel”. This shall be encoded in AIXM/GML using “linear” GML elements in combination with a geodetic CRS,

such as EPSG:4326. The **linear interpolation** in a 2D geodetic CRS between two points that have the same latitude corresponds to a parallel on the Earth's surface. This is shown graphically in Figure 5.

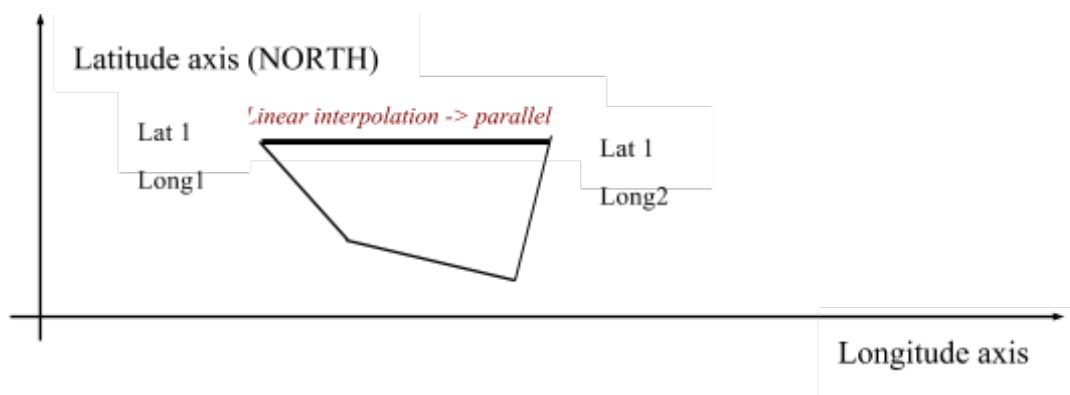


Figure 5 - Linear interpolation in a geodetic CRS

Note that the current GML 3.2.1 does not allow general “rhumbline” (constant angle with the meridians) interpolations to be specified directly (e.g. using a “RhumbLine” element or a “rhumbline” interpolation). However, linear interpolations in certain conformal projections do correspond to rhumbines on the ellipsoid Earth model.

For example, a LineStringSegment with two coordinates (i.e. a line segment with begin and end point) may be used with a srsName that references a Mercator projection (e.g. EPSG:3395), which is a well supported conformal projection. Note that the LineStringSegment element implies that linear interpolation must be used and srsName=”urn:ogc:def:crs:EPSG::3395” implies that the interpolation is done in the Mercator projection plane. Hence this geometry gets realized as a rhumbline on the WGS84 ellipsoid Earth model. Other conformal stereographic and conical projections can also be used to represent rhumbines on the earth ellipsoid model of WGS84 (this may be useful for regions of interest near the poles).

In conclusion:

- in the classical aeronautical information case of two consecutive points having the same latitude, a gml:LineStringSegment with "default" CRS EPSG:4326 shall be used for encoding. This is the case for most Airspace data published by States.
- in the particular case where one wants to express a rhumbline whereas the two consecutive latitudes are different, a gml:LineStringSegment with a "Mercator" CRS like EPSG:3395 may be used. However, this is a theoretical discussion since no real world aeronautical data like is known to require the use of arbitrary rhumbines.

Note: If srsName=EPSG:3395 is specified for the line and nothing is specified for the points defining the line then the inheritance rule applies. The points shall be expressed in the EPSG:3395 coordinates which are meters in the Mercator projection.

8.2.3. Arc by edge

This is a relatively simple case as it is represented by the element `gml:Arc` in GML, which does not have any ambiguities: 3 points always define a single arc. Unfortunately, this is rarely used in the AI domain. *When moving towards a fully digital AI chain, the use of this type of arc information should be encouraged and eventually imposed as the unique way for defining arcs.* However, this is not likely to be achieved on short term.

A border that uses arcs by 3 points looks like in Figure 6 - Arc by edge point:

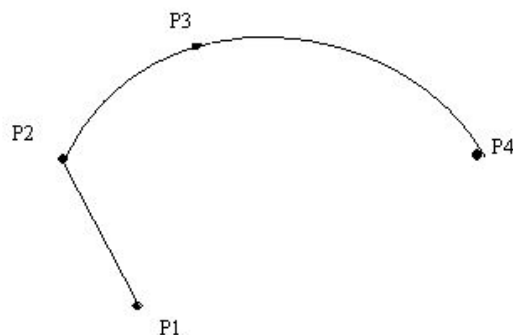


Figure 6 - Arc by edge point

A GML encoding example for this type of arcs is provided below.

```
...
<gml:PolygonPatch>
  <gml:exterior>
    <gml:Ring gml:id="...">
      ...
      <gml:curveMember>
        <gml:Curve gml:id="...">
          <gml:segments>
            <gml:Arc gml:id="...">
              <gml:pos>P2</gml:pos>
              <gml:pos>P3</gml:pos>
              <gml:pos>P4</gml:pos>
            </gml:Arc>
          </gml:segments>
        </gml:Curve>
      </gml:curveMember>
    ...
  ...
</gml:Ring>
</gml:exterior>
</gml:PolygonPatch>
...
```

In fact, this is a particular case of the more general GML concept of “ArcString”², which is a curve segment that uses three-point circular arc interpolation in a piecewise fashion to “string” the arc segments together. The number of control points in the string is $(2 \times numArc) + 1$, where

² Note that ArcString is not foreseen by the AIXM GML Profile, as defined in Chapter 9 of this document.

numArc is the property defining the number of the arcs in the string, such as in Figure 7 - ArcString in ISO 19107.

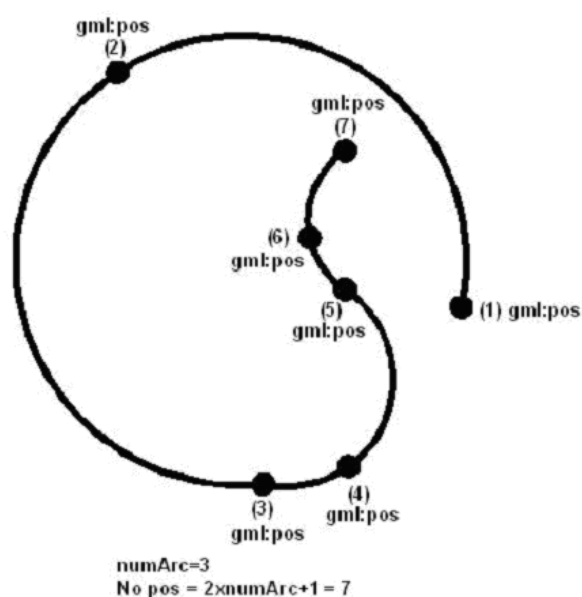


Figure 7 - ArcString in ISO 19107

8.2.4. Arc by centre point

Although this is the typical construct for arcs used in the definition of airspace borders in the AI domain, *it is recommended to avoid, as much as possible, the use of arcs by centre points*. It comes with two problems:

- The arc is over specified, as the start/end points, the centre and the radius are all provided. Typically, the calculated distance from the centre to the start and end point is not quite the same due to round-off error and is also usually different from the radius;
- This construction of arcs is not supported exactly this way in GML and in GIS systems in general.

The closest GML construct that can be used for encoding this type of arcs is ArcByCenterPoint. This requires calculating the start/end angles from the centre to the start/end points. Before calculating these angles, it is important to specify the angle measuring convention in AIXM, as GML seems to leave some degree of interpretation for this aspect.

8.2.4.1. Measuring angles in GML

GML explicitly implements³ the semantics of ISO 19107. The “startOfArc” (6.4.15.5) and “endOfArc” (6.4.15.6) are defined in terms of bearings. The definition of “bearing” is provided

³ Note that this is true for classes of spatial primitives (e.g. GM_Arc) but in general not their operations. In the ISO 19107, startOfArc and endOfArc are operations, aka constructors which derive other values from the GM_Arc. However, GML/ISO 19136 defines the UML Class ArcByCenterPoint (Figure D.48) in a GML profile of ISO 19107 where startOfArc and endOfArc are attributes. Although not explicitly stated in the GML standard, we can infer that the ISO 19107 semantics for

in Section 6.3.12, which states that “*Bearing is a data type used to represent direction in the coordinate reference system. In a 2D coordinate reference system, this can be accomplished using a “angle measured from true north” or a 2D vector point in that direction.*”

There are two variants mentioned in ISO 19107 for expressing bearings: angle and direction. The semantics for angle is given in 6.3.12.2 of the same document: “*In this variant of Bearing usually used for 2D coordinate systems, the first angle (azimuth) is measured from the first coordinate axis (usually north) in a counterclockwise fashion parallel to the reference surface tangent plane.*”

Although it may not be obvious, this definition matches the needs of the AI domain, as angles are usually expressed in degrees measured clockwise from the True North. The diagrams below explain why the “counter clockwise” convention stated in the ISO 19107 standard, **when combined with left-handed geodetic CRS** actually corresponds to a clockwise rotation in the AI domain.

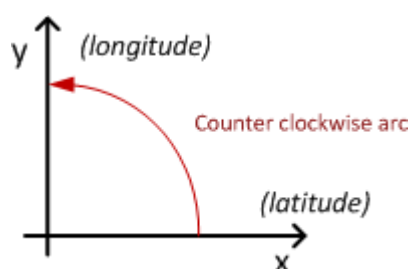


Figure 8 - Angle measured in a 2D geodetic CRS that has latitude as first axis

In the EPSG:4326 CRS, the first (x) axis is latitude and the second (y) axis is longitude. A counter clockwise angle means measuring it **from the first axis towards the second axis**. When transposing this coordinate system on the surface of the Earth, this corresponds to a clockwise rotation from the first axis (North) in order to measure angles, as shown in Figure 9. Practically, the x/y reference system is mirrored and rotated to be aligned with the meridians and the parallels.

operations carry over to the identical semantics for the corresponding attributes with the same name in the GML profile of ISO 19107 (ISO 19136, Annex D).

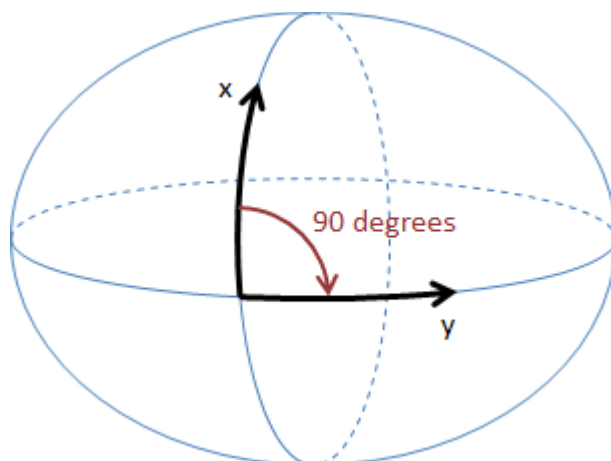


Figure 9 - Coordinate System Axes of EPSG:4326 CRS mapped on the Earth's surface

Therefore, when the EPSG:4326 CRS (or another 2D geodetic CRS that has latitude as first axis) is used, this translates to **angles that are measured clockwise starting from the True North in the AI Domain**. East is at 90 degrees from North, South at 180 degrees from North, etc. This convention is important for defining the startAngle and endAngle of the ArcByCenterPoint. Note that it is also possible to use negative values for angles. Negative angles are measured from the first axis through rotation in the direction opposite to the second axis.

The following diagram shows how angles are measured in WGS 84 2D with different coordinate systems:

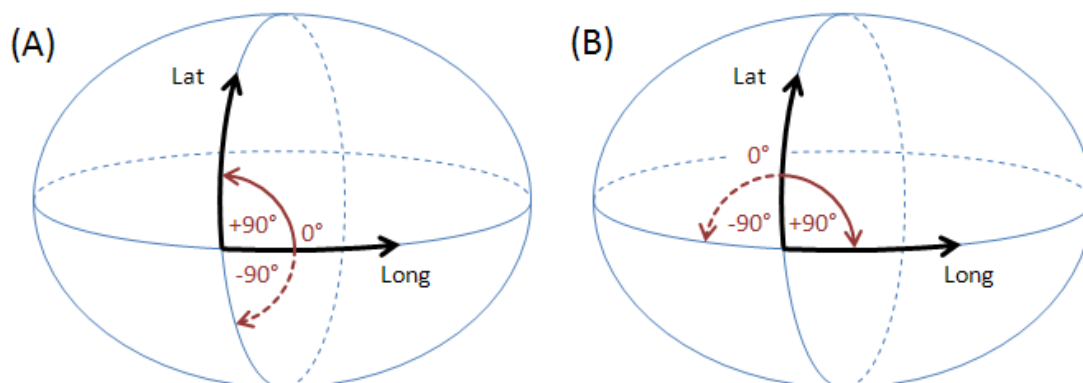


Figure 10 - Measuring angles in WGS 84 2D for different coordinate systems

(A) GeodeticCRS: urn:ogc:def:crs:OGC:1.3:CRS84 (right-handed)

- Datum: WGS84
- Ellipsoidal 2D CS.
 - Axes: (1st) longitude, (2nd) latitude.
 - Orientations: east, north. UoM: degree

(B) GeodeticCRS: urn:ogc:def:crs:EPSG::4326 (left-handed)

- Datum: WGS84

- Ellipsoidal 2D CS.
 - o Axes: (1st) latitude, (2nd) longitude.
 - o Orientations: north, east. UoM: degree

The order of the axes determines where 0° is located (on the positive part of the coordinate system's first axis).

8.2.4.2. Arc direction

Once the startAngle and endAngle are known, it is still necessary to establish how the arcs are interpolated/drawn. The semantics of the words “start” and “end” indicate that arcs shall be interpolated/drawn from the start angle to the end angle, similarly to a line that is always interpolated/drawn from its start to its end. However, this still leaves some room for interpretation, e.g. an arc that has startAngle=90 (East) and endAngle=180 (West) should be interpolated/drawn through South or through North?

The following convention shall apply in the aviation domain: ***if the start angle is smaller than the end angle then the arc direction is the direction in which the angle values increase***. If the opposite is true, then the arc direction is the one in which the angle values decrease. Depending upon the coordinate system that applies to a given ArcByCenterPoint, this results in a clockwise (left-handed system) or counter-clockwise (right-handed system) directed arc. The arguments for this convention are detailed in Annex B.

Applied with the ***EPSG:4326 CRS***, this means that ***arcs are drawn:***

- ***clockwise on the surface of the Earth when the startAngle is smaller than the endAngle;***
- ***counter-clockwise on the surface of the Earth when the startAngle is larger than the endAngle***

The same convention applies to any other geodetic CRS that has latitude as first (x) axis. This is exemplified in Figure 11. It is therefore possible to define arcs in both clockwise and counter-clockwise direction by choosing the right startAngle and endAngle values. This also requires the use of angle values between -360 and 360, both values included.

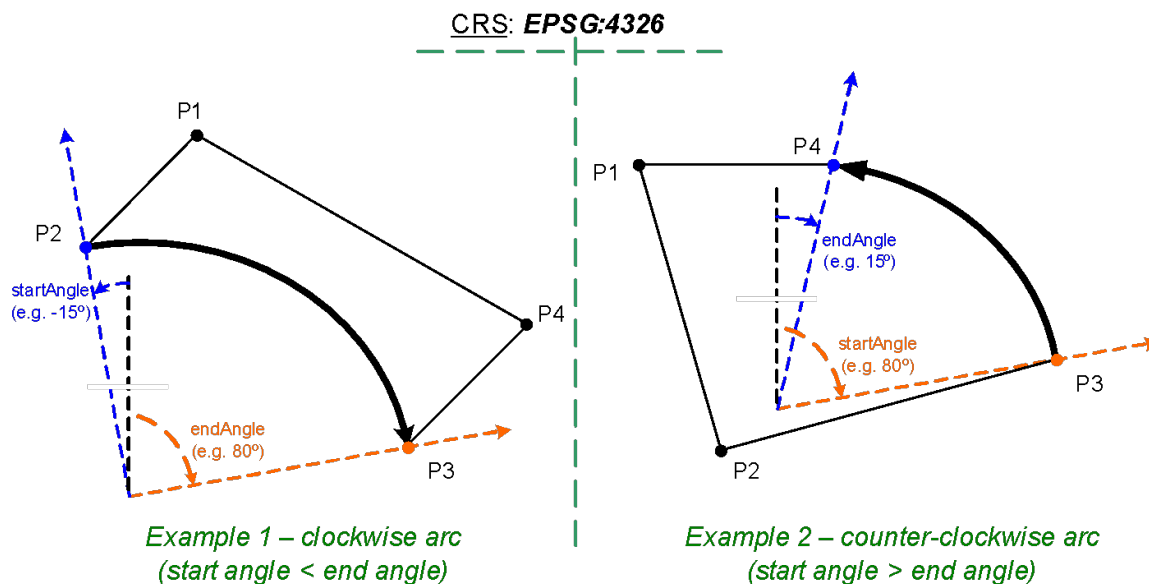


Figure 11 - Arcs are drawn from startAngle to endAngle

Applied with the **CRS:84 CRS**, this means that *arcs are drawn*:

- *counter-clockwise on the surface of the Earth when the startAngle is smaller than the endAngle;*
- *clockwise on the surface of the Earth when the startAngle is larger than the endAngle*

An example of the GML encoding for an ArcByCenterPoint is presented below.

```

...
<aixm:Surface gml:id="S01" srsName="urn:ogc:def:crs:EPSG::4326">
  <gml:patches>
    <gml:PolygonPatch>
      <gml:exterior>
        <gml:Ring>
          <gml:curveMember>
            <aixm:Curve gml:id="C01">
              <gml:segments>
                <gml:GeodesicString>
                  <gml:posList>lat_Px long_Px lat_Py long_Py</gml:posList>
                </gml:GeodesicString>
                <gml:ArcByCenterPoint gml:id="A01">
                  <gml:pos>lat_Pc long_Pc</gml:pos>
                  <gml:radius uom="m">radius</gml:radius>
                  <gml:startAngle uom="deg">calculated_start_angle</gml:startAngle>
                  <gml:endAngle uom="deg">calculated_end_angle</gml:endAngle>
                </gml:ArcByCenterPoint>
                <gml:GeodesicString>
                  <gml:posList>lat_Pz long_Pz lat_Pw long_Pw</gml:posList>
                </gml:GeodesicString>
                ...
              </gml:segments>
            </aixm:Curve>
          </gml:curveMember>
        </gml:Ring>
      </gml:exterior>
    </gml:PolygonPatch>
  </gml:patches>
</aixm:Surface>

```

```

        </gml:Ring>
      </gml:exterior>
    </gml:PolygonPatch>
  </gml:patches>
</aixm:Surface>
...

```

8.2.4.3. Units of measurement

The previous XML encoding example also shows recommended values for the uom attributes of gml:radius and gml:start(end)Angle. According to GML section 8.2.3.7, “*in an instance document, on elements of type gml:MeasureType the mandatory uom attribute shall carry a value corresponding to either:*

- *a conventional unit of measure symbol,*
- *a link to a definition of a unit of measure that does not have a conventional symbol, or when it is desired to indicate a precise or variant definition.*

For common units of measurement, such as meter for distances and degrees for angles, conventional units of measure symbols shall be used. A Note in the [GML] standard suggests the use of UCUM symbols: “*It is recommended that the symbol be an identifier for a unit of measure as specified in the Unified Code of Units of Measure’ (UCUM) [9]. This provides a set of symbols and a grammar for constructing identifiers for units of measure that are unique, and may be easily entered with a keyboard supporting the limited character set known as 7-bit ASCII.*”

The following UCUM “c/s” (case sensitive) values shall be used for gml:radius in AIXM/GML data sets:

- **m** – when the radius is expressed in meters
- **km** – when the radius is expressed in kilometers
- **[nmi_i]** – when the radius is expressed in Nautical Miles

The symbol “**deg**” shall be used for the uom attribute of gml:startAngle and gml:endAngle elements.

8.2.5. Circle by center point

Full circles are also used in order to define the geometry of certain airspace in the AI domain. They can be directly encoded using the gml:CircleByCenterPoint. It is quite deep in the structure, as presented in the example below.

```

...
<aixm:Surface gml:id="S001" srsName="urn:ogc:def:crs:EPSG::4326">
  <gml:polygonsPatches>
    <gml:PolygonPatch>
      <gml:exterior>
        <gml:Ring>
          <gml:curveMember>
            <gml:Curve gml:id="CUR001">
              <gml:segments>

```

```

    <gml:CircleByCenterPoint numArc="1">
      <gml:pos>51.01555556 2.57138889</gml:pos>
      <gml:radius uom="[nmi_i]">12</gml:radius>
    </gml:CircleByCenterPoint>
  </gml:segments>
</gml:Curve>
</gml:curveMember>
</gml:Ring>
</gml:exterior>
</gml:PolygonPatch>
</gml:polygonPatches>
</aixm:Surface>
...

```

Note that CircleByCenterPoint should be used only to define a simple circular airspace boundary and it should appear as *unique member of a gml:Curve*. Otherwise, the orientation of the circle boundary is not well defined.

8.2.6. Perimeter encoding direction

The GML Surface implements ISO 19107 GM_Surface whose exterior boundary shall be encoded counter-clockwise and any interior boundary encoded clockwise.

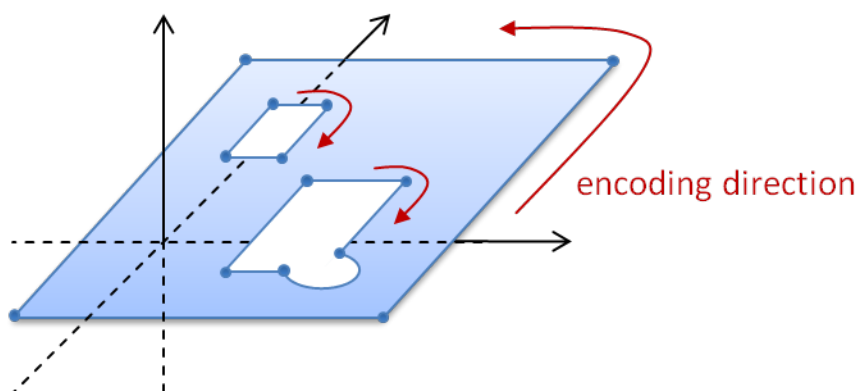


Figure 12 - Encoding direction for surface boundaries

For aeronautical data, this implies that the outside perimeter and any eventual holes shall be encoded as shown in Figure 12. This rule might be difficult to apply strictly on short term as existing aeronautical systems require data to be encoded in clockwise direction. For example, the Operating Procedures for Aeronautical Dynamic Data (OPADD), version 3.0, indicates in 2.3.22.12 that “*points defining lateral limits of an area should be enumerated in clockwise order*”. Further details can be found in Annex D.

In addition, in the current AIXM version 5, there is no need for encoding Surface holes. Eventual Airspace “holes” are encoded as subtractions of an AirspaceVolume, as explained in the “[Airspace aggregation](#)” section. Therefore, internal patches (rings) are excluded from the aviation data profile of GML.

Compliance with this ISO 19107 rule shall remain an objective for AIXM implementations. However, on short term it might be expected that certain AIXM data will not comply with this

rule and application developers shall be aware of this exception. As the use of GML internal patches (rings) is not supported by this profile, non-compliance with this rule is not expected to have practical consequences. Existing GIS software often is rather lenient when it comes to this rule and often ignores it.

8.2.7. Corridor

Corridors are sometimes used in the definition of airspace geometries. This is done by specifying a centerline and a width or half-width and it is supported in AIXM through the following properties of the AirspaceVolume class:

- “width” attribute
- “centreline” association with Curve

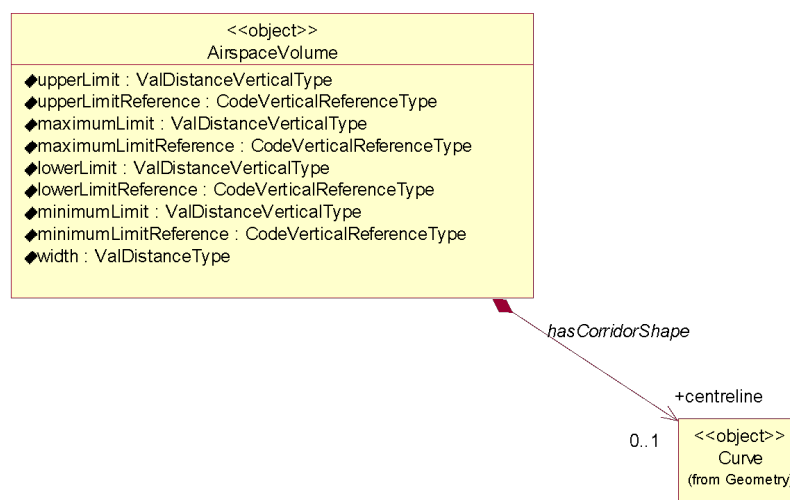


Figure 13 - Corridors defined as AirspaceVolume having a “centreline” association with Curve

The encoding of airspace corridors in this way does not make full use of the GML capabilities. Theoretically, the gml:OffsetCurve, which is an implementation of the GM_OffsetCurve, could be used to encode the corridor shape in GML. However, because of some lack of clarity that exists with regard to GM_OffsetCurve, the use OffsetCurve shall be avoided until clarified in a future ISO 19107 update.

8.2.8. Other geometries

Some specific geometrical constructs are used in the design of instrument approach/departure procedures and in the design of route segments in the AI domain. They are mentioned in this section for completeness sake. More details about their encoding in GML might be provided in a future version of this paper.

The first one is the “wind spiral”, as represented in Figure 14.

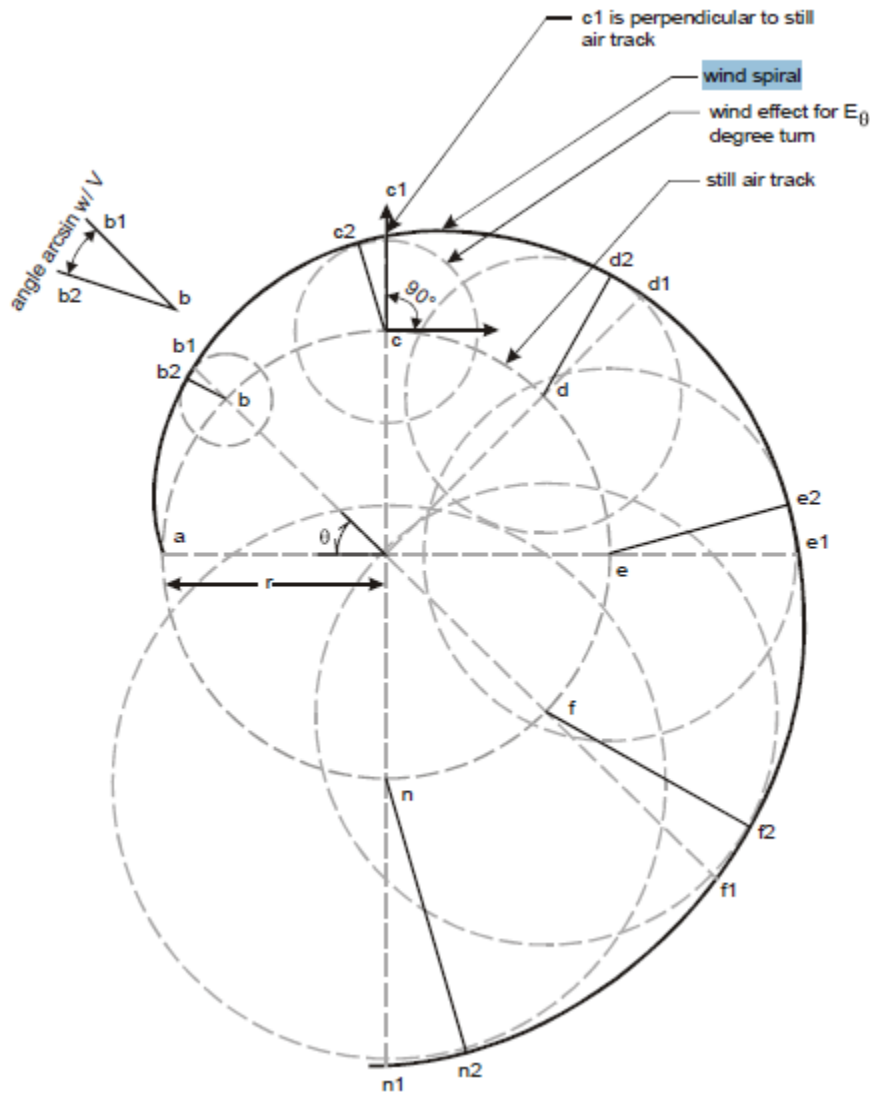


Figure 14 - Wind Spiral

Other such geometrical constructs are the “segment locus” (Figure 15) and “arc locus” (Figure 16).

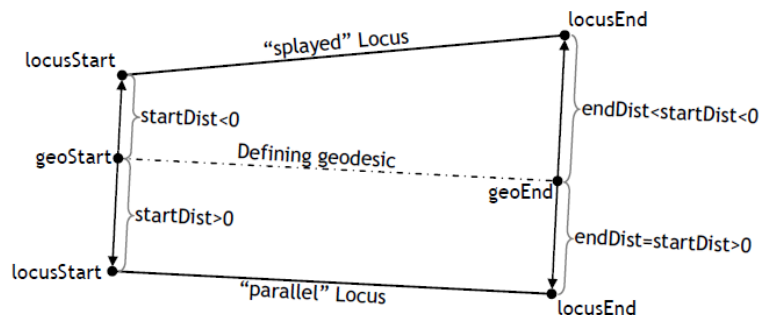


Figure 15 - Segment Locus (a “parallel” and a “splaying” locus with respect to a given geodesic line.)



Figure 16 - Arc Locus

Arc locus is the non-flat generalization of Archimedean spiral. An arc locus is a set of points whose (geodesic) distance from a given defining geodesic circle is a linearly varying function of the length parameter on the circle, of the point projection.

9. Airspace aggregation

9.1. Background

There are two main ways to describe the geometry of airspace volumes in the AI Domain:

- by providing a horizontal border and vertical limits;
- by providing a composition rule by which the airspace is defined as a series of unions, intersections, subtractions of other Airspace, such as in the following examples:
 - Airspace of type CTR defined as a “*circle of 50 NM from which the portion of airspace situated in a neighboring FIR is subtracted*”;
 - Airspace of type UIR that has “*the same horizontal projection as an FIR*”, but different vertical limits;
 - Airspace of type CTA which is the result of aggregating some Airspace of type SECTOR;
 - etc.

The AirspaceVolume class of the AIXM 5 model was designed in order to support the encoding of such aggregated Airspace. Note that the “operation” property of the AirspaceGeometryComponent is used. This makes this encoding not-directly understandable for a standard GML tool and requires software customisation.

Although GML supports the creation of composite surfaces using “patches”, it is not possible to leave this aggregation to the GML level because the vertical limits of the different components might be different. The possibility of using 3D geometries might be considered in future. Until then, using only 2D GML components requires custom processing of AIXM/GML files in order to correctly represent the result of an airspace aggregation, in particular the use of the operation and operationSequence attributes of the AirspaceGeometryComponent class.

9.2. GML encoding

The model gives the possibility for using several approaches for the encoding of airspace aggregations/dependencies. The use of a particular method, from the ones described further in this section, depends on the intended use of the data.

In order to exemplify these methods, the example of an Airspace of type “CTA” will be used.

9.2.1. By reference

The first method is limited to referring to another airspace, but without effectively copying the geometry of that Airspace as own AirspaceVolume(s). The UML diagram Figure 17 indicates which elements of the AIXM 5.1 model are used or not used when applying this method.

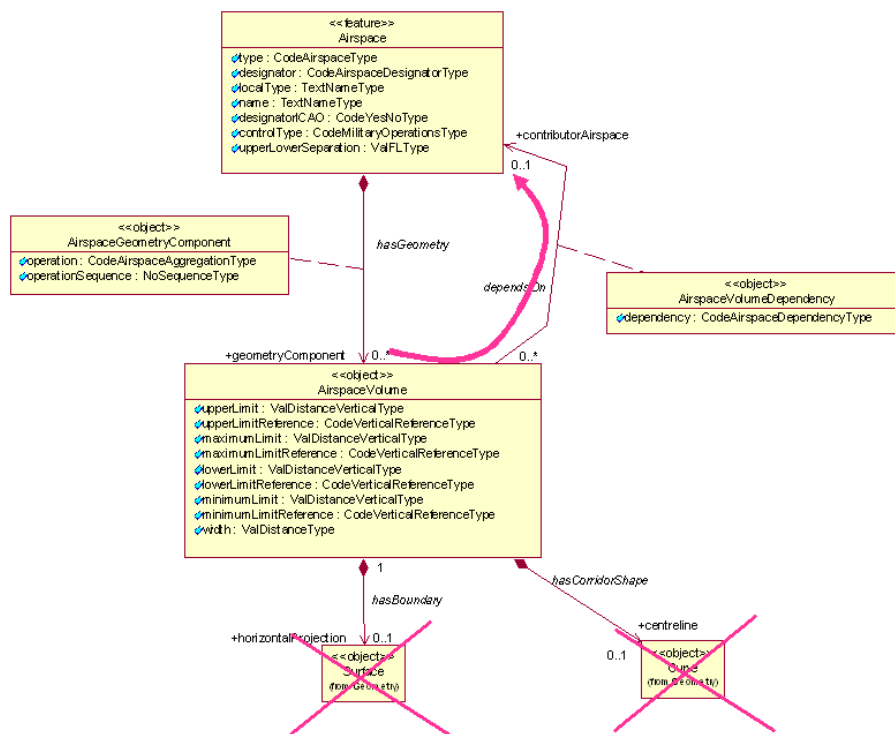


Figure 17 - Model use for airspace aggregation by reference

This method might be appropriate for data provision between synchronized databases, such as between a local and a regional database and it is equivalent to the approach of the previous AIXM 4.5 version (which is not based on GML). The disadvantage of this method is that the client needs to eventually retrieve the geometry of the referenced Airspace and do the geospatial calculations that are necessary in order to effectively get the actual geometry of the current Airspace in a GML usable form. The advantage is that it preserves a true association with the composing Airspace. An encoding example is provided below:

...

```

<aixm:type>CTA</aixm:type>
<aixm:designator>EADD</aixm:designator>
<aixm:name>CTA DONLON</aixm:name>
<aixm:geometryComponent>
  <aixm:AirspaceGeometryComponent gml:id="AV001">
    <aixm:operation>BASE</aixm:operation>
    <aixm:operationSequence>1</aixm:operationSequence>
    <aixm:theAirspaceVolume>
      <aixm:AirspaceVolume gml:id="V001">
        <aixm:contributorAirspace>
          <aixm:AirspaceVolumeDependency gml:id="VV001">
            <aixm:dependency>FULL_GEOMETRY</aixm:dependency>
          
```

```

                                <aixm:theAirspace      xlink:href="urn:uuid:204451c5-be5e-4eaf-8859-
0a62b24a389d" xlink:title="SECTOR DONLON EAST"/>
                                </aixm:AirspaceVolumeDependency>
                                </aixm:contributorAirspace>
                                </aixm:AirspaceVolume>
                                </aixm:theAirspaceVolume>
                                </aixm:AirspaceGeometryComponent>
</aixm:geometryComponent>
<aixm:geometryComponent>
  <aixm:AirspaceGeometryComponent gml:id="AV002">
    <aixm:operation>UNION</aixm:operation>
    <aixm:operationSequence>2</aixm:operationSequence>
    <aixm:theAirspaceVolume>
      <aixm:AirspaceVolume gml:id="V002">
        <aixm:contributorAirspace>
          <aixm:AirspaceVolumeDependency gml:id="VV002">
            <aixm:dependency>FULL_GEOMETRY</aixm:dependency>
            <aixm:theAirspace      xlink:href="urn:uuid:b936e0e4-2b58-404f-9d95-
d95c421c50d2" xlink:title="SECTOR DONLON WEST"/>
          </aixm:AirspaceVolumeDependency>
        </aixm:contributorAirspace>
      </aixm:AirspaceVolume>
    </aixm:theAirspaceVolume>
  </aixm:AirspaceGeometryComponent>
</aixm:geometryComponent>
...

```

9.2.1.1. Temporal aspects for abstract references

When using abstract references, an important issue arises that must be dealt with regard to the temporal synchronisation of the data. Since an abstract reference always points to a complete feature rather than a single time slice, it is very important that there is no ambiguity in which time slice is meant. The referenced object must not “change” during the valid time of the reference. More exactly, the referenced object (the Airspace in this case) must have a time slice whose valid time covers completely the valid time of the referencing feature’s time slice. See a more complete description of this issue in “[Temporal aspects for abstract references](#)”, later in this document.

9.2.2. By copying the geometry

The second method consists in effectively copying the geometry of the referenced Airspace as local AirspaceVolume. Note that this might be a recursive operation, as the referenced Airspace might have more than one AirspaceVolume and some or even all these could also depend on the geometry of other Airspace.

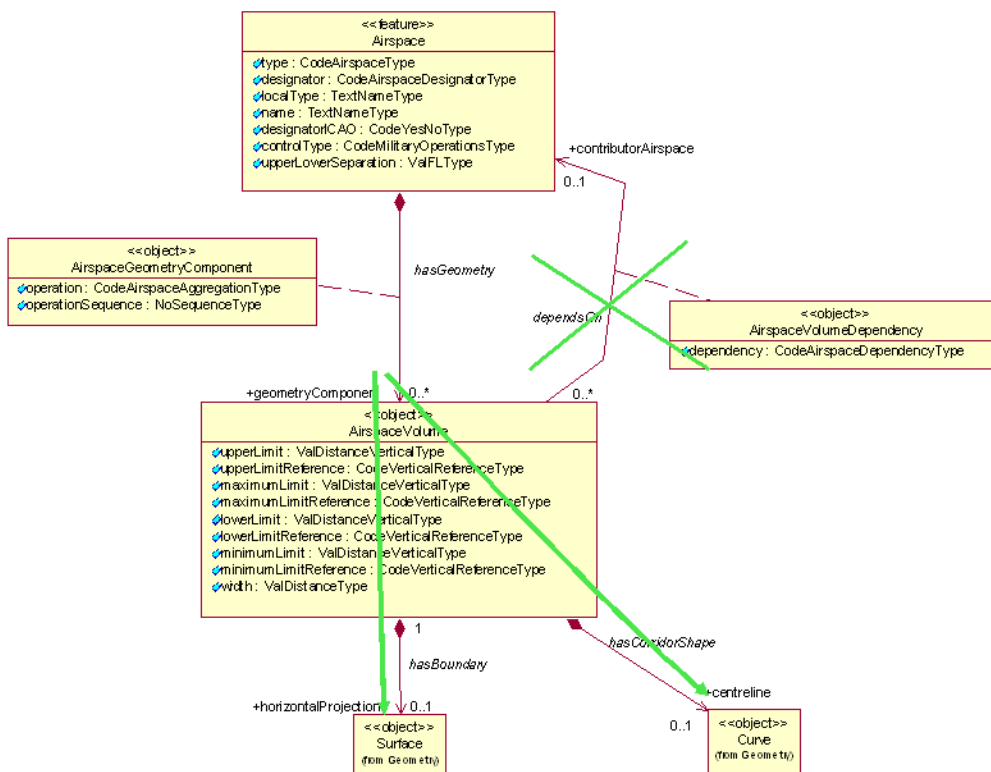


Figure 18 - Model use for airspace aggregations when copying referenced geometries

This method might be appropriate for applications that need to provide fully digested geometrical data for direct consumption (e.g. graphical visualization, spatial calculations). The disadvantage of this method is that the referenced geometry might also change in time. This is not a problem when the aggregation is used for the provision of SNAPSHOT data (valid at a time instant) but it might become problematic when providing Baseline data (which is valid for a period of time). Future changes of the geometry of referenced airspace needs to be propagated to the AirspaceVolume of the aggregated airspace. The advantage is that this method provides complete geometrical data for the aggregated Airspace and does not require further calculations by the client system. An encoding example is provided below:

...

```

<aixm:type>CTA</aixm:type>
<aixm:designator>EADD</aixm:designator>
<aixm:name>CTA DONLON</aixm:name>
<aixm:geometryComponent>
  <aixm:AirspaceGeometryComponent gml:id="AV001">
    <aixm:operation>BASE</aixm:operation>
    <aixm:operationSequence>1</aixm:operationSequence>
    <aixm:theAirspaceVolume>
      <aixm:AirspaceVolume gml:id="V001">
        <aixm:upperLimit uom="FL">245</aixm:upperLimit>
        <aixm:upperLimitReference>STD</aixm:upperLimitReference>
        <aixm:lowerLimit uom="FL">30</aixm:lowerLimit>
        <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
        <aixm:horizontalProjection>
          <aixm:Surface gml:id="S001">
            <gml:patches>
              <gml:PolygonPatch>
                <gml:exterior>
                  <gml:Ring>
                    <gml:curveMember>
                      <aixm:Curve gml:id="C001">
                        <gml:segments>
                          <gml:GeodesicString>
                            <gml:posList>52.18556 5.20833 52.20611
5.2875 52.18917 5.29889 52.16917 5.29889 52.18556 5.20833</gml:posList>
                          </gml:GeodesicString>
                        </gml:segments>
                      </aixm:Curve>
                    </gml:curveMember>
                  </gml:Ring>
                </gml:exterior>
              </gml:PolygonPatch>
            </gml:patches>
          </aixm:Surface>
        </aixm:horizontalProjection>
      </aixm:AirspaceVolume>
    </aixm:theAirspaceVolume>
  </aixm:AirspaceGeometryComponent>
</aixm:geometryComponent>
<aixm:geometryComponent>
  <aixm:AirspaceGeometryComponent gml:id="AV002">
    <aixm:operation>UNION</aixm:operation>
    <aixm:operationSequence>2</aixm:operationSequence>
    <aixm:theAirspaceVolume>
      <aixm:AirspaceVolume gml:id="V002">
        <aixm:upperLimit uom="FL">245</aixm:upperLimit>
        <aixm:upperLimitReference>STD</aixm:upperLimitReference>
        <aixm:lowerLimit uom="FL">50</aixm:lowerLimit>
        <aixm:lowerLimitReference>STD</aixm:lowerLimitReference>
        <aixm:horizontalProjection>
          <aixm:Surface gml:id="S002">
            <gml:patches>
              <gml:PolygonPatch>
                <gml:exterior>
                  <gml:Ring>

```

```

                    <gml:curveMember>
                      <aixm:Curve gml:id="C002">
                        <gml:segments>
                          <gml:GeodesicString>
                            <gml:posList>52.20611 5.2875 52.18917
5.29889 52.19117 5.3289 52.20611 5.2875</gml:posList>
                            </gml:GeodesicString>
                          </gml:segments>
                        </aixm:Curve>
                      </gml:curveMember>
                    </gml:Ring>
                  </gml:exterior>
                </gml:PolygonPatch>
              </gml:patches>
            </aixm:Surface>
          </aixm:horizontalProjection>
        </aixm:AirspaceVolume>
      </aixm:theAirspaceVolume>
    </aixm:AirspaceGeometryComponent>
  </aixm:geometryComponent>
</aixm:AirspaceTimeSlice>
...

```

9.2.3. Combined method

The two methods can also be combined. However this is only appropriate for SNAPSHOT TimeSlices, as discussed in “[Abstract reference to remote feature](#)” later in this document.

10. Point references and annotations

10.1. Background

Positions of Navaids or Designated Points may be used in the AI Domain in order to define the shape of an Airspace. They can be used as arc centers or even as boundary points. In the case of NOTAM messages, the latitude/longitude coordinates may be followed by geographical references, such as in the example below.

“E) AIR DISPLAY WILL TAKE PLACE WI LATERAL LIMITS: 443838N 0200818E (NDB OBR) - 444508N 0201455E (VILLAGE JAKOVO) - 443445N 0202447E - 443838N 0200818E (NDB OBR). F) GND G) 3000FT AMSL)”

This is not always a reference to a significant point, it can be simply an annotation of a position (e.g. “Village Jakovo”). However, the encoding solution being quite similar, this case also is discussed here.

The UML model of AIXM shows a “dependency” association between Surface and SignificantPoint in order to cater for such situations:

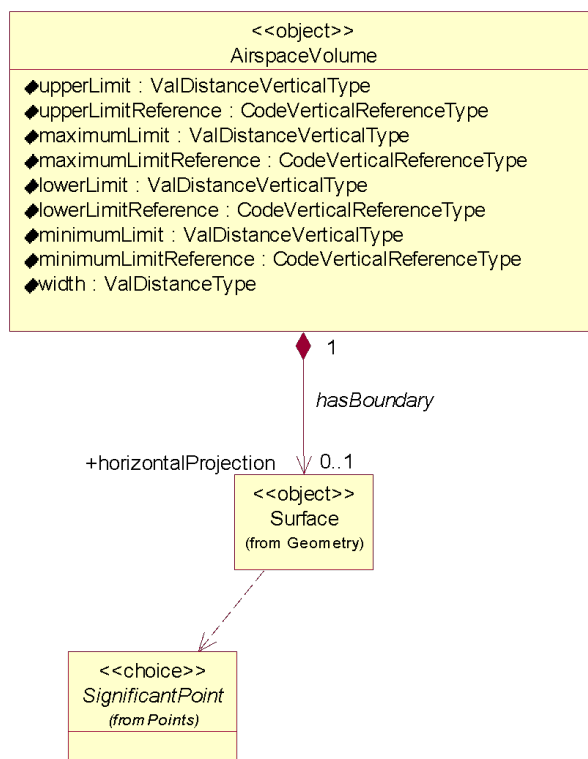


Figure 19 - Point references model in AIXM

The reason for using a “dependency” association and not a standard “object to feature association” is that this occurs deep inside the GML encoding of the Surface. Using a

“dependency” also indicates that the actual encoding can be done differently, based on the intended use of the data.

10.2. GML encoding

The encoding of point references and annotations can be done using `gml:pointProperty` elements, which can appear as a descendant of `gml:Curve`, for example in the construction of a `gml:GeodesicString`. Note that the `pointProperty` allows either referring to another `gml:Point` (by `xlink:href`) or providing a `gml:Point` child element.

According to the GML standard, para 10.2.2.2: “*A property that has a point as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). **Either the reference or the contained element shall be given, but neither both nor none.***”

10.2.1. Using `aixm:Point` annotations

In this case, a `gml:pointProperty` is used, including an `aixm:Point` with an annotation (`aixm:Note`). This encoding has the advantage that the geometry is self-contained (the position of the referenced object is directly copied as a `gml:pos` element).

This method should be used whenever the data is intended “for human consumption”, such as in the case of the NOTAM examples (e.g. “VILLAGE JAKOVO”). Even in the case when an arc center is located on a DME navaid and the distance information provided by the DME can be used to keep the aircraft inside or outside the arc, the provision of a `Point` annotation could be sufficient for the end user.

An example is provided below:

```
...
  <gml:exterior>
    <gml:Ring>
      <gml:curveMember>
        <gml:Curve gml:id="C001">
          <gml:segments>
            <gml:GeodesicString>
              <gml:posList>52.1855 5.2083 52.2061 5.2875 52.1891 5.2988 52.1691 5.2988</gml:posList>
            </gml:GeodesicString>
            <!-- The next segment contains a point annotation encoded as a Note-->
            <gml:GeodesicString>
              <gml:pos>52.16917 5.29889</gml:pos>
            <gml:pointProperty>
              <aixm:Point gml:id="P001">
                <gml:pos>52.16917 5.21972</gml:pos>
                <aixm:annotation>
                  <aixm:Note gml:id="N001">
                    <aixm:translatedNote>
                      <aixm:LinguisticNote gml:id="N002">
                        <aixm:note lang="ENG">VILLAGE JAKOVO</aixm:note>
                      </aixm:LinguisticNote>
                    </aixm:translatedNote>
                  </aixm:Note>
                </aixm:annotation>
              </aixm:Point>
            </gml:pointProperty>
          </gml:GeodesicString>
        </gml:segments>
      </gml:Curve>
    </gml:curveMember>
  </gml:Ring>
</gml:exterior>
```

```

        </aixm:annotation>
      </aixm:Point>
    </gml:pointProperty>
  </gml:GeodesicString>
  <!-- This is the final straight segment encoded as a Geodesic, which closes the surface-->
  <gml:GeodesicString>
    <gml:posList>52.16917 5.21972 52.18556 5.20833</gml:posList>
  </gml:GeodesicString>
</gml:segments>
</gml:Curve>
</gml:curveMember>
</gml:Ring>
</gml:exterior>
...

```

10.2.2. Using xlink:href

When necessary to preserve as a true reference the information that the current position depends on the location of another aeronautical feature, then a `gml:PointProperty` with a `xlink:href` attribute can be used. In this case, there shall be no child `gml:Point/gml:pos` element. The GML standard requires a local reference, using a `gml:id` value.

10.2.2.1. Local reference to `gml:Point` (or equivalent)

In the example below, the position of the Navaid is used as centre for the circle that defines the horizontal geometry of the Airspace.

```

<aixm:Navaid gml:id="urn:uuid.791fb712-6c7a-46bb-8e98-49d76942573e">
...
  <aixm:type>VOR_DME</aixm:type>
  <aixm:name>DONLON</aixm:name>
  <aixm:location>
    <aixm:ElevatedPoint gml:id="#P0001" srsName="urn:ogc:def:crs:EPSG::4326">
      <gml:pos>52.2889 -32.0350</gml:pos>
      <aixm:elevation uom="FT">365</aixm:elevation>
    </aixm:ElevatedPoint>
  </aixm:location>
...
</aixm:Navaid>
...
<aixm:Airspace gml:id="urn:uuid.fc5b4fb3-004e-42c4-8552-6566d25a09f7">
...
  <aixm:theAirspaceVolume>
    <aixm:AirspaceVolume gml:id="V001">
      <aixm:horizontalProjection>
        <aixm:Surface gml:id="S001" srsName="urn:ogc:def:crs:EPSG::4326">
          <gml:polygonPatches>
            <gml:PolygonPatch>
              <gml:exterior>
                <gml:Ring>
                  <gml:curveMember>
                    <gml:Curve gml:id="CUR001">
                      <gml:segments>
                        <gml:CircleByCenterPoint numArc="1">
                          <gml:pointProperty xlink:href="#P0001"

```

```

xlink:title="VOR/DME DONLON"/>
  <gml:radius uom="[nmi_i]">12</gml:radius>
</gml:CircleByCenterPoint>
</gml:segments>
</gml:Curve>
</gml:curveMember>
</gml:Ring>
</gml:exterior>
</gml:PolygonPatch>
</gml:polygonPatches>
</aixm:Surface>
</aixm:horizontalProjection>
</aixm:AirspaceVolume>
...
</aixm:Airspace>

```

This solution is appropriate when the data is provided for direct consumption by a GML tool for display or other calculation purpose. Obviously, it requires that both the Airspace and the referenced feature (Navaid, DesignatedPoint, etc.) are included in the same file. It might be problematic to apply this solution in the case of WFS getFeature requests, because the referenced feature will not be present in the response.

Note also that the xlink:title attribute is used to provide a human readable identification of the Navaid that is referred, which can be used in printed documents.

This solution does not imply the persistence of the gml:id value. It is still a temporary identifier, which enables linking the gml:PointProperty with the gml:Point or one of its allowed substitutions (aixm:Point, aixm:ElevatedPoint) inside the file.

This direct link between gml:PointProperty and gml:Point is a deviation from the general AIXM principle of having xlink:href associations towards the feature level only. However, this direct association with the gml:Point property of the aixm:Navaid is the only solution identified for really encoding geometry dependencies at the GML level. In a source database, the association can still be towards the Navaid itself (as detailed in the next section). Only for data export/import purpose the reference would be towards the gml:Point directly.

10.2.3. Summary

In conclusion, there are two options for encoding point references in AIXM/GML:

- as a simple annotation
- as a local concrete xlink:href reference using gml:id

The most appropriate one depends on the intended usage of the data. Therefore, AIXM applications should offer the client the possibility to specify how such references should be exported: to be preserved or be replaced with copies of gml:Point elements, eventually including the reference as an annotation.

11. Geographical border references

11.1. Background

In the AI Domain, Airspace boundaries may be based on national borders or on other geographical features, such as shorelines, rivers, etc.

An example of such an Airspace border is provided below:

“UBP3
400300N 0455323E - 400300N 0465600E - 392545N 0472148E -
then along the state border with Islamic Republic of Iran up to 385222N 0463250E -
then along the state border with Armenia up to 400300N - 0455323E”

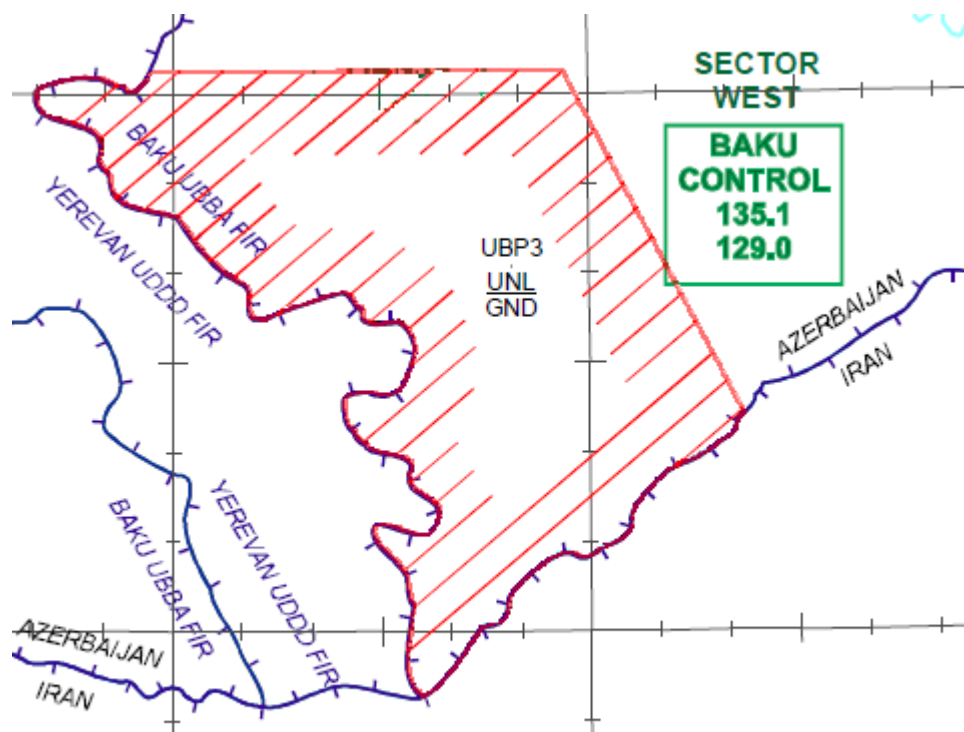


Figure 20 - Airspace UBP3 borders

A particularity of this situation is that official definitions of the airspace, as provided in the Aeronautical Information Publication (AIP) or in NOTAM messages, do not include the actual geometry of the referenced geographical border. It is left for the end users to derive the actual geometry of the airspace by using a source of geographical border data.

The UML model of AIXM shows a “dependency” association between Surface and GeoBorder in order to cater for such situations:

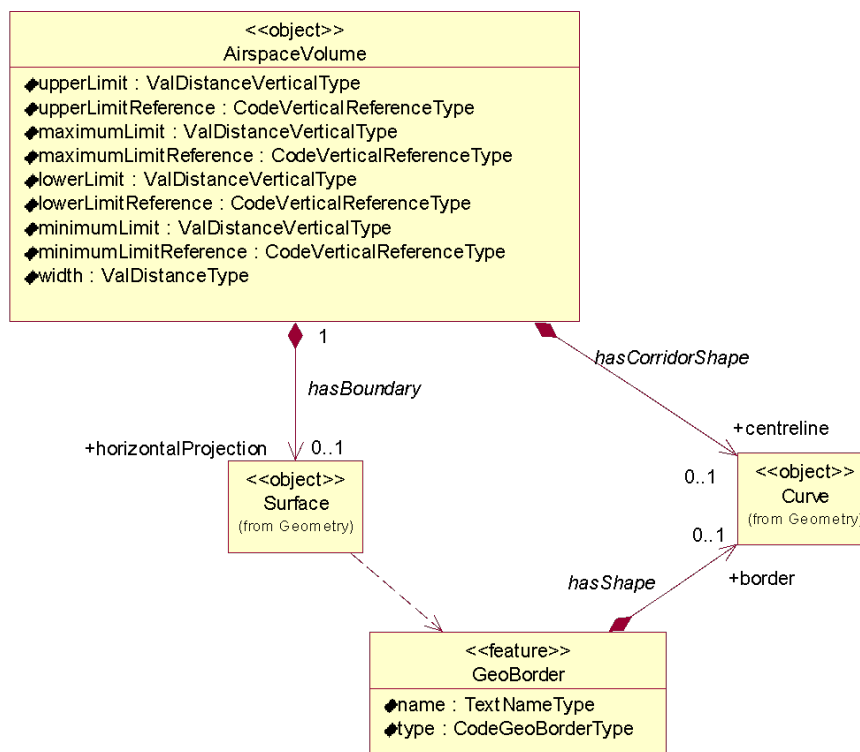


Figure 21 - GeoBorder references model in AIXM

Similarly to Significant Point references (as discussed in 7.1), the reason for using a “dependency” association and not a standard “object to feature association” is that this occurs deep inside the GML encoding of the Surface. Using a “dependency” also indicates that the actual encoding can be done differently, based on the intended use of the data.

11.2. GML encoding

The encoding of GeoBorder references can be done in two ways:

- either using the “annotation” property of an aixm:Curve, for applications where a simple text remark is sufficient;
- or using the xlink:href attribute of a gml:curveMember, for applications where a true reference needs to be preserved.

11.2.1. Using aixm:Curve annotations

In this case, an aixm:Curve is used as content of a gml:curveMember, which allows including an annotation (aixm:Note). This encoding has the advantage that the geometry is self-contained (the relevant series of latitude/longitude pairs from the referenced GeoBorder are directly copied in a gml:GeodesicString/gml:posList element).

This method should be used when the information (that this part of the airspace border actually comes from a GeoBorder) is intended “for human consumption”, such as for directly displaying the Airspace on a screen or printing on paper. An example is provided below:

```

<aixm:Airspace gml:id="urn.uuid.1965dd58-6898-4065-8f21-b1774c959bbb">
  ...
  <aixm:horizontalProjection>
    <aixm:Surface gml:id="S001" srsName="urn:ogc:def:crs:EPSG::4326">
      <gml:polygonPatches>
        <gml:PolygonPatch>
          <gml:exterior>
            <gml:Ring>
              <gml:curveMember>
                <gml:Curve gml:id="CUR001">
                  <gml:segments>
                    <gml:LineStringSegment interpolation="linear">
                      <!-- because the two consecutive points have the same latitude, the first
segment is encoded as a parallel (linear interpolation in EPSG:4326) -->
                      <gml:posList>          40.05          45.88972222          40.05
46.93333333</gml:posList>
                    </gml:LineStringSegment>
                    <gml:GeodesicString interpolation="geodesic">
                      <gml:posList>40.05 46.93333333 39.42916667
47.36333334</gml:posList>
                    </gml:GeodesicString>
                  </gml:segments>
                </gml:Curve>
              </gml:curveMember>
              <!-- Here starts the first portion of the Airspace border that was extracted (copied) from
a GeoBorder. In this case, the reference to the GeoBorder is a simple text annotation.-->
              <gml:curveMember>
                <aixm:Curve gml:id="CUR002">
                  <gml:segments>
                    <gml:GeodesicString interpolation="geodesic">
                      <gml:posList>39.42916667 47.36333334 39.426818 47.353277
39.405269 47.340623 39.370714 47.303951 39.317977 47.213494 39.304679 47.147725 39.253854 47.075486 39.209945
47.064339 39.138967 46.950699 39.160036 46.929733 39.135505 46.835985 39.113835 46.826968 39.110637 46.818902
39.104488 46.811639 39.084995 46.792338 39.079520 46.774177 39.063644 46.761250 39.035348 46.762985 39.015666
46.733063 39.019323 46.695507 38.991019 46.690236 38.987819 46.672322 38.930984 46.626817 38.906155 46.600158
38.885792 46.560942 38.885160 46.554943 38.87277778 46.54722222</gml:posList>
                    </gml:GeodesicString>
                  </gml:segments>
                <!-- Here starts the first portion of the Airspace border that was extracted (copied) from
a GeoBorder. In this case, the reference to the GeoBorder is a simple text annotation.-->
                <aixm:annotation>
                  <aixm:Note gml:id="N001">
                    <aixm:translatedNote>
                      <aixm:LinguisticNote gml:id="N002">
                        <aixm:note lang="ENG">along the state border with
Islamic Republic of Iran</aixm:note>
                      </aixm:LinguisticNote>
                    </aixm:translatedNote>
                  </aixm:Note>
                </aixm:annotation>
              </gml:Curve>
            </gml:curveMember>
          </gml:Ring>
        </gml:PolygonPatch>
      </gml:polygonPatches>
    </aixm:Surface>
  </aixm:horizontalProjection>
</aixm:Airspace>

```

```

...
    </gml:curveMember>
  </aixm:Airspace>

```

Note that the first point of the curveMember that contains the portion extracted from the GeoBorder (39.42916667 47.36333334) is equal with the last point of the previous curveMember. This satisfies the requirement stated in the GML Standard (ISO 19136, 10.5.11.1 - Ring, RingType, curveMember) that *“In the context of a ring, the curves describe the boundary of the surface. The sequence of curves shall be contiguous and connected in a cycle.”* In order to match this requirement when encoding existing Airspace data an additional gml:curveMember might be necessary. This is required in order to connect the last specified point of the Airspace boundary before the GeoBorder with the first vertex of the GeoBorder. It is typically a very short segment which does not alter the geometry. However, its presence is important in order to have a contiguous Surface border.

11.2.2. Using xlink:href

When necessary to preserve as a true reference the (that a part of the airspace border actually comes from a GeoBorder), then a gml:curveMember with a xlink:href attribute can be used. In this case, there shall be no child aixm:Curve or gml:Curve element. The GML standard requires a local reference, using a gml:id value. For compatibility reasons with previous AIXM versions and to satisfy the operational needs of the AI domain, it is also allowed to use a remote reference. The two solutions are detailed here.

11.2.2.1. Local reference to aixm:Curve

In this example, the gml:curveMember contains a local reference (xlink:href) to the gml:id value of an aixm:Curve that contains the relevant geo border points. The referred aixm:Curve is embedded into an ad-hoc GeoBorder SNAPSHOT TimeSlice. This GeoBorder TimeSlice is “ad-hoc” because it contains just the sub-set of point that need to be embedded.

```

<aixm:Airspace gml:id="urn.uuid.1965dd58-6898-4065-8f21-b1774c959bbb">
  ...
  <aixm:horizontalProjection>
    <aixm:Surface gml:id="S001" srsName="urn:ogc:def:crs:EPSG::4326">
      <gml:polygonPatches>
        <gml:PolygonPatch>
          <gml:exterior>
            <gml:Ring>
              <gml:curveMember>
                <gml:Curve gml:id="CUR001">
                  <gml:segments>
                    <gml:LineStringSegment interpolation="linear">
                      <!-- because the two consecutive points have the same latitude,
the first segment is encoded as a parallel (linear interpolation in EPSG:4326) -->

```

⁴ It would be better if it was possible to encode an aixm:Curve made of multiple segments and then give a gml:id to one of these segments. Then, use only that segment for the reference from the Airspace Border. Unfortunately, gml:id is not present in any of the segment types in GML (LineString, Geodesic, etc...). This could be an improvement suggestion for the next GML version. An (alternative) AIXM solution could be to allow multiple aixm:Curve in the definition of a GeoBorder. Thus, the encoding could refer to the gml:id of the Curve.


```

46.93333333</gml:posList>
                                <gml:posList>          40.05      45.88972222      40.05
                                </gml:LineStringSegment>
                                <gml:GeodesicString interpolation="geodesic">
                                  <gml:posList>40.05 46.93333333 39.42916667
                                      47.36333334</gml:posList>
                                </gml:GeodesicString>
                                </gml:segments>
                                </gml:Curve>
                                </gml:curveMember>
                                <gml:curveMember xlink:href="#CRV002" xlink:title="along the state border with
Islamic Republic of Iran"/>
                                <gml:curveMember xlink:href="#CRV003" xlink:title="along the state border with
Armenia">
                                </gml:curveMember>
                                </gml:Ring>
                                </gml:exterior>
                                </gml:PolygonPatch>
                                </gml:polygonPatches>
                                </aixm:Surface>
                                </aixm:horizontalProjection>
                                ...
                                </aixm:Airspace>

```

----- further down in the same file -----
 ...

```

<aixm:GeoBorder gml:id="urn.uuid.cee41f01-849e-44d1-9aaf-573580980c69">
  <gml:identifier codeSpace="urn:uuid:">cee41f01-849e-44d1-9aaf-573580980c69</gml:identifier>
  <aixm:timeSlice>
    <aixm:GeoBorderTimeSlice gml:id="NID2168343">
      <gml:validTime>
        <gml:TimeInstant gml:id="NID00056">
          <gml:timePosition>2011-09-15T00:00:00</gml:timePosition>
        </gml:TimeInstant>
      </gml:validTime>
      <aixm:interpretation>SNAPSHOT</aixm:interpretation>
      <aixm:name>AZERBAIJAN_IRAN_EXTRACT</aixm:name>
      <aixm:type>STATE</aixm:type>
      <aixm:border>
        <aixm:Curve gml:id="CRV002">
          <gml:segments>
            <gml:GeodesicString interpolation="geodesic">
              <gml:posList>39.42916667 47.36333334 39.426818 47.353277 39.405269
47.340623 39.370714 47.303951 39.317977 47.213494 39.304679 47.147725 39.253854 47.075486 39.209945 47.064339
39.138967 46.950699 39.160036 46.929733 39.135505 46.835985 39.113835 46.826968 39.110637 46.818902 39.104488
46.811639 39.084995 46.792338 39.079520 46.774177 39.063644 46.761250 39.035348 46.762985 39.015666 46.733063
39.019323 46.695507 38.991019 46.690236 38.987819 46.672322 38.930984 46.626817 38.906155 46.600158 38.885792
46.560942 38.885160 46.554943 38.8727778 46.54722222</gml:posList>
            </gml:GeodesicString>
          </gml:segments>
        </aixm:Curve>
      </aixm:border>
    </aixm:GeoBorderTimeSlice>

```

```

    </aixm:timeSlice>
  </aixm:GeoBorder>...

```

...

This solution is appropriate when the data is provided for direct consumption by a GML tool for display or other calculation purpose, but there is a need to also preserve a true reference (by xlink:href) to the GeoBorder. Obviously, it requires that both the Airspace and the referenced GeoBorder are included in the same file. It might be problematic to apply this solution in the case of WFS getFeature requests, because the referenced feature will not be present in the response.

Note also that the xlink:title attribute is used to provide a human readable identification of the GeoBorder that is referred, which can be used in printed documents (e.g. “along the state border with...”).

This solution does not imply the persistence of the gml:id value. It is still a temporary identifier, which enables linking the gml:curveMember with the aixm:Curve inside the file.

This direct link between gml:curveMember and aixm:Curve is a deviation from the general AIXM principle of having xlink:href associations towards the feature level only. However, this direct association is the only solution identified for really encoding geometry dependencies at the GML level. In a source database, the association can still be towards the GeoBorder itself (as detailed in the next section). Only for data export/import purpose the reference would be towards the aixm:Curve directly.

11.2.2.2. Abstract reference to remote feature

Note: The encoding described in this section is strongly discouraged on long term. However, for now, it is the only practical solution for keeping abstract references to GeoBorder curves. We strive to find a better alternative in the future.

The second possibility is to use an xlink:href towards a remote feature, as in the example below:

```

<aixm:Airspace gml:id="urn.uuid.1965dd58-6898-4065-8f21-b1774c959bbb">
  ...
  <aixm:horizontalProjection>
    <aixm:Surface gml:id="S001" srsName="urn:ogc:def:crs:EPSG:4326">
      <gml:polygonPatches>
        <gml:PolygonPatch>
          <gml:exterior>
            <gml:Ring>
              <gml:curveMember>
                <gml:Curve gml:id="CRV001">
                  <gml:segments>
                    <gml:LineStringSegment interpolation="linear">
                      <!-- because the two consecutive points have the same latitude,
the first segment is encoded as a parallel (linear interpolation in EPSG:4326) -->
                      <gml:posList> 40.05 45.88972222 40.05
46.93333333</gml:posList>

```

```

47.36333334</gml:posList>
    </gml:LineStringSegment>
    <gml:GeodesicString interpolation="geodesic">
      <gml:posList>40.05 46.93333333 39.42916667
    </gml:GeodesicString>
  </gml:segments>
</gml:Curve>
</gml:curveMember>
<!-- Here is the first reference to a GeoBorder.-->
<gml:curveMember xlink:href="urn:uuid:cee41f01-849e-44d1-9aaf-573580980c69"
xlink:title="along the state border with Islamic Republic of Iran"/>
<!-- Here is the second reference to a GeoBorder.-->
<gml:curveMember xlink:href="urn:uuid:2bc135fa-0a1a-451c-ad99-61ed3e194e1c"
xlink:title="along the state border with Armenia">
  </gml:curveMember>
</gml:Ring>
</gml:exterior>
</gml:PolygonPatch>
</gml:polygonPatches>
</aixm:Surface>
</aixm:horizontalProjection>
...
</aixm:Airspace>

```

The `xlink:href` value is a remote reference to the `gml:identifier` of the `GeoBorder` that includes the series lat/long positions used in the airspace horizontal projection definition. However, that `GeoBorder` is likely to be much longer, including also points that are beyond the portion used for the current airspace border definition. A native GML tool might identify this as an error, because the target `aixm:GeoBorder` is not a `gml:curveMember` legal child.

However, this solution is still considered appropriate for the situations when the AIXM encoding is used for exchanging data between a local system and a reference database. In such cases the data will not be directly used for graphical visualization and other spatial calculations. The recipient system would preserve this link and would eventually recuperate the `gml:pos` data of the referred feature in case this is necessary for further data use. This third solution is practically equivalent (with the same advantages and disadvantages) to the use of the “FNT” elements from the previous AIXM 4.5 version, which was not based on GML.

11.2.2.3. Temporal aspects for abstract references

When using abstract references, an important issue arises that must be dealt with regard to the temporal synchronisation of the data. Since an abstract reference always points to a complete feature rather than a single time slice, it is very important that there is no ambiguity in which time slice is meant. The referenced object must not “change” during the valid time of the reference. More exactly, the referenced object must have a time slice whose valid time covers completely the valid time of the referencing feature’s time slice.

For example, assume that the following `GeoBorder` feature is defined (only relevant parts are shown):

```

<aixm:GeoBorder ...>
  <gml:identifier ...>123...

```

```

<aixm:timeSlice>
  <gml:validTime>
    <gml:beginPosition>2013-05-02T00:00:00Z
    <gml:endPosition>2014-04-03T00:00:00Z
  ...
  <aixm:timeSlice>
    <gml:validTime>
      <gml:beginPosition>2014-04-03T00:00:00Z
      <gml:endPosition>2014-12-11T00:00:00Z

```

Then, the following reference is used (abstract to remote feature):

```

<aixm:Airspace ...>
  <aixm:timeSlice>
    <gml:validTime>
      <gml:beginPosition>2013-05-02T00:00:00Z
      <gml:endPosition>2014-04-03T00:00:00Z
    ...
    <gml:curveMember xlink:href="urn:uuid:123..."/>

```

In the above example, the Airspace and the referred GeoBorder TimeSlices have identical validity period. This is a valid and unambiguous encoding.

This following one is valid too since the validity of the Airspace TimeSlice is inside the validity period of the GeoBorder that it references:

```

<aixm:Airspace ...>
  <aixm:timeSlice>
    <gml:validTime>
      <gml:beginPosition>2013-08-22T00:00:00Z
      <gml:endPosition>2014-02-06T00:00:00Z
    ...
    <gml:curveMember xlink:href="urn:uuid:123..."/>

```

However, the following example is ambiguous and it shall be avoided:

```

<aixm:Airspace ...>
  <aixm:timeSlice>
    <gml:validTime>
      <gml:beginPosition>2013-08-22T00:00:00Z
      <gml:endPosition>2014-06-26T00:00:00Z
    ...
    <gml:curveMember xlink:href="urn:uuid:123..."/>

```

This is invalid because the reference is no longer unique. It points to the curve of the GeoBorder's first time slice as well as the curve of its second time slice. Moreover, since the remaining airspace curve must fit with the GeoBorder curve, it may be that one interpretation

of this reference is valid, while the other forms an invalid airspace geometry, e.g. by slightly moving the first curve point of the GeoBorder from one time slice to the next.

In that situation, the data provider is responsible for splitting the referencing Airspace time slice as needed:

```
<aixm:Airspace ...>
  <aixm:timeSlice>
    <gml:validTime>
      <gml:beginPosition>2013-08-22T00:00:00Z
      <gml:endPosition>2014-04-03T00:00:00Z
    ...
    <gml:curveMember xlink:href="urn:uuid:123..." />
  </aixm:timeSlice>
  <gml:validTime>
    <gml:beginPosition>2014-04-03T00:00:00Z
    <gml:endPosition>2014-06-26T00:00:00Z
  ...
  <gml:curveMember xlink:href="urn:uuid:123..." />
```

More generally, any “change” of the referenced object must also be marked as a “change” of the referencing feature, even if there are no changes otherwise.

12. AIXM GML Profile

12.1. Introduction

AIXM is built on a foundation of ISO standards, such as ISO 19107 (Spatial Schema), ISO 19108 (Temporal Schema), and ISO 19136 (Geography Markup Language, GML). These standards facilitate the interoperable exchange of spatial information in and between various application domains.

ISO 19107 and ISO 19108 define spatial and temporal types that can be used in an application schema (such as AIXM). These two standards target the conceptual level of schema modeling. A conceptual schema is implementation technology independent, meaning that it can be used as the basis for different implementations (based on XML, JSON, etc.). ISO 19136 provides realizations of the types defined in ISO 19107 as well as ISO 19108, targeting an XML implementation⁵.

It is important to recognize and understand the distinction between these two levels of application schema modeling (conceptual and implementation). This profile starts at the conceptual level, defining which conceptual types are relevant in general. It then also establishes the link to the XML implementation based upon ISO 19136. The structure supports adding of other implementations (for example based upon JSON) in the future.

12.2. Purpose of the profile

In order to satisfy the needs of multiple domains regarding the representation of spatial information, the amount of concepts and functionality covered by ISO 19107, ISO 19108 and ISO 19136 is far more than that required by the Aviation domain. There is therefore a need to ‘subset’ these specifications to create a ‘profile’ that contains only the features required for Aviation. This document provides a textual description of the profile.

12.3. Scope of the profile

The GML Profile for aeronautical data is limited to 2D elements and includes only the features necessary for encoding geometries of type point, line/curve and polygon/surface (outer boundaries only).

The following components of GML are out of scope for the AIXM GML Profile: Topology, Linear Referencing, Coverages.

This additional GML components necessary to the ISO 19139 (gco /basicTypes.xsd), which is imported in the GML and AIXM metadata, needs additional types. These are not included in this profile.

⁵ It also defines a number of conceptual types, for example ArcByCenterPoint, that are not defined in ISO 19107. The focus of ISO 19136, however, is on the realization of conceptual types in what is called an “implementation schema”.

12.4. Using the AIXM GML Profile

This section describes how to use the AIXM GML profile and how to declare that a particular application schema (such as the AIXM schema) adheres to that profile.

According to the [ISO 19136] GML Standard, 20.4 “*Instance documents of a profile shall be valid against the full GML schema*”. Further more [ISO 19136, 20.5] “*A GML application schema shall reference the full GML schema in the schemaLocation attribute of the <import> element*”.

Importantly whilst all GML profiles are defined in the gml namespace they must not be directly imported as a replacement for the full GML application schema. Doing so would redefine GML and break interoperability.

A GML application schema document conforming to one or more GML Profiles shall provide an appInfo annotation element <gml:gmlProfileSchema> for every profile in the root schema document <schema> element where the value is a schema location of the profile schema.

```
<annotation>
  <appinfo>
    <gml:gmlProfileSchema>http://www.aixm.aero/schema/GML_profile/gml321forAIXM.xsd
    </gml:gmlProfileSchema>
  </appinfo>
</annotation>
```

The annotation tag is annotation of the top level xsd:schema type as shown in the full example below. Note how the full gml.xsd is still imported.

12.5. XML Namespaces

This section uses a number of XML namespace prefixes; they are listed in Table 2. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 2: Prefixes and Namespaces

<i>Prefix</i>	<i>Namespace</i>
aixm	http://www.aixm.aero/schema/5.1
gml	http://www.opengis.net/gml/3.2
xlink	http://www.w3.org/1999/xlink
xs	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

12.6. AIXM GML Profile Requirements

12.6.1. Basic Types

Basic types are defined within ISO 19103 – Conceptual Schema Language. These basic types are grouped into two categories: Primitive Types and Complex Types.

Primitive Types are basic types for representing values such as strings, numeric values, etc. These types are defined in the W3C XML Schema specification which is imported by GML, therefore these types are not explicitly defined within the GML Profile but are fully supported.

Table 3 - Basic GML supported value types

<i>AIXM GML Profile Primitive Type</i>	<i>XML/GML Element</i>
Integer	xsd:integer
PositiveInteger	xsd:positiveInteger
NonNegativeInteger	xsd:nonNegativeInteger
Real	xsd:real
CharacterString	xsd:string
Date	xsd:date
Time	xsd:time
DateTime	xsd:DateTime
Boolean	gml:Boolean

In addition to the primitive types defined within the profile, URI shall be supported by the AIXM GML Profile as this is required by other types.

The gml:Boolean element extends the xsd:Boolean type to allow the inclusion of a nilReason.

Complex Types are basic types for representing measures, names, values from codelists/vocabularies (i.e. URI). The following complex types are supported in the AIXM GML Profile: UnlimitedInteger, NumericRange, UnitsOfMeasure, Codelists, NilReason. These complex types are defined in GML and shall therefore be defined within the AIXM GML Profile.

A unit of measurement is a well defined comparator for a magnitude. In the AIXM GML profile the units of measure shall be realized by the uom property value, which should reference to a value defined in a codelist register which provides the symbol, name and definition. See section [Units of measurement](#) for more details on how units of measure are encoded.

Codelists are similar to enumerations used to indicate a list of possible values. However, the list can be expanded, which means that a codelist is a union between the explicitly enumerated values and free text. Codelists can be either encoded directly within the XML Schema or can be maintained externally to the implementation schema within a code list dictionary.

A nilReason attribute is used to allow recording of an explanation for a void value or other exception. Depending on the situation gml:nilReasonType can be either one of the following enumerated values: inapplicable, missing, template, unknown, withheld, other or it can refer to a resource which describes the reason for the exception through the means of a URI.

12.6.2. AIXM Types

AIXM has one package (named “geometry”) which includes six types that are used to define geometric information in aeronautical features and objects. The types fall into three main categories: points, curves and surfaces. They are derived from types defined by ISO 19107. The following figure depicts the types and their relationships.

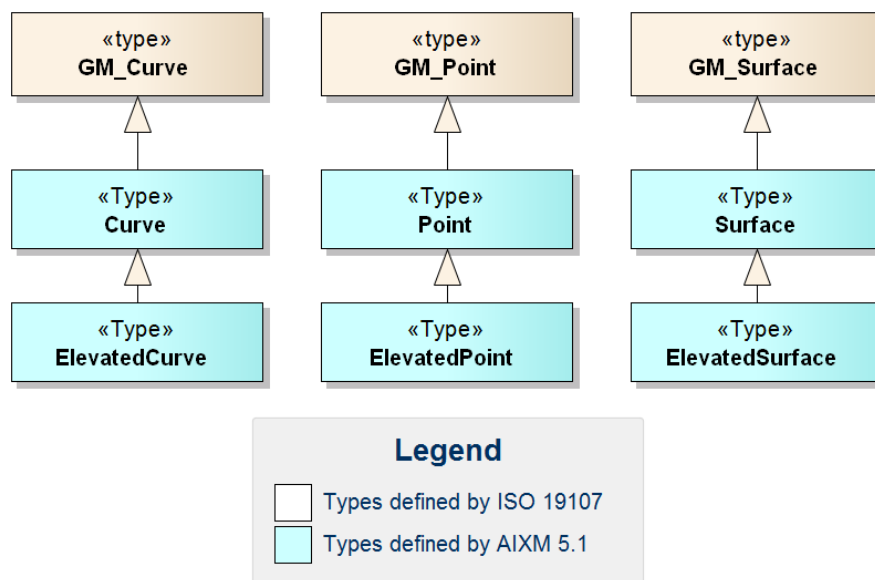


Figure 22 – Overview of AIXM Geometry Types

12.6.3. ISO 19107 and ISO 19136 (GML) Types - Overview

A number of types defined by ISO 19107 and ISO 19136 (GML) are relevant for the definition of spatial information in AIXM features. This section provides an overview of these types and the relationship between them. This information is intended to help AIXM developers to get a better understanding of which types can actually be used in the representation of points, curves and surfaces in AIXM features.

The basic ISO 19107 types for the definition of position information are shown in the following figure.

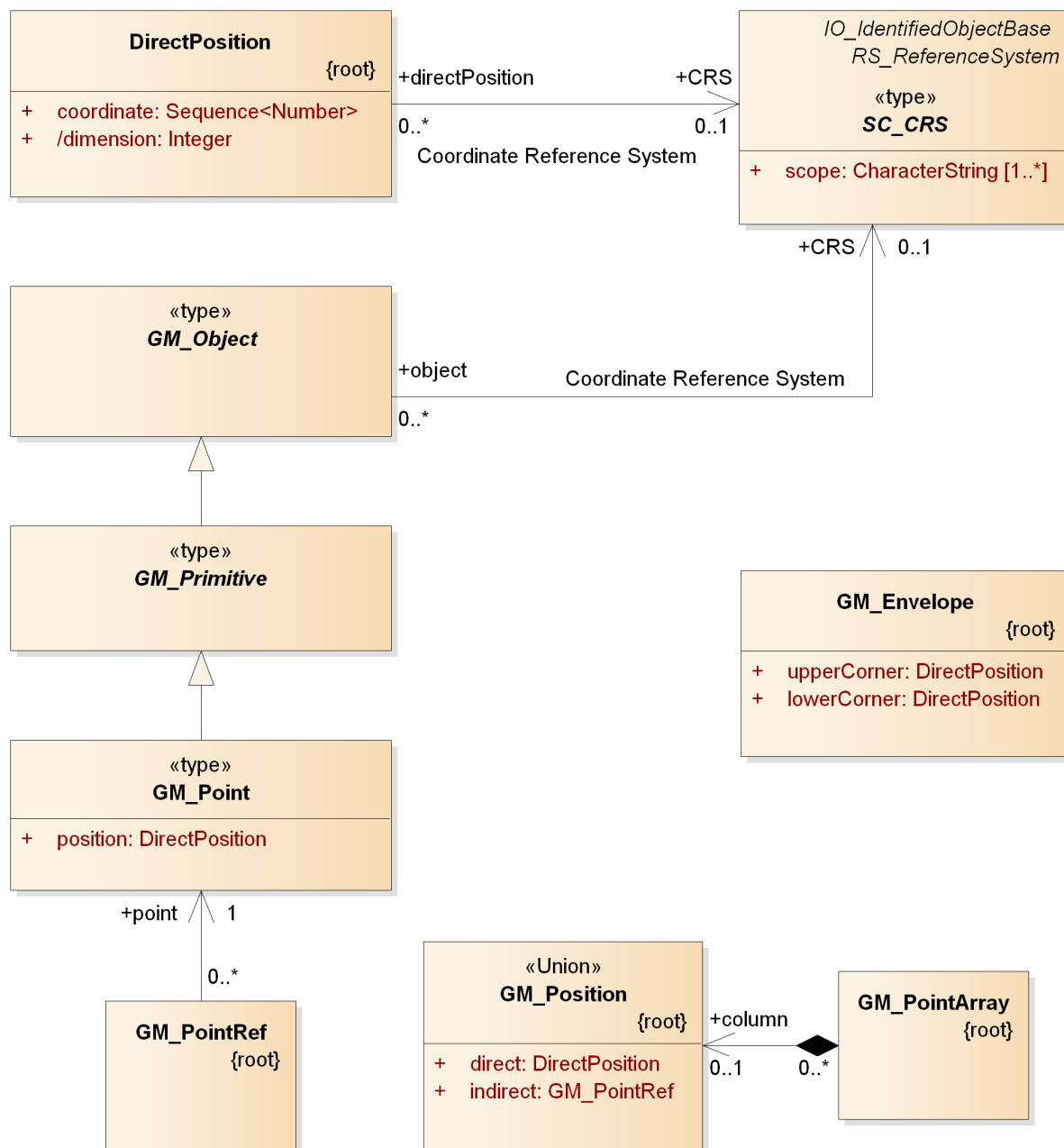


Figure 23: Overview of basic types from ISO 19107 for the definition of position information

A DirectPosition defines the coordinate values of a position in a given CRS. GM_Envelope can be used to define the bounding box of AIXM features and feature collections. GM_Object is the base type of all geometry types. GM_Point is derived from GM_Primitive⁶ and is the parent of the AIXM Point type (see section 9.5.1). GM_PointRef is used to include (the position of)

⁶ For the purpose of this profile, it is sufficient to understand that all geometric primitives (points, curves, surfaces) ultimately – via GM_Primitive and other types - derive from GM_Object.

an existing point by reference (in XML usually via xlink). GM_Position represents a choice between a position given as direct value or given indirectly via a point reference. The GM_PointArray represents a sequence of positions and therefore is often found in the definition of curve segments.

Curves are used in the representation of route segments and airspace boundaries, among others. GM_Curve is the parent of the AIXM curve types (see section 9.5.1). There are a number of types that can be used to define the segments of a curve. Most of these types are defined in ISO 19107. However, there are some special curve segment types which have been added by ISO 19136. The curve segment types relevant for the definition of a curve in AIXM instances are shown in the following figure.

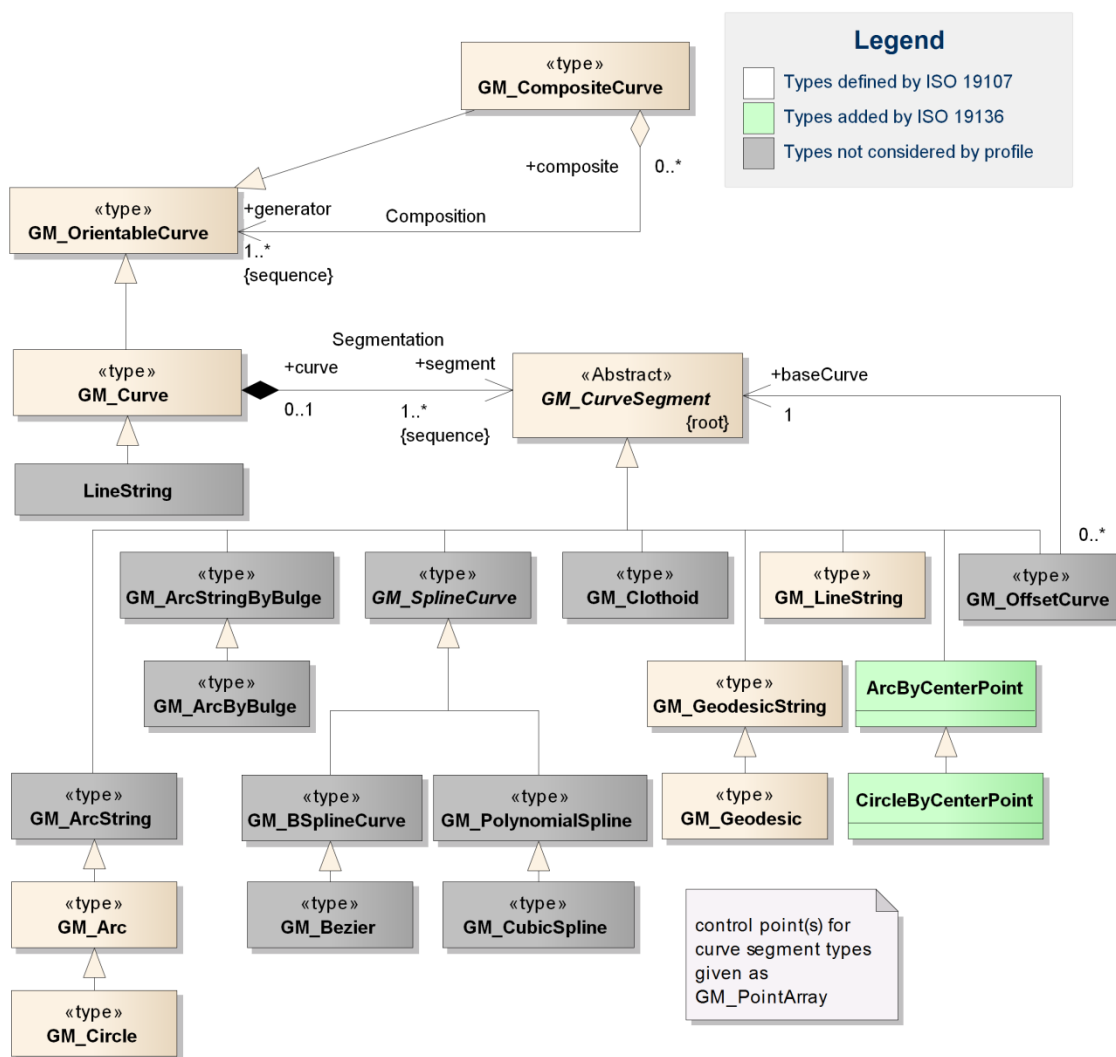


Figure 24: Overview of types from ISO 19107 / ISO 19136 used for the definition of a curve in spatial properties of AIXM features – types irrelevant for the profile marked with grey background

Note: the *ArcString*, *ArcStringByBulge*, *ArcByBulge*, *BSplineCurve*, *Bezier*, *CubicSpline*, *Clothoid* and *OffsetCurve* types are additional curve segment types. However, the use of these types in GML geometries for aviation data is currently not foreseen by this document. Issues have been identified for the use of *OffsetCurve*; for

more detailed information about these issues, see Annex E. The *LineString* type defined by GML provides a shortcut to represent curves that are only composed of line segments with linear interpolation. However, the same can be achieved using *GM_Curve* with *GM_LineString* members. In addition, the use of geodesics for straight lines is encouraged by this profile because of less ambiguity in relation with a CRS. This profile therefore does not consider the use of GML *LineString*. Consequently, a detailed documentation of all these types is not provided. This may change in future versions of this document.

Just as AIXM curve types are based on *GM_Curve*, AIXM surface types are based on *GM_Surface* – see following figure.

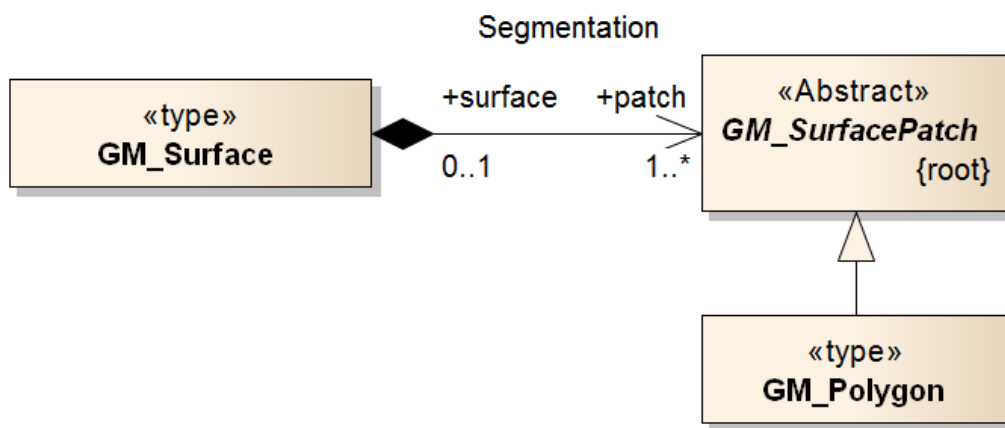


Figure 25: Overview of types from ISO 19107 that are relevant for the definition of a surface in spatial properties of AIXM features

In AIXM, only the *GM_Polygon* is relevant for the definition of surface patches.

Note: other possible subtypes of *GM_SurfacePatch* in GML applications are *Rectangle* and *Triangle* as well as parametric curve surfaces (such as *Cone*, *Cylinder* and *Sphere*). In the Aviation domain, there is no business need to distinguish between a generic polygon and the particular case of a triangle or rectangle. Furthermore, the use of parametric curve surfaces in AIXM surface geometries is not foreseen by this document.

The model of *GM_Polygon* is shown in the following figure. As we can see, the boundary of a polygon is defined by one or two rings, which are composite curves. This creates a relationship between the curve types shown in Figure 24 and the types shown in Figure 25 and Figure 26.

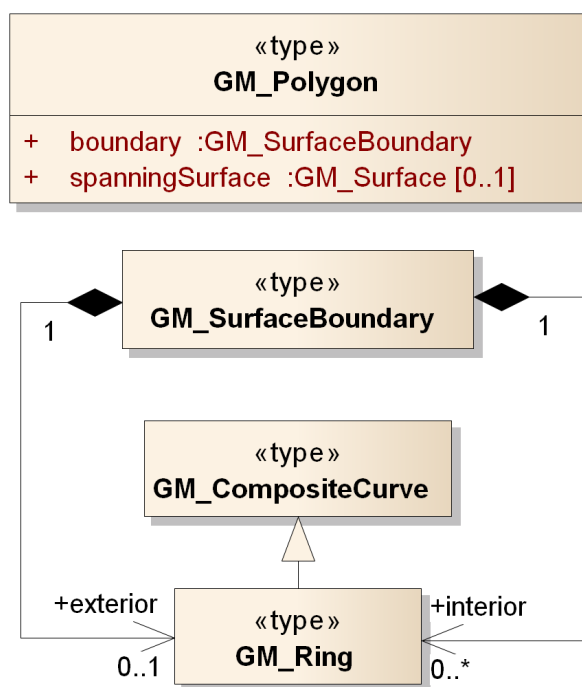


Figure 26: Overview of types from ISO 19107 that are relevant for the definition of a GM_Polygon in spatial properties of AIXM features

Note: the *LinearRing* type (defined by GML) is a child of *GM_Ring* (see section 9.6.22) and thus is a valid substitute for it. However, it only supports linear interpolation. This may be useful in the particular case where a surface boundary/ring is expressed as a sequence of rhumbines (at the moment requiring a “Mercator” projection – for further details see section 5.2.2). However, the same can be achieved using a Ring with *LineStringSegments*. In order to keep the AIXM GML Profile as slim as possible, the use of *LinearRing* is therefore not foreseen by this document.

12.6.4. ISO 19108 and ISO 19136 (GML) Types

For temporal types the GML profile for AIXM supports the ISO 19108 *TM_GeometricPrimitive* type and its subtypes (*TM_Instant* and *TM_Period*). This section provides an overview of these types.

The following figure depicts the *TM_Period* and *TM_Instant* types included in the profile and the relationship between them.

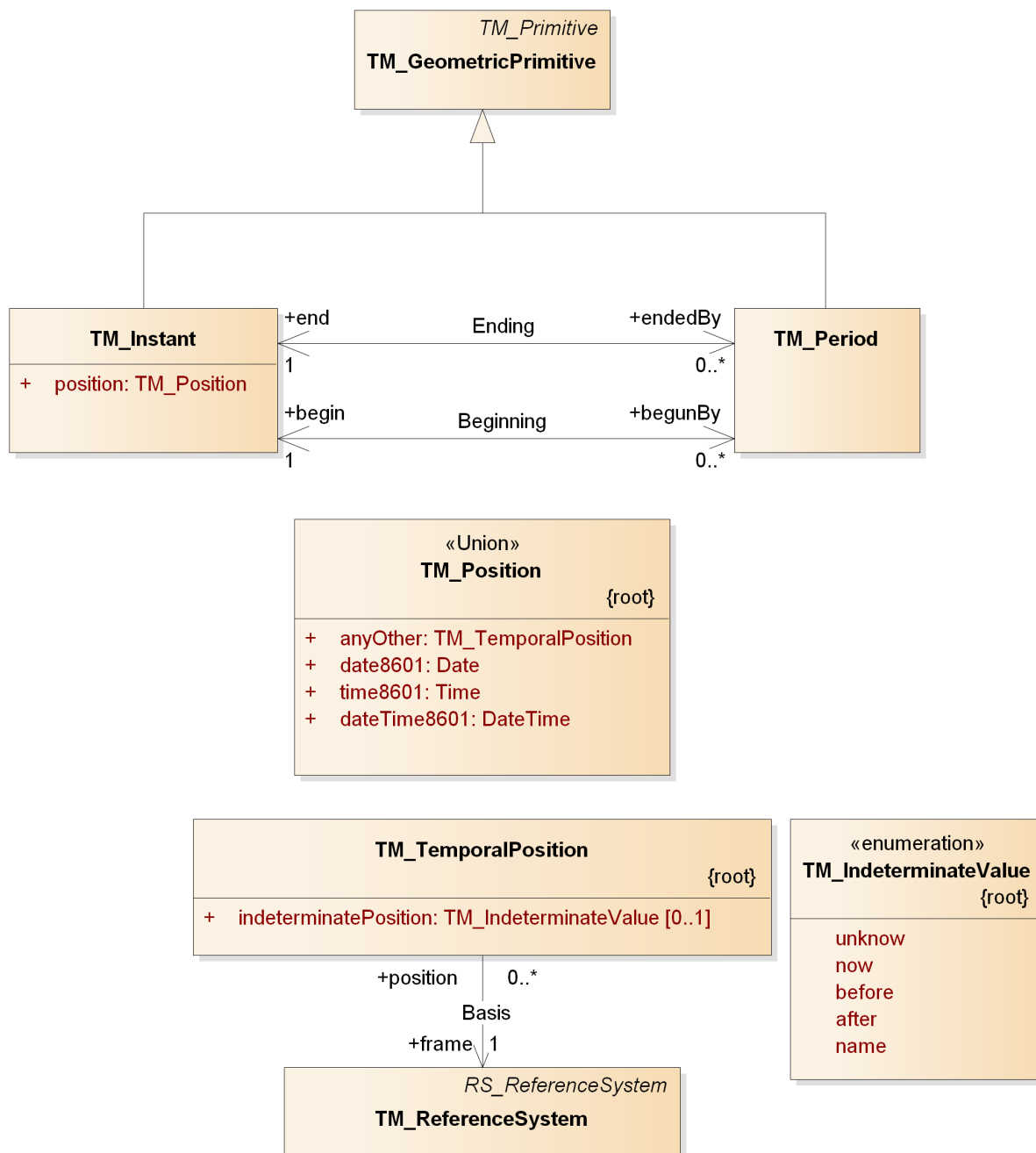


Figure 27: Overview of types from ISO 19108 that are relevant for the definition of time properties of AIXM features

TM_Period and TM_Instant derive from TM_GeometricPrimitive.[1] TM_Position is a union that consists of a Date, Time, DateTime or TM_TemporalPosition. The former three are defined by ISO 19103, and are expressed as character strings complying with ISO 8601 (referenced to the Gregorian calendar and UTC). TM_TemporalPosition is used to describe temporal positions referenced to any temporal reference system, including but not limited to the

Gregorian calendar and UTC. It also supports expression of an indeterminate position (“unknown”, “now”, etc).

A temporal geometry shall be associated with a temporal reference system through the frame attribute that provides a URI reference which identifies a description of the reference system. Following ISO 19108, the Gregorian calendar with UTC is the default reference system, but others may also be used.

TM_Instant is used to describe a SNAPSHOT time slice, because by definition a SNAPSHOT is a kind of Time Slice that describes the state of a feature at a time instant. TM_Period is used to describe the time range of a Time Slice.

The XML Schema implementation of the time geometric types from ISO 19108 that are relevant for AIXM is documented in the following sections.

12.6.5. Use of xsi:type is forbidden

AIXM encodings shall not make use of the xsi:type attribute. That attribute would allow for using arbitrary element names in any namespace. For example, the following XML snippet would be valid according to the schema:

```
<myns:Bogus xsi:type="aixm:PointType" srsName="..." gml:id="...">
  ...
</myns:Bogus>
```

This is equivalent to:

```
<aixm:Point srsName="..." gml:id="...">
  ...
</aixm:Point>
```

Allowing for xsi:type would add an additional source of errors, and an unnecessary complexity to implementations. Moreover, it is strongly discouraged since GML 3.2. The revision notes of the GML 3.2.1 specification state:

6.4 Element substitutability vs type derivation

In GML 3.1, requirements were often focused on content model definitions (e.g. “feature-type types must be ultimately derived from gml:AbstractFeatureType”) which are an artefact of the W3C XML Schema language and are thus normally invisible. The requirements did not focus on components that can appear in instance documents, i.e. elements representing features whose name is the feature type and which are substitutable for gml:AbstractFeature. This has been changed.

Note that this does not weaken the requirement since XML Schema rules require the type derivation in order that the substitution group is valid.

[\(Revision Notes for OpenGIS® Implementation Specification: Geographic information - Geography Markup Language Version 3.2.1, page 13\)](#)

Since `xsi:type` cannot be removed at the XSD schema level, a dedicated business rule will be defined for this purpose.

12.7. AIXM GML Profile Definition

The following tables provide an overview of the conceptual types that are included in the profile and their XML Schema implementation. The detailed definition of each component of the Aviation GML Profile is provided in Annex G.

Table 4 – AIXM Conceptual Types and the relevant XSD Implementation to document

<i>AIXM Conceptual Type</i>	<i>AIXM XSD Implementation (Element and Type)</i>
Point	aixm:Point, -Type
ElevatedPoint	aixm:ElevatedPoint, -Type
Curve	aixm:Curve, -Type
ElevatedCurve	aixm:ElevatedCurve, -Type
Surface	aixm:Surface, -Type
ElevatedSurface	aixm:ElevatedSurface, -Type

Table 5 – ISO 19107 conceptual types and the relevant XSD implementation to document

<i>ISO 19107 Type</i>	<i>ISO 19136 / GML Implementation (Element and Type)</i>
DirectPosition	gml:pos, gml:DirectPositionType
GM_Object	gml:AbstractGeometry, -Type
GM_Point	gml:Point, -Type
GM_Envelope	gml:Envelope, -Type
GM_PointRef	gml:pointProperty, gml:PointPropertyType
GM_Position	<i>Defined in GML as a group, not as an element/type:</i> gml:geometricPositionGroup
GM_PointArray	gml:posList, gml:DirectPositionListType
-	gml:AbstractCurve, -Type
GM_Curve	gml:Curve, -Type
GM_CurveSegment	gml:AbstractCurveSegment, -Type
-	gml:ArcByCenterPoint, -Type
-	gml:CircleByCenterPoint, -Type
GM_Arc	gml:Arc, -Type
GM_Circle	gml:Circle, -Type
GM_GeodesicString	gml:GeodesicString, -Type
GM_Geodesic	gml:Geodesic, -Type
GM_LineString	gml:LineStringSegment, -Type
GM_Surface	gml:Surface, -Type

GM_SurfacePatch	gml:AbstractSurfacePatch, -Type
GM_Polygon	gml:PolygonPatch, -Type
-	gml:AbstractRing, -Type
GM_Ring	gml:Ring, -Type
GM_OrientableCurve	gml:OrientableCurve, -Type
GM_CompositeCurve	gml:CompositeCurve, -Type

Table 6 - ISO 19108 conceptual types and the relevant XSD implementation document

<i>ISO 19108 Type</i>	<i>ISO 19136 / GML Implementation (Element and Type)</i>
TM_Instant	gml:TimeInstant, -Type
TM_Period	gml:TimePeriod, -Type

Annex A - List of CRS used for aeronautical data

The WGS-84 coordinate reference system shall be used for aeronautical data, based on the very strict ICAO requirements in regard to this aspect. However, some data might still be available only with reference to legacy coordinate reference systems, or to local reference frames, that are compatible with the WGS-84 requirements. These are provided in the following list, by their srsName. This list was established taking into consideration the previous AIXM 4.5 version list, which was in turn based on a list contained in the ARINC 424 specification.

Left-handed CRS

Ellipsoidal 2D CS. Axes: latitude, longitude. Orientations: north, east. UoM: degree

<i>srsName</i>	<i>Description</i>
urn:ogc:def:crs:EPSG::4326	WGS-84 (GRS-80)
urn:ogc:def:crs:EPSG::4258	ETRS-89
urn:ogc:def:crs:EPSG::4322	WGS-72
urn:ogc:def:crs:EPSG::4230	European 1950 (ED 50)
urn:ogc:def:crs:EPSG::4668	European 1979 (ED 79)
urn:ogc:def:crs:EPSG::4312	Austria NS
urn:ogc:def:crs:EPSG::4215	Belgium 50
urn:ogc:def:crs:EPSG::4801	Berne 1873
urn:ogc:def:crs:EPSG::4149	CH-1903
urn:ogc:def:crs:EPSG::4326	Danish GI 1934
urn:ogc:def:crs:EPSG::4275	Nouvelle Triangulation de France (Greenwich Zero Meridian)
urn:ogc:def:crs:EPSG::4746	Potsdam
urn:ogc:def:crs:EPSG::4121	GGRS 87 (Greece)
urn:ogc:def:crs:EPSG::4658	Hjorsey 55 (Iceland)
urn:ogc:def:crs:EPSG::4299	Ireland 65
urn:ogc:def:crs:EPSG::4806	Rome (Italy) 1940
urn:ogc:def:crs:EPSG::4326	Nouvelle Triangulation de Luxembourg

urn:ogc:def:crs:EPSG::4277	Ordnance Survey of Great Britain 36
urn:ogc:def:crs:EPSG::4207	Portugal DLX
urn:ogc:def:crs:EPSG::4274	Portugal 1973
urn:ogc:def:crs:EPSG::4740	PZ-90 (Russian Federation)
urn:ogc:def:crs:EPSG::4313	RNB 72 (Belgium)
urn:ogc:def:crs:EPSG::4124	RT90 (Sweden)
urn:ogc:def:crs:EPSG::4267	North American 1927
urn:ogc:def:crs:EPSG::4269	North American 1983

Right-handed CRS

Ellipsoidal 2D CS. Axes: longitude, latitude. Orientations: east, north. UoM: degree

<i>srsName</i>	<i>Description</i>
urn:ogc:def:crs:OGC:1.3:CRS84	WGS-84 (GRS-80)

Annex B - ArcByCenterPoint Interpretation Summary

Introduction

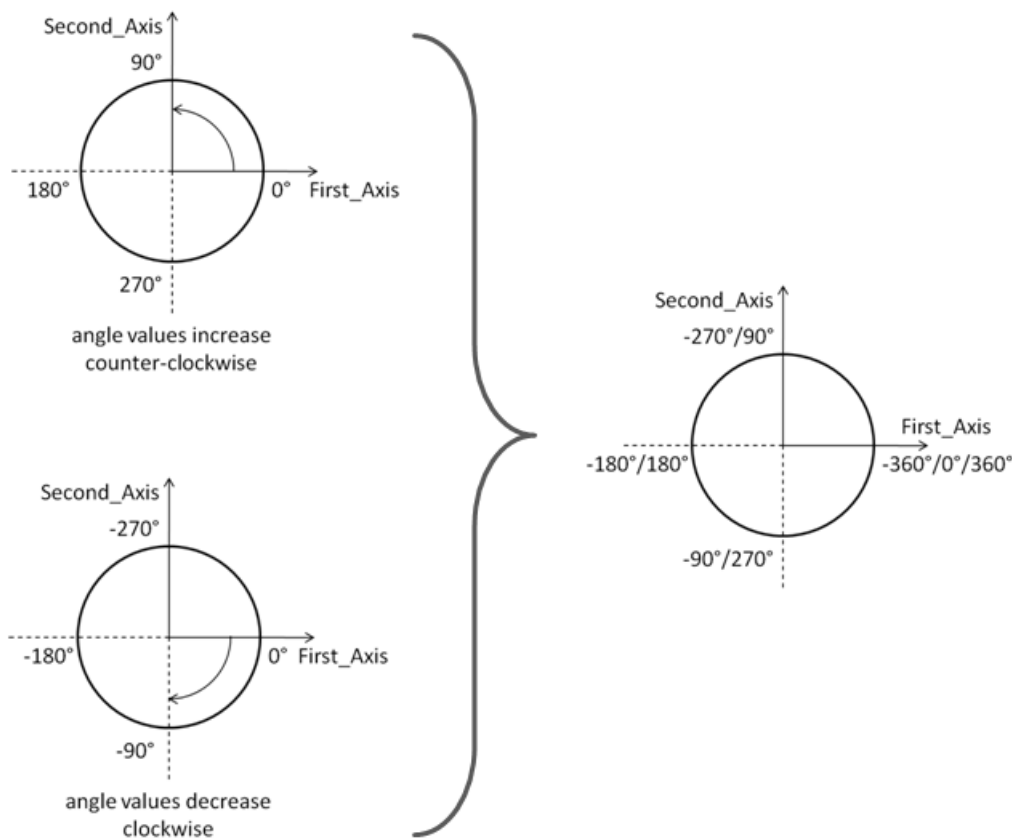
This annex contains the summary of discussions on the correct interpretation of ArcByCenterPoint.

Angle Measuring Convention in GML

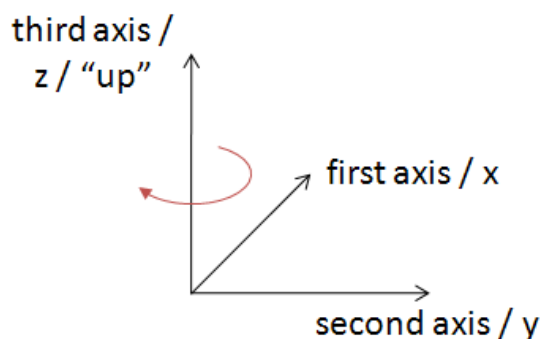
GML implements the semantics of ISO 19107. As such, the semantics for an angle encoded in GML is given by ISO 19107 clause 6.3.12.2: *"In this variant of Bearing usually used for 2D coordinate systems, the first angle (azimuth) is measured from the first coordinate axis (usually north) in a counterclockwise fashion parallel to the reference surface tangent plane."*

General Direction of Increasing and Decreasing Angle Values

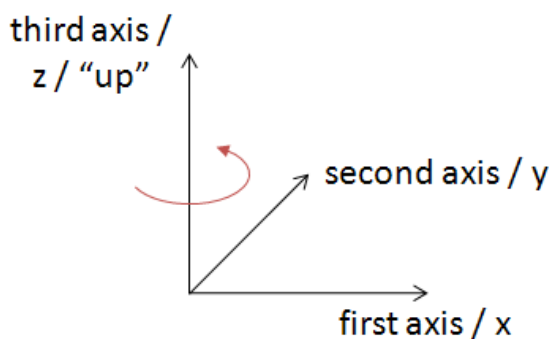
ISO 19107 thus defines in which direction angle values increase and in which direction they decrease - see the following diagram:



Depending upon the "up" and the orientation of the first two axes, we have a left-handed or right handed coordinate system (see [wikipedia](http://en.wikipedia.org/wiki/Right-handed_coordinate_system) and the following figure).



left-handed system



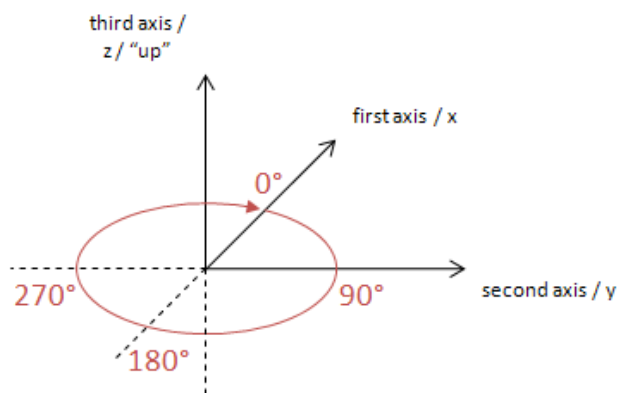
right-handed system

(Note: you can try this yourself with the following convention: thumb=x, index=y, middle=z)

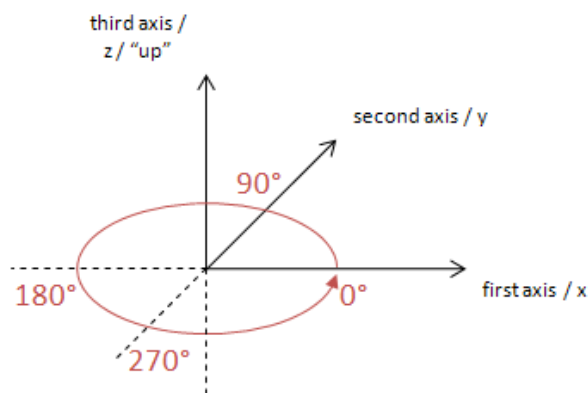
Left-handed systems have a clockwise rotation from first to second axis (positive axis to positive axis). Right-handed systems have a counterclockwise rotation from first to second axis (positive axis to positive axis).

Note: the direction of "clockwise" and "counterclockwise" therefore depends on the "up", not the order of the first two axes in the coordinate system. Swapping the order of the first and second axis does not affect the direction of clockwise, just the direction of the rotation between the two axes.

In a left-handed system, angle values increase in clockwise direction. In a right-handed system, angle values increase in counterclockwise direction.



left-handed system



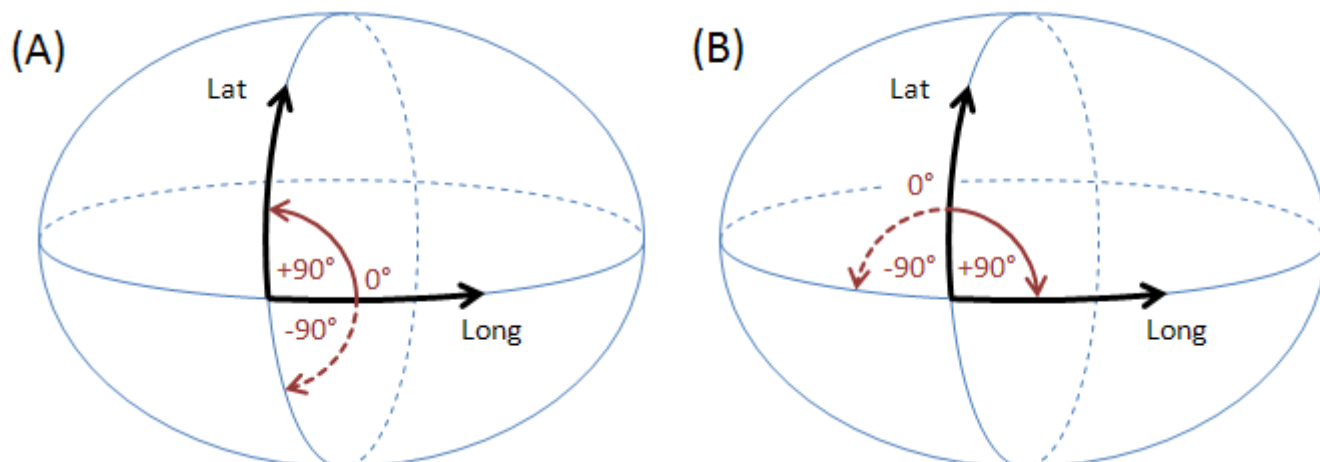
right-handed system

Note: The planes in geographic information are representations of the ground, and as such have a "natural" up direction (usually represented by a normal vector). In a 3D system, "up" is given by the definition of the third axis in the chosen CRS. In a 2D system "up" is usually implied.

Angle Measurement in Different Coordinate Systems

Apparently the way that angle values are expressed heavily depends upon the axis order defined by the coordinate system that is used.

The following diagram shows how angles are measured in WGS 84 2D with different coordinate systems:



(A) GeodeticCRS: urn:ogc:def:crs:OGC:1.3:CRS84 (right-handed)

- Datum: WGS84
- Ellipsoidal 2D CS.
 - Axes: (1st) longitude, (2nd) latitude.
 - Orientations: east, north. UoM: degree

(B) GeodeticCRS: urn:ogc:def:crs:EPSG::4326 (left-handed)

- Datum: WGS84
- Ellipsoidal 2D CS.
 - Axes: (1st) latitude, (2nd) longitude.
 - Orientations: north, east. UoM: degree

The order of the axes determines where 0° is located.

Definition: The 0° angle is located on the positive part of the coordinate system's first axis.

Note: As explained in the previous section, the order of the first two axes and their location "on the ground" in combination with the "up" direction defines in which directions the angle values increase/decrease.

Direction of an Arc

The direction of a line is always from its start to its end. The same is true for arcs defined by a start and end angle.

Definition: if the start angle is smaller than the end angle then the arc direction is the direction in which the angle values increase.

Note: depending upon the coordinate system that applies to a given ArcByCenterPoint, this results in a clockwise (left-handed system) or counter-clockwise (right-handed system) directed arc.

Example A: CRS urn:ogc:def:crs:OGC:1.3:CRS84 has long/lat axis order. Because the orientation of the axes on the earth surface is the same as that of the first_axis (=x) / second_axis (=y) coordinate system assumed by ISO 19107, angle values increase in counter-clockwise direction. Thus, if the start angle of an ArcByCenterPoint in this CRS is smaller than its end angle, then the direction of the arc is counter-clockwise; otherwise it is clockwise.

Example B: CRS urn:ogc:def:crs:EPSG::4326 has lat/long axis order. Even though the axes have the same orientation on the earth surface as in example A, the axis order of the coordinate system is different. In the EPSG:4326 CRS, the "counter-clockwise" convention from ISO 19107 actually corresponds to a clockwise rotation, because the first_axis (=x) / second_axis (=y) coordinate system assumed by ISO 19107 is mirrored (reflected) through the x=y diagonal when transposing the coordinate system used by EPSG:4326 on the surface of the Earth. Because of this, if the start angle of an ArcByCenterPoint in EPSG:4326 is smaller than its end angle, then the direction of the arc on the earth surface is clockwise; otherwise it is counter-clockwise.

In other, more mathematical words: angle values increase in counter-clockwise direction if the coordinate system of the CRS used for an ArcByCenterPoint can be transformed into the coordinate system implied by ISO 19107 through a simple rotation. Whenever a reflection across the x=y line is needed (+ any rotation) to achieve the axis orientation of the given CRS's coordinate system then the angle values of an arc increase in clockwise direction.

Arc encoding examples in a left-handed CRS system

The following diagrams exemplify the conventions defined above, considering a left-handed CRS, such as EPSG:4326. They show the arc on a line representing a flattened circle first, followed by its representation in the general case.

Notes:

- as angle values have an unlimited range, the flattened circle is a theoretically unlimited line.
- each angle (from $[0^\circ, 360^\circ]$) on the circle has an unlimited number of representations: $angle + X * 360^\circ$, *X being an integer*
- to define arcs on a circle, it is important that the difference between end and start angle is smaller than 360° : $|end-angle - start-angle| < 360^\circ$ (else the result would be an arc that is greater than or equal to 360° and looks like a circle)
- computing the length of an arc can be performed as follows: $|startAngle - endAngle| / 180^\circ * radius * pi$ - the definition conforms to this formula

The according ArcByCenterPoint would look like the following (with start and end angles as used in the examples):

```
<gml:ArcByCenterPoint numArc="1">
  <gml:pointProperty>
    <gml:Point gml:id="ID1">
      <gml:pos>0 0</gml:pos>
```



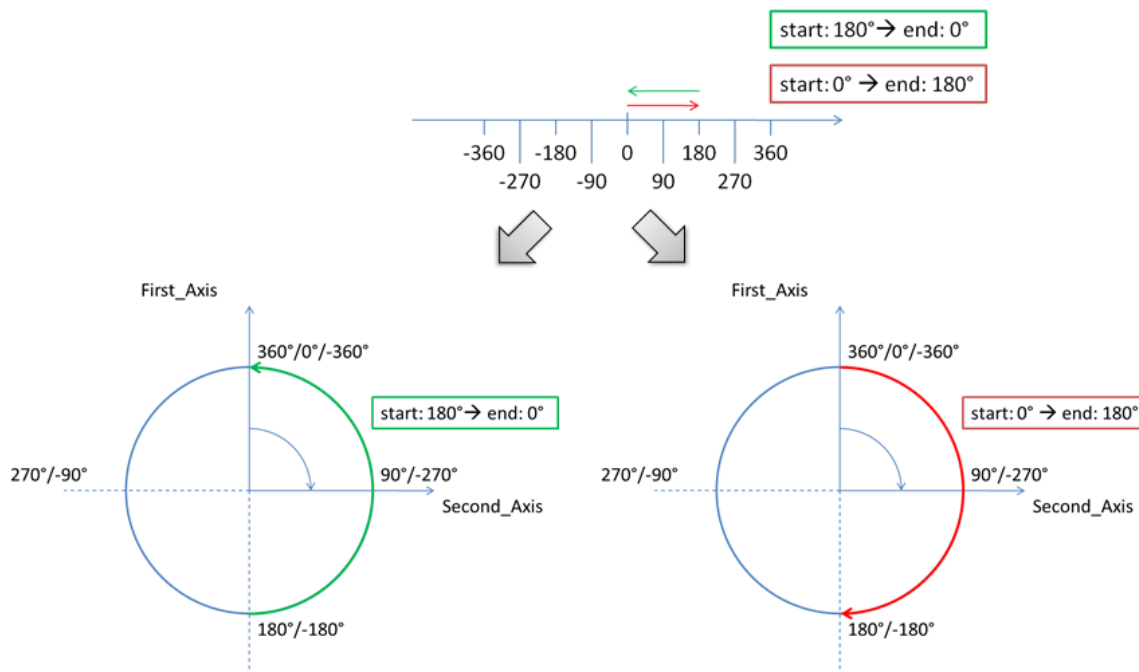
```

    </gml:Point>
  </gml:pointProperty>
  <gml:radius uom="m">1</gml:radius>
  <gml:startAngle uom="deg">actual_start_value</gml:startAngle>
  <gml:endAngle uom="deg">actual_end_value</gml:endAngle>
</gml:ArcByCenterPoint>

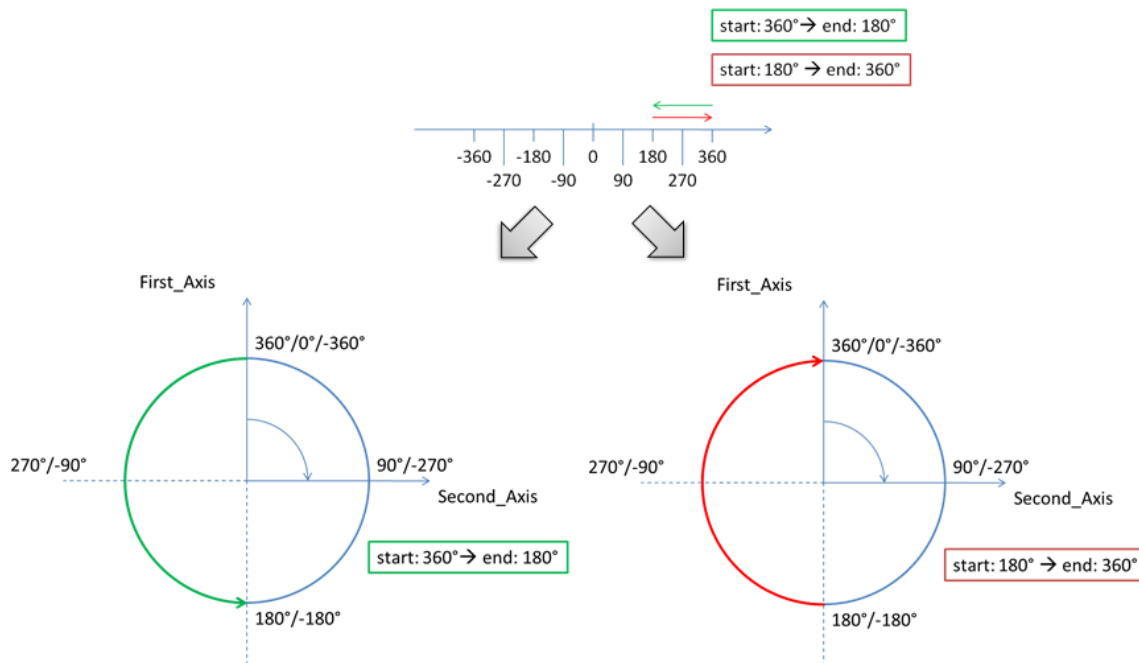
```

Note that the spatial reference system - and thus the coordinate system - to be used cannot be directly defined in the ArcByCenterPoint. It is the same as that of the Curve that the ArcByCenterPoint is a segment of.

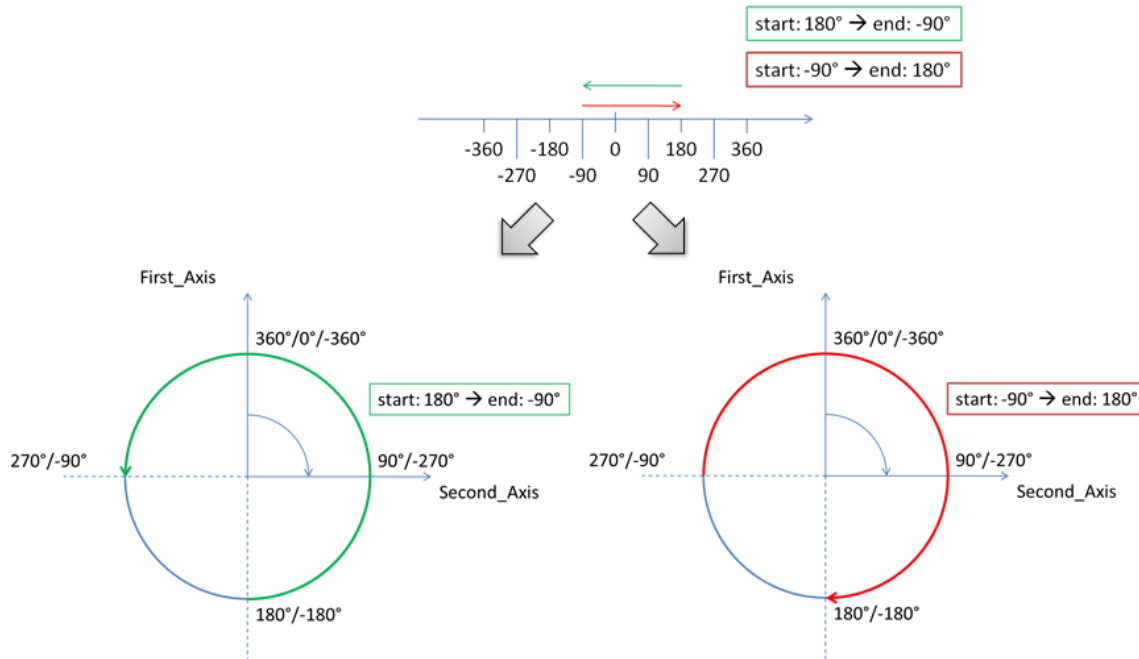
example - one



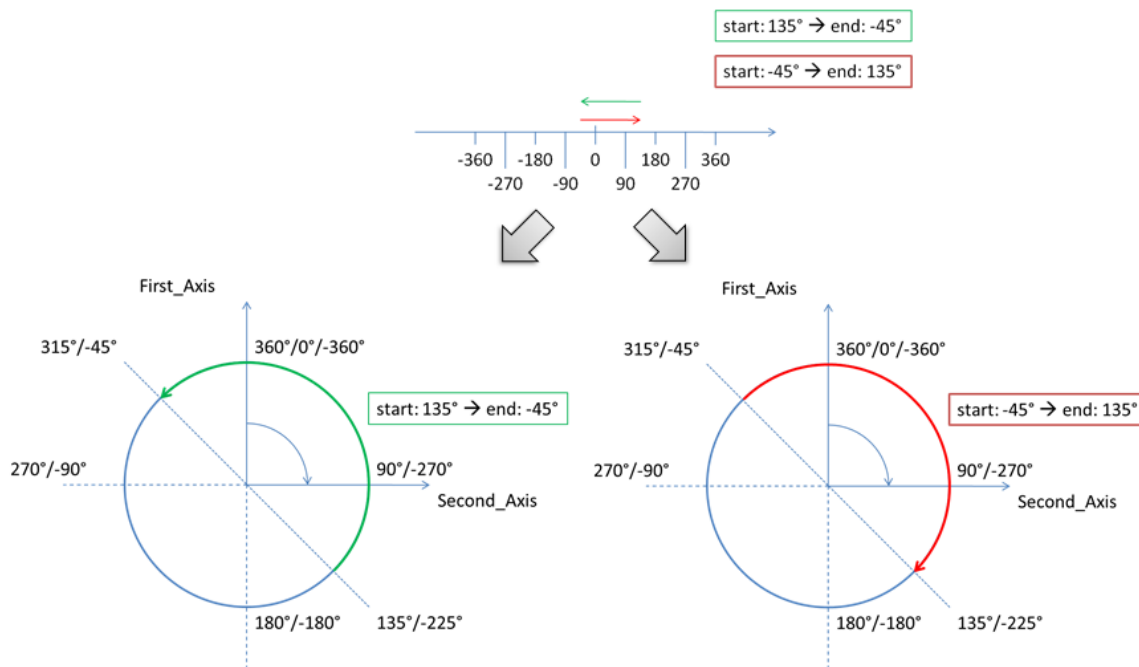
example - two



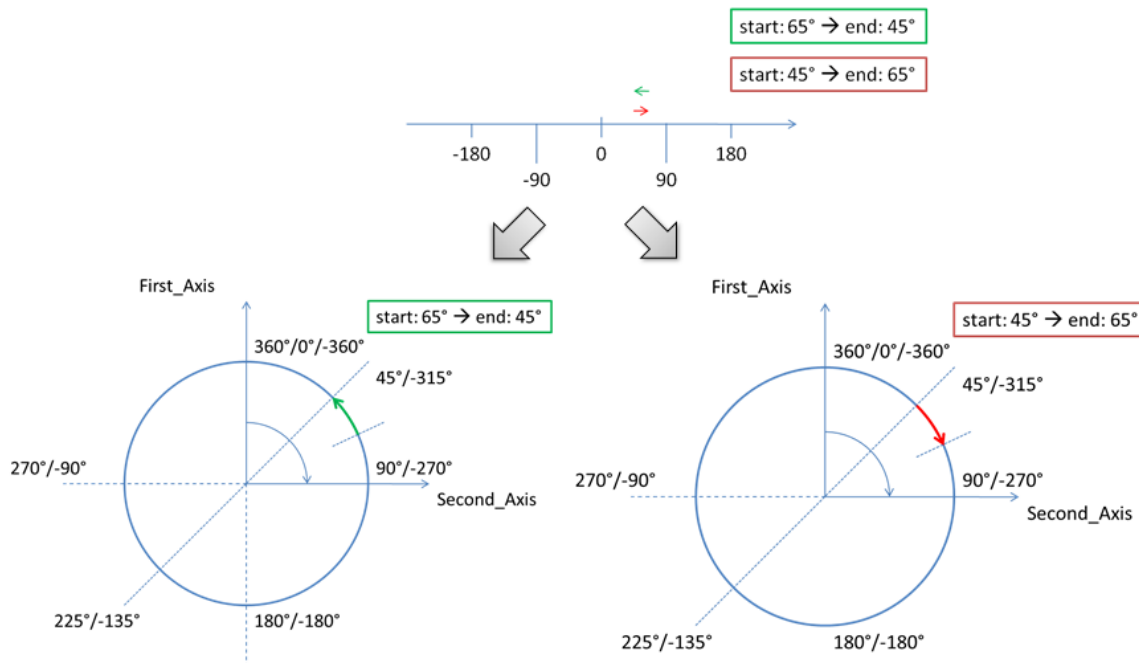
example - three



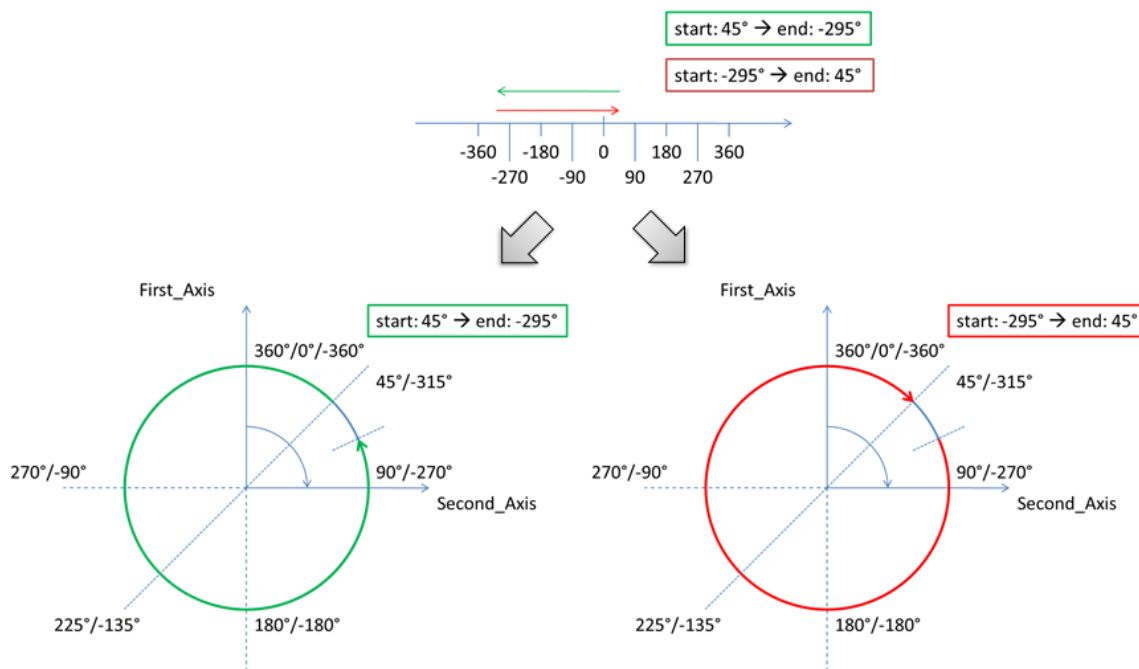
example - four



example - five



example - six



Arc interpolation

Typically, there are no statements in AI sources, such as State AIP, about how arcs and circles should be interpolated. Such arcs are usually designed using a map (into a specific projection, which is not communicated). Thus, they would represent points of equal distance from the arc centre, on that specific projection. Considering that the airspace design processes always include buffers in the definition of an area that contains a dangerous activity, the relative imprecision of an arc or circle interpolation is not operationally significant. However, it is important to have a common interpretation of how arcs and circles should be interpolated when defined in GML “by centre point”.

Therefore, it is recommended to interpolate ArcByCentrePoint through points situated at equal geodesic distance (the radius of the arc) from the arc centre. This makes the interpolation independent of the CRS used.

Annex C - Mapping for ArcByCenterPoint

This Annex indicates how to calculate the start/end angles of an ArcByCenterPoint when the information provided by the (official) source is in the form of start/end positions, centre and radius. The situation presented in Figure C.1 exemplifies a part of an Airspace boundary that is a sequence of:

- a straight segment (P1-P2)
- followed by a clockwise arc from P2 to P4 with the centre in P3
- followed again by a straight line to P5.

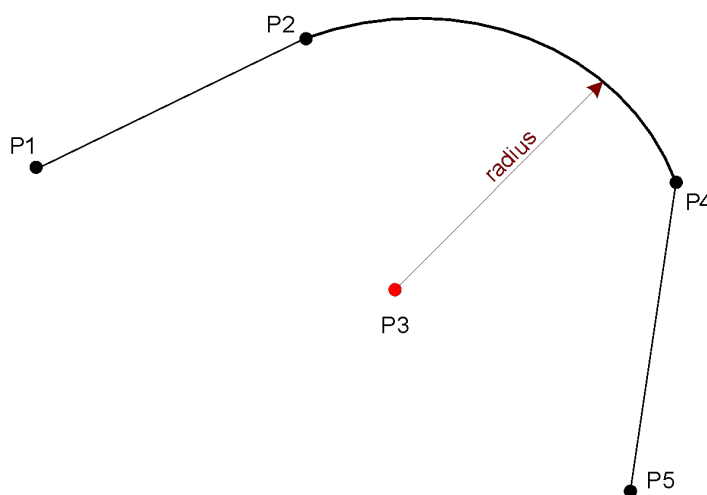


Figure C.1 - Mapping arc by start/end point into ArcByCenterPoint

Sometimes, trying to draw the arc using the exact values provided in the official sources for the centre point, radius and positions P2 and P4 does not work. The declared radius may be significantly ⁷different from the calculated distances from the centre P3 to points P2 or P4. An objective of the GML encoding is to preserve as much as possible the original (official) information. Frequently, the centre of the arc corresponds to the position of a physical object (such as a Navaid) and the radius value is meant to prevent airspace users from crossing the airspace boundary. Therefore, it is recommended that the original centre and radius data is used directly for the ArcByCenterPoint in the AIXM/GML encoding. The startAngle shall be calculated using the vector P3-P2 and the endAngle shall be calculated using the vector P3-P4. The resulting arc is represented in Figure C.2. This shows that it is possible that the resulting GML arc does not align at points P2 and P4, because of the imprecision of the original centre and/or radius data.

⁷ For example, in the European AIS Database, a difference of more than 1% is considered significant and raised as an error, that requires corrective action.

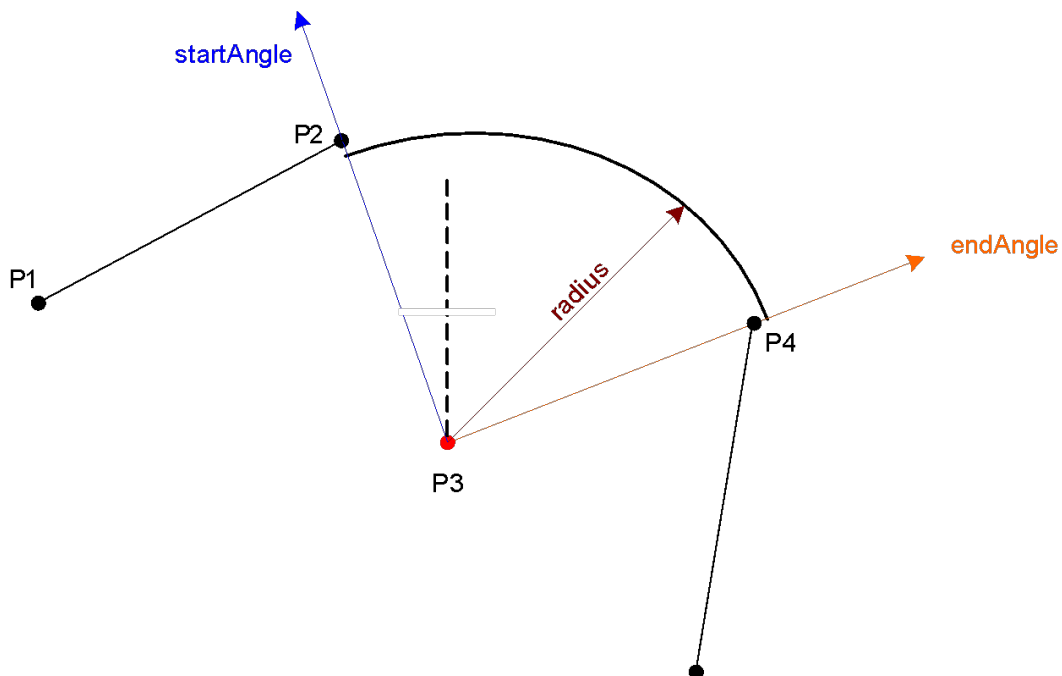


Figure C.2 - Calculating startAngle and endAngle for ArcByCenterPoint

If the potential misalignment of the arc (as indicated in Figure C.2) is a problem for an application, then it is suggested that, for example, the original centre and radius data are replaced with calculated values, by that application, as indicated in Figure C.3.

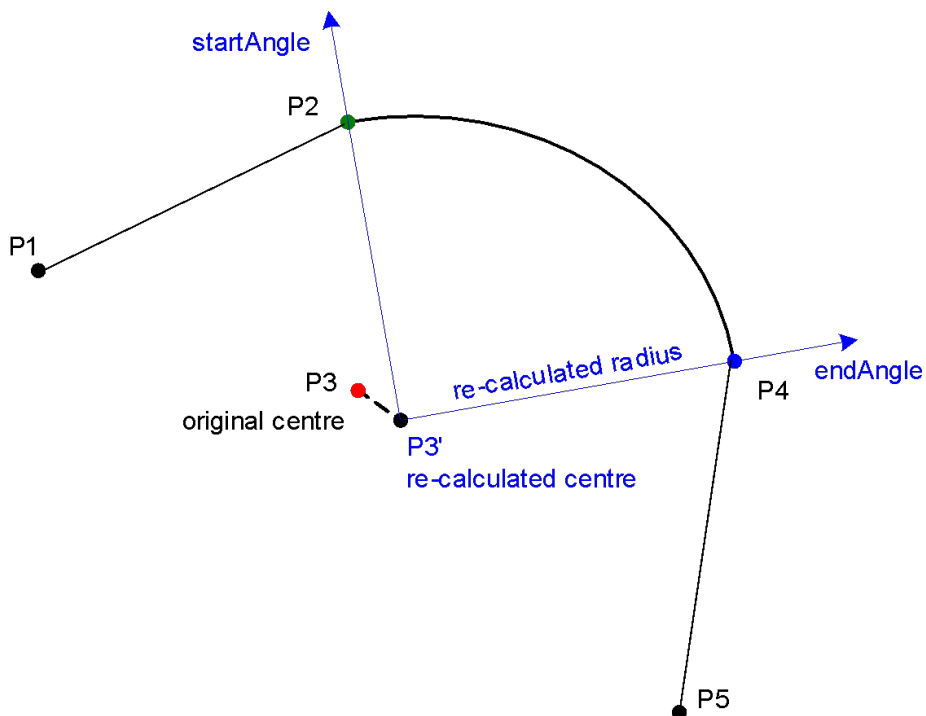


Figure C.3 – Re-calculating the whole arc so that it generates a closed surface

The objective is to find a point that is at equal geodesic distance from the start P2 and end P4 points. This can be done with an algorithm that iteratively tries to find the point of which the distance to both P2 and P4 is the same and which is as close as possible to the original center. However, if the difference between the original distances P3-P2 and P3-P4 is higher than 1%, then the application shall raise an error message and abort the calculation. This rule was also specified in previous AIXM versions (4.5) and it did prevent the provision of geometrically inconsistent arc data.

The calculated point (P3') can then be used as the corrected center point. The corrected radius is the distance from the corrected center P3' to P1 (because of the algorithm applied, this distance is also the distance to P2). The start/end angles are the bearings from the corrected center P3' to P1/P2.

There might be situations where the centre P3 is an Aerodrome Reference Point (ARP) or a Navaid. In this situation, it might be more appropriate to re-calculate the points P2 and P4 in order to correctly close the surface. This could be problematic when P2 or P4 are also ends of arcs. The best solution depends on the intended use of the data; therefore this decision is left to application developers.

Annex D - Perimeter encoding direction considerations

This Annex contains some further consideration about perimeter encoding directions. However, as indicated in the main body of the document this is a theoretical discussion since the current GML profile for aviation data does not allow for internal patches.

The GML Surface implements ISO 19107 GM_Surface whose exterior boundary shall be encoded counter clockwise and any interior boundary encoded clockwise.

The direction of “clockwise” and “counter-clockwise” do not depend on the axis order of the given CRS. They just depend on where “up” is. In EPSG:4326, “up” is implied and pointing from the center of the earth outwards. When a surface is viewed from the side where “up” is, then – following ISO 19107 - the exterior boundary of the surface shall be encoded counter-clockwise while the interior shall be encoded clockwise.

Accordingly, the encoding of surface boundaries – counter-clockwise for exterior boundary, clockwise for any interior boundary - is the same in both left- and right-handed systems; see the following figure.

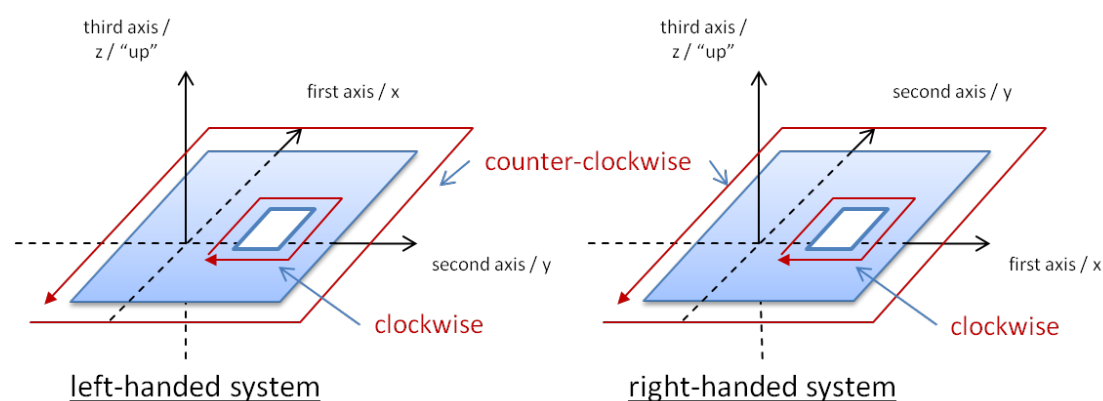


Figure D.1 - The boundary encoding is the same in both left- and right-handed systems

If arcs are part of the internal or external perimeter, the direction of these arcs needs to be in line with the overall orientation of the boundary. For arcs described by centre point and start/end angles, the encoding needs to take into consideration the convention for measuring arcs in left-handed and right-handed geodetic CRS, as detailed in the body of this document. Figure D.2 shows some examples where arcs are part of a surface’s boundary.

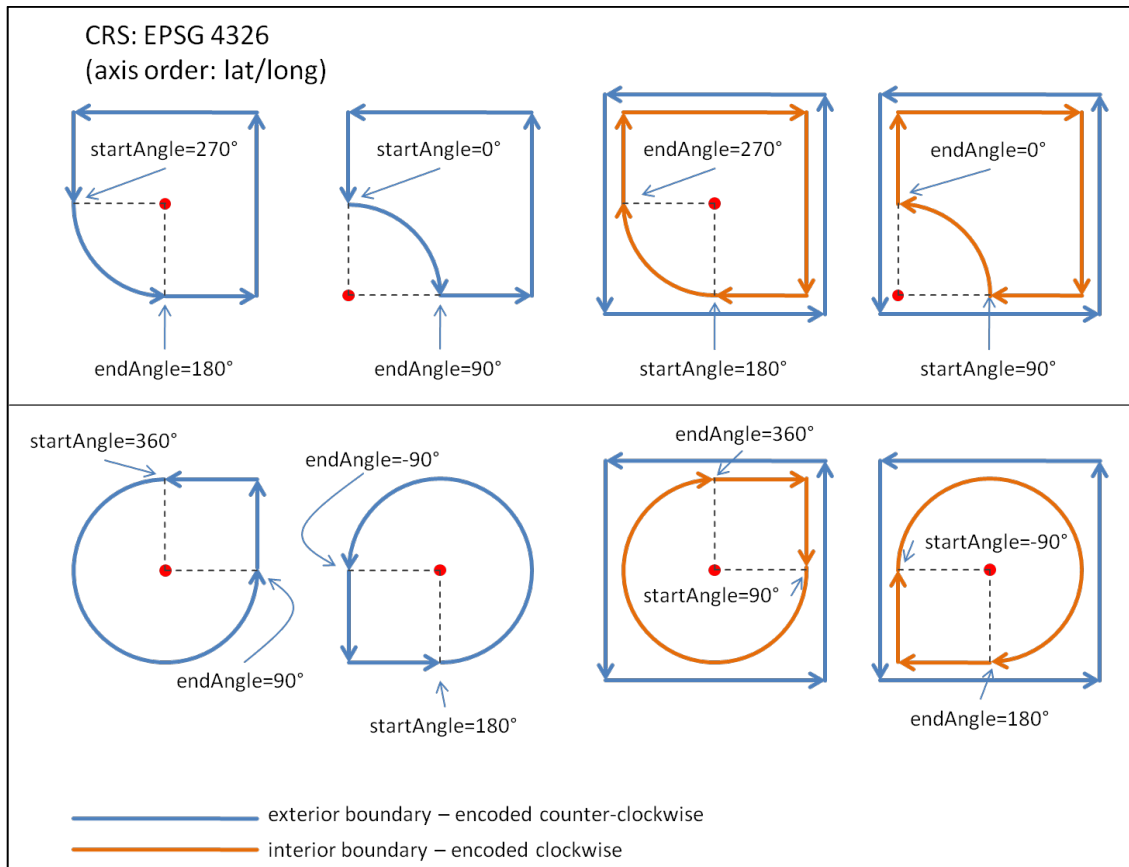


Figure D.2 - arcs and their direction when being part of a surface boundary

Annex E – Offset Curve and Airspace Corridor encoding issues

An offset curve is a curve at a constant distance from the basis curve. It is specified by providing:

- a “baseCurve”, which is a reference to the curve from which this curve is defined as an offset;
- a “distance”, which is the distance at which the offset curve is generated from the basis curve; in a 2D system, positive distances are to be left of the basis curve, and negative distances are right of the basis curve;
- a “refDirection” (optional), which is used to define the vector direction of the offset curve from the basis curve. It shall not be used in AIXM/GML encodings, where all offset curve are used in 2D cases. In that case, distance defines left side (positive distance) or right side (negative distance) with respect to the tangent to the basis curve.

The first thing to be clarified about offset curves is how they are constructed in case the base curve is not a straight line. The correct interpretation is as shown in the diagram A below, because the offset needs to be perpendicular to the tangent of the base curve at every point. Diagram B is not a correct interpretation.

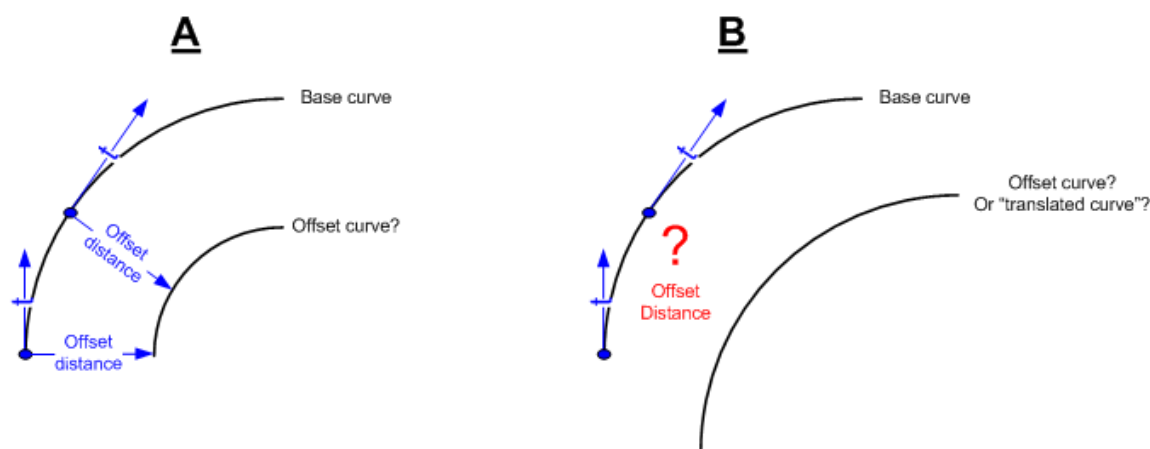


Figure E.1 – Example “A” is the correct interpretation of an offsetCurve

The second issue is constructing an offset curve when the base curve is composed of segments that suddenly change direction. The simplest situation is when the base curve is composed of two consecutive straight lines. The tangent “jumps” at the intersection of two segments, therefore creating either a gap or an intersection between the two offset segments, as shown in Figure E.2.

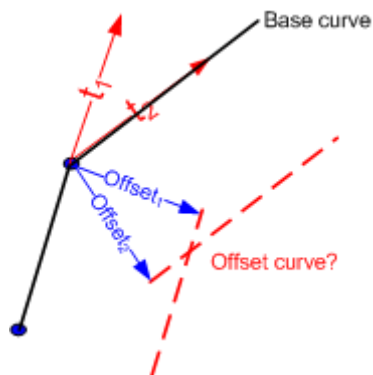


Figure E.2 - What about sudden changes in the tangent direction?

In fact, the encoding of an airspace corridor requires the use of two `gml:OffsetCurve` elements as shown in Figure E.3. In this example, we consider that the curve is closed by two arcs. However, it is possible that straight segments are also used to close the airspace. The official source should clearly specify this aspect. The points $P1'$, $P1''$, $P1'''$ need to be calculated in order to define the closing arc, and similar for the points $P4'$, $P4''$, $P4'''$.

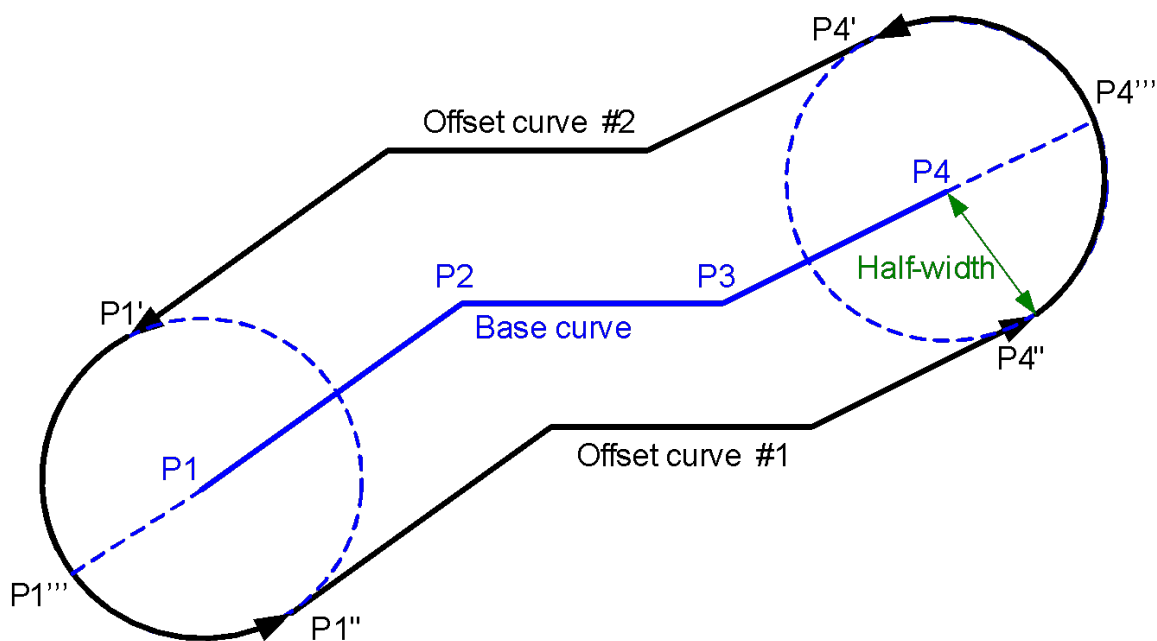


Figure E.3 - Airspace corridor encoding

What we actually need is a “buffer” around the base curve, as shown in Figure E.4. This does not seem to exist in GML as a specific construct.

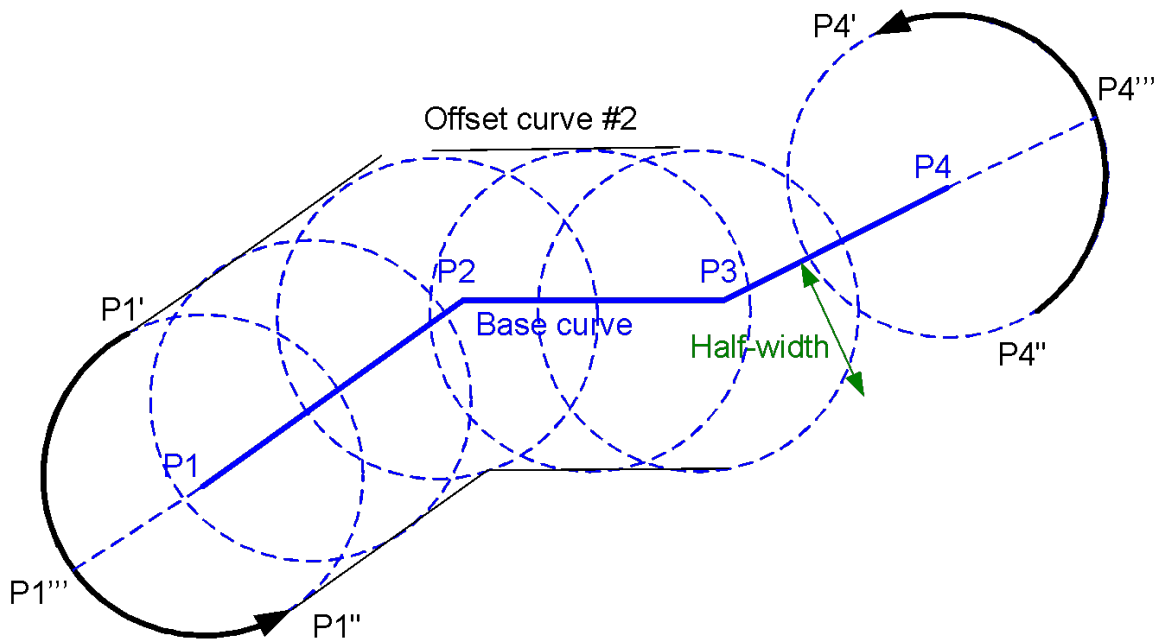


Figure E.4 - Airspace corridor as buffer around a centreline

Annex F – Considerations about interpolations

Role of interpolation

GML (in conformity with ISO:19107) establishes standards to specify the CRS of geographic points, in order to unambiguously identify their position on the Earth surface. Particular attention has to be paid to the definition and Earth location of extended geometric entities (curves and surfaces).

In GML, curve segments are defined by means of a set of control points and control parameters. They also have an enumerator attribute called “interpolation”, specifying the mechanism used to determine the exact geographic extent of the curve, out of the control points and parameters. Its possible values include “linear” and “geodesic”.

The notion of “**linear**” interpolation relies on the existence of a “vector” (or “linear”) structure on the surface. We are used to take for granted that such a kind of structure exists. This is the case for a “flat” surface, isomorphic to $R^2 \times R^2$, because $R^2 \times R^2$ is endowed with a natural linear structure. But this is not true in general, and in particular, it is not true for a curved surface, such as the Earth. So, in principle, **it doesn’t make sense to talk about linear segments on the Earth!** Actually we can bypass this conceptual problem by slightly changing the definition of linear interpolation. This can be done by **relying on the choice of a local CRS**⁸: a linear segment joining two given points, A and B, will be the locus of points whose coordinates belong to the convex linear combination the endpoint coordinates:

$$\begin{cases} x = x_B t + x_A(1 - t) \\ y = y_B t + y_A(1 - t) \\ 0 \leq t \leq 1 \end{cases}$$

The drawback of this definition is that two different CRS on the same surface will give rise to two different “linear segments” (two different sets of points) joining the same couple of points.

Why geodesic is better

Things are a little bit different with the notion of “**geodesic**” interpolation. This is independent from the chosen CRS and just **relies on the existence of a specific metric** structure on the surface. So, contrary to linear ones, geodesic segments joining two given points on the Earth are well defined and independent from the choice of a projection (CRS). The same is true for any other kind of curve. For instance, a circumference (circular arc) of given center and radius, will have a fixed shape on any (projected) chart but will correspond to different curves on the Earth. On the contrary, a “geodesic arc” (the locus of points equidistant, with respect to the ellipsoidal metric, from a given center) will have a different shapes in different charts, corresponding to the same set of points on the Earth.

The geodesic interpolation is a better choice than the linear one, for at least two further reasons:

1. many curves in aeronautical applications (e.g. the footprint of a GPS driven aircraft) are actually geodesic in nature rather than linear

⁸ a linear structure is locally implicitly defined by the coordinate system itself, namely by the local mapping of the surface on

2. worldwide aeronautical data originators adopt many different CRSs (the most convenient one for their own location). When putting together data coming from different originators (hence different CRSs), it is by far computationally easier, for a GIS system, to deal with geodesic entities on a given ellipsoid (such as WGS 84) rather than dealing with linear objects coming from different CRSs

Practical considerations

The purpose of the GML aeronautical profile should be ensuring that the geometric content of AIXM messages is unambiguously interpreted by any aeronautical application. A good mathematical definition is necessary but it is by no means sufficient in order to unambiguously exchange geographic information. If the applications exchanging data are not able to correctly deal with its geometric content, the effort of defining a rigorous language is almost in vain.

One of the most critical issues is the accuracy with which geometric entities have to be stored and exchanged. The magnitude of the errors implicitly committed by erroneously interpreting the interpolation of an extended object may vary broadly, depending on many factors, such as the choice of CRS, the location of the object with respect to the CRS natural domain, the extension of the object itself, etc. In many cases they are (almost) negligible for aeronautical purposes. But it is not rare to run into big trouble which can be brought back to the inaccuracy of the underlying computation engines of the applications dealing with geographic data.

Hardly any GIS application, nowadays, is actually able to correctly deal with geodesic entities. Apart for a few specific tools, GIS usually treats geodesic objects as linear in many respects (for computation and visualization purposes). One of the most commonly used tricks to reduce approximation errors of extended objects is their “densification”. Densification has its drawbacks, anyways:

- the original geometric data (the extended curve) is altered, with **integrity loss**;
- the subsampling introduces a given amount of **arbitrariness and irreversibility** (from the subsampled geometry it is not possible to recover the original data);
- the approximated representation implies some computational errors (consider, for instance the difference between a circular arc and a chord approximating the arc);
- different applications may require different accuracy levels and hence a different amount of “densification”.

For all of these reasons, the GML aeronautical profile should look forward to the future and set down the foundations for a rigorous geographic information exchange among the next generation of aeronautical applications, leaving to each application the responsibility to manage the geometric content in the most proper way for its purposes.

Interpolation and Densification Considerations

Background

Within geospatial standards and technologies there are two classes of geometry models in common use:

1. Comprehensive models based on ISO 19107;
2. Simple models based on a subset of ISO 19107;

To date, there are numerous implementations of simple geometry models, namely:

- OGC Simple Feature Access: OGC’s abstract model for simple geometry.
 - OGC Well Known Text (WKT)
 - OGC Simple Features SQL – Binary Geometry
 - OGC Simple Features - CORBA
- SQL MM (Multi-Media) : This provides an ISO model for database access to geometries which is similar to the OGC Simple Features for SQL.

The majority of mainstream ‘off the shelf’ database technologies which handle geographical data for storage and query are based on the simple feature model. For example:

Platform	Model Support
Oracle Spatial	Oracle SDO Geometry (proprietary interface very similar to SQL MM), SQL MM.
PostGIS	OGC WKT, SQL MM
SQL Server 2008	OGC WKT, SQL MM

In addition, most popular desktop GIS applications also implement a simple geometry model and are interoperable with these databases.

Comparison of models

The ISO19107/GML model contains a number of structures which do not occur in the simple models:

- Composite geometries
- Curve interpolations other than line and 3 point circular arc e.g. arc by centre point, geodesic, offset curve.
- Surface patches other than polygon.
- Composition of geometries by reference (GML allows xpointer references to member geometries within a composite or aggregate geometry).
- Simple geometry only allows circles as polygon boundaries, not as a curve type in their own right.

ISO/GML allows an unlimited level of nesting of geometry structure. For example a composite curve may have members which are composite curves. These curves may in turn contain multiple segments. The simple model only allows a maximum of 3 levels of nested structure e.g. a multcurve may contain compound curves; compound curves may contain primitive curves; compound curves may not contain other compound curves.

For practical management of spatial data it is therefore often necessary to convert geometry types only available within ISO19107/GML 3.2 geometries to ‘simple’ geometries in order to make use of the spatial storage, index and query functionality of mainstream technologies.

Mapping of GML to Simple Geometry

Flattening of geometry structure

Because of the limited amount of nesting allowed in simple geometry deeply nested structures in GML must be “flattened” to fit the simple model. This can be done by removing intermediate layers of structure. For example, a composite curve can be mapped to a compound curve containing primitive elements corresponding to each segment in each curve within the composite.

Densification of curves

The biggest obstacle to converting ISO/GML geometries to simple geometries is the limited number of curve interpolations in simple geometry models. Simple geometry allows for

- Straight lines interpolation between a series of control points.
- Arcs and circles defined by three control points on the edge of the curve.

ISO/GML allows for several additional interpolations. Since these have no direct equivalent in the simple geometry the ISO/GML geometries must be converted to an approximation of the original geometry which is made up of only the interpolation types in the simple model. All curve segment types which cannot be mapped directly to a simple curve should be converted to LineStrings since this is the most interoperable interpolation type.

The process of converting to LineString is one of “densification”. In densification a set of control points are generated along the path of the curve. A LineString is created from the set of control points generated. The linestring is therefore an approximation of the shape of the original curve.

The accuracy of the approximation depends on how many additional control points are generated and their spacing. This can be controlled by parameters. ISO 19107 specified two parameters, either or both of which can be used to control densification. These are:

- Maximum distance between control points.
- Maximum offset i.e. the maximum distance between the densified curve and the original.

The maximum distance between control points allows for more efficient processing, so only this parameter should be used.

The size of the maximum distance between control points required will depend on circumstances. Clearly the level of accuracy required is the principle factor in selecting the parameter, but for practical purposes this must be balanced with the number of control points generated. For example, a trans-Atlantic flight path probably needs less precision than a runway boundary. Equally, setting a maximum control point distance of 10 meters would create an unmanageable number of points in the flight path, but not in the runway boundary.

Table F.1 - ISO segment type mapping in simple geometry

ISO curve segment type	CRS type	Transformatio	Simple representation
------------------------	----------	---------------	-----------------------

		n	
GM_LineString	Any	Direct	LineString
GM_Arc	Geodetic	Densified	LineString
GM_Arc	Non-geodetic	Direct	Arc
GM_ArcString	Geodetic	Densified	LineString**
GM_Circle	Geodetic	Densified	LineString
GM_Circle	Non-Geodetic	Direct	Circle, but only when used as a polygon boundary
GML ArcByCenterPoint ⁹	Any	Densified	LineString *
GML CircleByCenterPoint ¹⁰	Any	Densified	LineString *
GM_Geodesic	Any	Densified	LineString
GM_GeodesicString	Any	Densified	LineString

* Arcs and circles are interpreted as being the set of points at an equal geodesic distance from the centre point on the ellipsoid. In non-geodetic coordinate systems this can result in shapes which are not circular in the projected coordinate system. More information on arc interpretation is provided in Annex A

** Some implementations of simple geometry are strict about arc interpolation and only allow circular arcs in projected planar coordinate systems. This is because, on an ellipsoid, points at an equal distance from a centre point do not form a circular arc.

Since both geometry models use a boundary value representation, the densification of curves can be applied to surface boundaries. No additional transformations are required for surface geometries.

Loss of data structure

The simple model contains less information than the ISO/GML model. ISO/GML geometries which have been mapped to simple geometry therefore cannot be reconstructed from their simple representation.

Where curves have been densified the original (and potentially more precise) representation is not preserved. The simplified geometry is therefore very useful for most practical purposes, but may not be sufficient close to the original for audit and accountability purposes, i.e. it is not exactly the same geometry which was supplied in the GML.

⁹ defined by GML only, not by ISO 19107

¹⁰ defined by GML only, not by ISO 19107

Annex G - GML Profile - XML Schema Implementation

AIXM Point – Documentation

The XML Schema implementation of the *AIXM Point* type is given by the *aixm:Point* XSD element and *aixm:PointType* XSD complex type, documented in the following table.

XSD Element	aixm:Point
Type	aixm:PointType
BaseType	gml:PointType
Restriction	<i>none</i>
Usage	Used to indicate the geographical location – in 2D - of an airport reference point, navaid, waypoint, runway threshold, etc.
Definition	AIXM Point derived from GM_Point containing horizontal accuracy data. In AIXM horizontal accuracy is considered a property of the geometry.
Comments	Annotations of an AIXM Point can be used to provide additional information about the point for human consumption.
Used in	Used in a number of AIXM features. Can also be used as a substitute for GM_Point / gml:Point.
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre><element name="Point" type="aixm:PointType" substitutionGroup="gml:Point"/> <complexType name="PointType"> <complexContent> <extension base="gml:PointType"> <sequence> <group ref="aixm:PointPropertyGroup"/> </sequence> </extension> </complexContent> </complexType> <group name="PointPropertyGroup"> <sequence> <element name="horizontalAccuracy" type="aixm:ValDistanceType" nillable="true" minOccurs="0"/> <element name="annotation" type="aixm:NotePropertyType" nillable="true" minOccurs="0" maxOccurs="unbounded"/> </sequence> </group></pre>
Example	<pre><aixm:Point ... gml:id="P001"> <gml:pos>52.16917 5.21972</gml:pos> <aixm:annotation> <aixm:Note gml:id="N001"> <aixm:translatedNote></pre>

	<pre> <aixm:LinguisticNote gml:id="N002"> <aixm:note lang="ENG">VILLAGE JAKOVO</aixm:note> </aixm:LinguisticNote> </aixm:translatedNote> </aixm:Note> </aixm:annotation> </aixm:Point> </pre>
--	---

AIXM ElevatedPoint – Documentation

The XML Schema implementation of the *AIXM ElevatedPoint* type is given by the *aixm:ElevatedPoint* XSD element and *aixm:ElevatedPointType* XSD complex type, documented in the following table.

XSD Element	aixm:ElevatedPoint
Type	aixm:ElevatedPointType
BaseType	aixm:PointType
Restriction	<i>none</i>
Usage	Used to indicate the geographical location – in 3D - of an airport reference point, navaid, waypoint, runway threshold, etc.
Definition	An AIXM Point derived from (AIXM) Point that includes properties for describing a point with elevation and vertical extent. Used in obstacles, navaids, etc.
Comments	<i>none</i>
Used in	Used in a number of AIXM features. Can also be used as a substitute for AIXM Point.
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre> <element name="ElevatedPoint" type="aixm:ElevatedPointType" substitutionGroup="aixm:Point"/> <complexType name="ElevatedPointType"> <complexContent> <extension base="aixm:PointType"> <sequence> <group ref="aixm:ElevatedPointPropertyGroup"/> <element name="extension" minOccurs="0" maxOccurs="unbounded"> <complexType> <choice> <element ref="aixm:AbstractElevatedPointExtension"/> </choice> <attributeGroup ref="gml:OwnershipAttributeGroup"/> </complexType> </element> </sequence> </pre>

	<pre> </extension> </complexContent> </complexType> <group name="ElevatedPointPropertyGroup"> <sequence> <element name="elevation" type="aixm:ValDistanceVerticalType" nillable="true" minOccurs="0"/> <element name="geoidUndulation" type="aixm:ValDistanceSignedType" nillable="true" minOccurs="0"/> <element name="verticalDatum" type="aixm:CodeVerticalDatumType" nillable="true" minOccurs="0"/> <element name="verticalAccuracy" type="aixm:ValDistanceType" nillable="true" minOccurs="0"/> </sequence> </group> </pre>
Example	<pre> <aixm:ElevatedPoint ... gml:id="P0001" srsName="urn:ogc:def:crs:EPSG::4326"> <gml:pos>52.2889 -32.0350</gml:pos> <aixm:elevation uom="FT">365</aixm:elevation> </aixm:ElevatedPoint> </pre>

AIXM Curve – Documentation

The XML Schema implementation of the *AIXM Curve* type is given by the *aixm:Curve* XSD element and *aixm:CurveType* XSD complex type, documented in the following table.

XSD Element	aixm:Curve
Type	aixm:CurveType
BaseType	gml:CurveType
Restriction	<i>none</i>
Usage	To define shapes (e.g. of a linear obstacle), trajectories (e.g. in a SegmentLeg), centerlines (e.g. of an airspace volume), and extent (e.g. of a route segment).
Definition	An AIXM curve derived from GM_Curve and extended to include Horizontal Accuracy Properties.
Comments	Annotations of an AIXM Curve can be used to provide additional information about the curve for human consumption.
Used in	Used in some AIXM features. Can also be used as a substitute for GM_Curve / gml:Curve.
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre> <element name="Curve" type="aixm:CurveType" substitutionGroup="gml:Curve"/> <complexType name="CurveType"> </pre>

	<pre> <complexContent> <extension base="gml:CurveType"> <sequence> <group ref="aixm:CurvePropertyGroup"/> </sequence> </extension> </complexContent> </complexType> <group name="CurvePropertyGroup"> <sequence> <element name="horizontalAccuracy" type="aixm:ValDistanceType" nillable="true" minOccurs="0"/> <element name="annotation" type="aixm:NotePropertyType" nillable="true" minOccurs="0" maxOccurs="unbounded"/> </sequence> </group> </pre>
<p>Example</p>	<pre> <aixm:Curve ... gml:id="CUR002"> <gml:segments> <gml:GeodesicString interpolation="geodesic"> <gml:posList>39.42916667 47.36333334 39.426818 47.353277 38.87277778 46.54722222</gml:posList> </gml:GeodesicString> </gml:segments> <aixm:annotation> <aixm:Note gml:id="N001"> <aixm:translatedNote> <aixm:LinguisticNote gml:id="N002"> <aixm:note lang="ENG">along the state border with Islamic Republic of Iran</aixm:note> </aixm:LinguisticNote> </aixm:translatedNote> </aixm:Note> </aixm:annotation> </aixm:Curve> </pre>

AIXM ElevatedCurve – Documentation

The XML Schema implementation of the *AIXM ElevatedCurve* type is given by the *aixm:ElevatedCurve* XSD element and *aixm:ElevatedCurveType* XSD complex type, documented in the following table.

XSD Element	aixm:ElevatedCurve
Type	aixm:ElevatedCurveType
BaseType	aixm:CurveType
Restriction	<i>None</i>
Usage	To define centerlines (e.g. of a seaplane ramp site) and extent (e.g. of a guidance line).
Definition	An AIXM elevated curve which extends (AIXM) Curve with

	properties that represent the vertical position (elevation, datum, accuracy).
Comments	<i>None</i>
Used in	Used in some AIXM features. Can also be used as a substitute for AIXM Curve.
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre> <element name="ElevatedCurve" type="aixm:ElevatedCurveType" substitutionGroup="aixm:Curve"/> <complexType name="ElevatedCurveType"> <complexContent> <extension base="aixm:CurveType"> <sequence> <group ref="aixm:ElevatedCurvePropertyGroup"/> <element name="extension" minOccurs="0" maxOccurs="unbounded"> <complexType> <choice> <element ref="aixm:AbstractElevatedCurveExtension"/> </choice> <attributeGroup ref="gml:OwnershipAttributeGroup"/> </complexType> </element> </sequence> </extension> </complexContent> </complexType> <group name="ElevatedCurvePropertyGroup"> <sequence> <element name="elevation" type="aixm:ValDistanceVerticalType" nillable="true" minOccurs="0"/> <element name="geoidUndulation" type="aixm:ValDistanceSignedType" nillable="true" minOccurs="0"/> <element name="verticalDatum" type="aixm:CodeVerticalDatumType" nillable="true" minOccurs="0"/> <element name="verticalAccuracy" type="aixm:ValDistanceType" nillable="true" minOccurs="0"/> </sequence> </group> </pre>
Example	<pre> <aixm:ElevatedCurve ... gml:id="CUR002"> <gml:segments> <gml:GeodesicString interpolation="geodesic"> <gml:posList>39.42916667 47.36333334 39.426818 47.353277 38.87277778 46.54722222</gml:posList> </pre>

	<pre> </gml:GeodesicString> </gml:segments> <aixm:elevation uom="FT">365</aixm:elevation> </aixm:ElevatedCurve> </pre>
--	--

AIXM Surface – Documentation

The XML Schema implementation of the *AIXM Surface* type is given by the *aixm:Surface* XSD element and *aixm:SurfaceType* XSD complex type, documented in the following table.

XSD Element	aixm:Surface
Type	aixm:SurfaceType
BaseType	gml:SurfaceType
Restriction	<i>None</i>
Usage	To define horizontal projection (e.g. of an airspace volume) and extent (e.g. of a circling area), among others.
Definition	An AIXM surface derived from GM_Surface and extended to include Horizontal Accuracy Properties.
Comments	<i>None</i>
Used in	Used in some AIXM features. Can also be used as a substitute for GM_Surface / gml:Surface.
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre> <element name="Surface" type="aixm:SurfaceType" substitutionGroup="gml:Surface"/> <complexType name="SurfaceType"> <complexContent> <extension base="gml:SurfaceType"> <sequence> <group ref="aixm:SurfacePropertyGroup"/> </sequence> </extension> </complexContent> </complexType> <group name="SurfacePropertyGroup"> <sequence> <element name="horizontalAccuracy" type="aixm:ValDistanceType" nillable="true" minOccurs="0"/> <element name="annotation" type="aixm:NotePropertyType" nillable="true" minOccurs="0" maxOccurs="unbounded"/> </sequence> </group> </pre>
Example	<pre> <aixm:Surface ... gml:id="S001" srsName="urn:ogc:def:crs:EPSG::4326"> <gml:patches> <gml:PolygonPatch> <gml:exterior> </pre>

	<pre> <gml:Ring> <gml:curveMember> <aixm:Curve gml:id="C002"> <gml:segments> <gml:GeodesicString> <gml:posList>52.20611 5.2875 52.18917 5.29889 52.19117 5.3289 52.20611 5.2875</gml:posList> </gml:GeodesicString> </gml:segments> </aixm:Curve> </gml:curveMember> </gml:Ring> </gml:exterior> </gml:PolygonPatch> </gml:patches> </aixm:Surface> </pre>
--	--

AIXM ElevatedSurface – Documentation

The XML Schema implementation of the *AIXM ElevatedSurface* type is given by the *aixm:ElevatedSurface* XSD element and *aixm:ElevatedSurfaceType* XSD complex type, documented in the following table.

XSD Element	aixm:ElevatedSurface
Type	aixm:ElevatedSurfaceType
BaseType	aixm:SurfaceType
Restriction	<i>None</i>
Usage	Usually to define the extent (e.g. of a taxiway element) but also to define aviation boundaries (e.g. of an airport heliport) or the area of an airport hot spot.
Definition	An AIXM elevated surface which extends (AIXM) Surface with properties that represent the vertical position (elevation, datum, accuracy).
Comments	<i>None</i>
Used in	Used in some AIXM features. Can also be used as a substitute for AIXM Surface.
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre> <element name="ElevatedSurface" type="aixm:ElevatedSurfaceType" substitutionGroup="aixm:Surface"/> <complexType name="ElevatedSurfaceType"> <complexContent> <extension base="aixm:SurfaceType"> <sequence> <group ref="aixm:ElevatedSurfacePropertyGroup"/> <element name="extension" minOccurs="0" maxOccurs="unbounded"> </pre>

	<pre> <complexType> <choice> <element ref="aixm:AbstractElevatedSurfaceExtension"/> </choice> <attributeGroup ref="gml:OwnershipAttributeGroup"/> </complexType> </element> </sequence> </extension> </complexContent> </complexType> <group name="ElevatedSurfacePropertyGroup"> <sequence> <element name="elevation" type="aixm:ValDistanceVerticalType" nillable="true" minOccurs="0"/> <element name="geoidUndulation" type="aixm:ValDistanceSignedType" nillable="true" minOccurs="0"/> <element name="verticalDatum" type="aixm:CodeVerticalDatumType" nillable="true" minOccurs="0"/> <element name="verticalAccuracy" type="aixm:ValDistanceType" nillable="true" minOccurs="0"/> </sequence> </group> </pre>
Example	<pre> <aixm:ElevatedSurface ... gml:id="S001" srsName="urn:ogc:def:crs:EPSG::4326"> <gml:patches> <gml:PolygonPatch> <gml:exterior> <gml:Ring> <gml:curveMember> <aixm:Curve gml:id="C002"> <gml:segments> <gml:GeodesicString> <gml:posList>52.20611 5.2875 52.18917 5.29889 52.19117 5.3289 52.20611 5.2875</gml:posList> </gml:GeodesicString> </gml:segments> </aixm:Curve> </gml:curveMember> </gml:Ring> </gml:exterior> </gml:PolygonPatch> </gml:patches> <aixm:elevation uom="M">1000</aixm:elevation> </aixm:ElevatedSurface> </pre>

DirectPosition / gml:pos – Documentation

The XML Schema implementation of the *DirectPosition* type (see ISO 19107:2003, section 6.4.1) is given by the *gml:pos* XSD element and *gml:DirectPositionType* XSD complex type (see ISO 19136:2007, section 10.1.4.1 and D.2.3.4), documented in the following table.

XSD Element	gml:pos
Type	gml:DirectPositionType
BaseType	<i>none</i>
Restriction	<i>The srsDimension, axisLabel and uomLabels attributes should not be used in AIXM geometries.</i>
Usage	Defines the coordinates of a position in a given CRS.
Definition	Direct position instances hold the coordinates for a position within some coordinate reference system (CRS). Since direct positions, as data types, will often be included in larger objects (such as geometry elements) that have references to CRS, the srsName attribute will in general be missing, if this particular direct position is included in a larger element with such a reference to a CRS. In this case, the CRS is implicitly assumed to take on the value of the containing object's CRS. If no srsName attribute is given, the CRS shall be specified as part of the larger context this geometry element is part of, typically a geometric object like a point, curve, etc.
Comments	This is one of the key basic types, used to define geometric position values – for example the location of a point.
Used in	Used in the definition of various geometry types, such as GM_Point, GM_Envelope, GM_Position, GM_LineString, GM_Arc, GM_Circle, ArcByCenterPoint and CircleByCenterPoint.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre><complexType name="DirectPositionType"> <simpleContent> <extension base="gml:doubleList"> <attributeGroup ref="gml:SRSReferenceGroup"/> <attribute name="count" type="positiveInteger"/> </extension> </simpleContent> </complexType> <element name="pos" type="gml:DirectPositionType"/></pre>
Example	<gml:pos>46.1 3.2</gml:pos>

GM_Object / gml:AbstractGeometry – Documentation

The XML Schema implementation of the *GM_Object* type (see ISO 19107:2003, section 6.2.2) is given by the *gml:AbstractGeometry* XSD element and *gml:AbstractGeometryType* XSD complex type (see ISO 19136:2007, section 10.1.3), documented in the following table.

XSD Element	gml:AbstractGeometry
Type	gml:AbstractGeometryType
BaseType	gml:AbstractGMLType
Restriction	<i>The srsDimension, axisLabel and uomLabels attributes should not be used in AIXM geometries.</i>
Usage	Represents the parent of all geometry types.
Definition	The gml:AbstractGeometry element is the abstract head of the substitution group for all geometry elements. This includes predefined and user-defined geometry elements. NOTE: the CRS of the geometry can be defined locally (in the geometry itself) but also globally (i.e. external to the geometry).
Comments	All geometry types defined in this profile, including the AIXM geometry types, ultimately derive from gml:AbstractGeometry.
Used in	Used as (direct or indirect) parent type for GM_Point, GM_Curve, GM_Surface, and thus also the derived AIXM geometry types.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre> <element name="AbstractGeometry" type="gml:AbstractGeometryType" abstract="true" substitutionGroup="gml:AbstractGML"/> <complexType name="AbstractGeometryType" abstract="true"> <complexContent> <extension base="gml:AbstractGMLType"> <attributeGroup ref="gml:SRSReferenceGroup"/> </extension> </complexContent> </complexType> <attributeGroup name="SRSReferenceGroup"> <attribute name="srsName" type="anyURI"/> <attribute name="srsDimension" type="positiveInteger"/> <attributeGroup ref="gml:SRSInformationGroup"/> </attributeGroup> <attributeGroup name="SRSInformationGroup"> <attribute name="axisLabels" type="gml:NCNameList"/> <attribute name="uomLabels" type="gml:NCNameList"/> </attributeGroup> </pre>

Example	<i>not applicable, because GM_Object is an abstract type</i>
---------	--

GM_Point / gml:Point – Documentation

The XML Schema implementation of the *GM_Point* type (see ISO 19107:2003, section 6.3.11) is given by the *gml:Point* XSD element and *gml:PointType* XSD complex type (see ISO 19136:2007, section 10.3.1), documented in the following table.

XSD Element	gml:Point
Type	gml:PointType
BaseType	gml:AbstractGeometricPrimitiveType
Restriction	The use of the child element “gml:coordinates” is deprecated. Use “gml:pos” instead.
Usage	Use to identify a geographic point.
Definition	A Point is defined by a single coordinate tuple. The direct position of a point is specified by the pos element which is of type DirectPositionType.
Comments	In AIXM applications the reference to a gml:Point (or subtype, such as aixm:Point) that is given in an XML instance document via an xlink:href may point to an AIXM feature instead of the gml:Point. A GML application may mark this as an error. However, in AIXM applications this can be used to model that the current value of the point property depends on the location of another aeronautical feature (e.g. a Navaid).
Used in	Used as parent type for AIXM Point. Also used in a number of geometry types.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre> <element name="Point" type="gml:PointType" substitutionGroup="gml:AbstractGeometricPrimitive"/> <complexType name="PointType"> <complexContent> <extension base="gml:AbstractGeometricPrimitiveType"> <sequence> <choice> <element ref="gml:pos"/> <element ref="gml:coordinates"/> </choice> </sequence> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:Point srsName="urn:ogc:def:crs:EPSG::4326" ... gml:id="IDX"> <gml:pos>46.1 3.2</gml:pos> </gml:Point> </pre>

GM_Envelope / gml:Envelope – Documentation

The XML Schema implementation of the *GM_Envelope* type (see ISO 19107:2003, section 6.4.3) is given by the *gml:Envelope* XSD element and *gml:EnvelopeType* XSD complex type (see ISO 19136:2007, section 10.1.4.6), documented in the following table.

XSD Element	gml:Envelope
Type	gml:EnvelopeType
BaseType	-
Restriction	The use of the child elements "gml:coordinates" and "gml:pos" has been deprecated. The explicitly named properties "lowerCorner" and "upperCorner" shall be used instead.
Usage	Used in <i>gml:boundedBy</i> property of a feature or feature collection to provide its overall spatial extent as a bounding box. The Envelope may also provide the CRS information for contained geometries.
Definition	Envelope defines an extent using a pair of positions defining opposite corners in arbitrary dimensions. The first direct position is the "lower corner" (a coordinate position consisting of all the minimal ordinates for each dimension for all points within the envelope), the second one the "upper corner" (a coordinate position consisting of all the maximal ordinates for each dimension for all points within the envelope).
Comments	<i>none</i>
Used in	gml:boundedBy property of AIXM feature or feature collection
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre> <element name="Envelope" type="gml:EnvelopeType" substitutionGroup="gml:AbstractObject"/> <complexType name="EnvelopeType"> <choice> <sequence> <element name="lowerCorner" type="gml:DirectPositionType"/> <element name="upperCorner" type="gml:DirectPositionType"/> </sequence> <element ref="gml:pos" minOccurs="2" maxOccurs="2"/> <element ref="gml:coordinates"/> </choice> <attributeGroup ref="gml:SRSReferenceGroup"/> </complexType> </pre>
Example	<pre> <gml:Envelope ... srsName="urn:ogc:def:crs:EPSG::4326"> <gml:lowerCorner>46.1 3.2</gml:lowerCorner> <gml:upperCorner>54.9 15.7</gml:upperCorner> </gml:Envelope> </pre>

GM_PointRef / gml:pointProperty – Documentation

The XML Schema implementation of the *GM_PointRef* type (see ISO 19107:2003, section 6.4.2) is given by the *gml:pointProperty* XSD element and *gml:PointPropertyType* XSD complex type (see ISO 19136:2007, section 10.3.2), documented in the following table.

XSD Element	gml:pointProperty
Type	gml:PointPropertyType
BaseType	<i>none</i>
Restriction	<i>none</i>
Usage	Used to reference an existing point.
Definition	<p>A property that has a point as its value domain may either be an appropriate geometry element encapsulated in an element of this type or an XLink reference to a remote geometry element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element shall be given, but neither both nor none.</p> <p>This property element either references a point via the XLink-attributes or contains the point element. <i>pointProperty</i> is the predefined property which may be used by GML Application Schemas whenever a GML feature has a property with a value that is substitutable for <i>gml:Point</i>.</p>
Comments	The GML implementation of <i>GM_PointRef</i> supports not only the pure reference to a point but also including a point directly.
Used in	Used in the definition of various geometry types.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre><complexType name="PointPropertyType"> <sequence minOccurs="0"> <element ref="gml:Point"/> </sequence> <attributeGroup ref="gml:AssociationAttributeGroup"/> <attributeGroup ref="gml:OwnershipAttributeGroup"/> </complexType> <element name="pointProperty" type="gml:PointPropertyType"/></pre>
Example	<code><gml:pointProperty xlink:href="#P001"/></code>

GM_Position / gml:geometricPositionGroup – Documentation

The XML Schema implementation of the *GM_Position* type (see ISO 19107:2003, section 6.4.5) is given by the *gml:geometricPositionGroup* XSD group (see ISO 19136:2007, section 10.1.4.3 and D.2.3.4), documented in the following table.

XSD group	<code>gml:geometricPositionGroup</code>
Type	<i>NA</i>
BaseType	<i>NA</i>
Restriction	<i>none</i>
Usage	Allows the identification of a position either directly as a coordinate or indirectly as a reference to a <code>GM_Point</code> .
Definition	GML supports two different ways to specify a geometric position: either by a direct position (a data type) or a point (a geometric object). <ul style="list-style-type: none"> • <code>gml:pos</code> elements are positions that are —owned by the geometric primitive encapsulating this geometric position. • <code>gml:pointProperty</code> elements contain a point that may be referenced from other geometry elements or reference another point defined elsewhere (reuse of existing points).
Comments	Relevant in this profile only in the XML Schema Implementation of <code>GM_GeodesicString</code> .
Used in	The XML Schema Implementation of the <code>GM_GeodesicString</code> type.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre><group name="geometricPositionGroup"> <choice> <element ref="gml:pos"/> <element ref="gml:pointProperty"/> </choice> </group></pre>
Example	<pre><gml:GeodesicString> <!-- start of group --> <gml:pos>39.42916667 47.36333334</gml:pos> <gml:pos>39.426818 47.353277</gml:pos> <gml:pointProperty xlink:href="#POXYZ"/> <!-- end of group --> </gml:GeodesicString></pre>

GM_PointArray / `gml:posList` – Documentation

The XML Schema implementation of the `GM_PointArray` type (see ISO 19107:2003, section 6.4.6) is given by the `gml:posList` XSD element and `gml:DirectPositionListType` XSD complex type (see ISO 19136:2007, section 10.1.4.2 and D.2.3.4), documented in the following table.

Note: the `gml:geometricPositionListGroup` XSD group is an alternative XML Schema implementation of `GM_PointArray` defined by GML. However, in GML it is only used in the definition of `PointGrid`, which is irrelevant for this profile.

XSD Element	<code>gml:posList</code>
Type	<code>gml:DirectPositionListType</code>
BaseType	<i>none</i>
Restriction	<i>none</i>

Usage	To encode a sequence of direct positions.
Definition	<p><code>gml:posList</code> instances (and other instances with the content model specified by <code>DirectPositionListType</code>) hold the coordinates for a sequence of direct positions within the same coordinate reference system (CRS). If no <code>srName</code> attribute is given, the CRS shall be specified as part of the larger context this geometry element is part of, typically a geometric object like a point, curve, etc.</p> <p>NOTE: It is expected that the attribute <code>srName</code> will be specified at the direct position level only in rare cases.</p> <p>The optional attribute <code>count</code> specifies the number of direct positions in the list. If the attribute <code>count</code> is present then the attribute <code>srDimension</code> shall be present, too.</p> <p>The number of entries in the list is equal to the product of the dimensionality of the coordinate reference system (i.e. it is a derived value of the coordinate reference system definition) and the number of direct positions.</p>
Comments	<i>none</i>
Used in	Used in the XML Schema Implementation of <code>GM_LineString</code> , <code>GM_Arc</code> , <code>GM_Circle</code> , <code>ArcByCenterPoint</code> , <code>CircleByCenterPoint</code> , <code>GM_GeodesicString</code> and <code>GM_Geodesic</code> .
XML Schema File	(./ISO_19136_Schemas/) <code>geometryBasic.xsd</code>
XML Schema Component	<pre><complexType name="DirectPositionListType"> <simpleContent> <extension base="gml:doubleList"> <attributeGroup ref="gml:SRSReferenceGroup"/> <attribute name="count" type="positiveInteger"/> </extension> </simpleContent> </complexType> <element name="posList" type="gml:DirectPositionListType"/></pre>
Example	<code><gml:posList>39.42916667 47.36333334 39.426818 47.353277 38.87277778 46.54722222</gml:posList></code>

gml:AbstractCurve – Documentation

The XML Schema implementation of the *GML AbstractCurve* type is given by the *gml:AbstractCurve* XSD element and *gml:AbstractCurveType* XSD complex type (see ISO 19136:2007, section 10.4.1), documented in the following table.

XSD Element	gml:AbstractCurve
Type	<code>gml:AbstractCurveType</code>
BaseType	<code>gml:AbstractGeometricPrimitiveType</code>
Restriction	<i>none</i>

Usage	Abstract type defined by GML as supertype for all curve types.
Definition	The AbstractCurve element is the abstract head of the substitution group for all (continuous) curve elements.
Comments	<i>none</i>
Used in	As supertype for gml:Curve, gml:OrientableCurve and gml:CompositeCurve. Also used in the content model of gml:OrientableCurve, gml:CompositeCurve and gml:Ring. Consequently, the GML encoding is not a direct mapping of the ISO 19107 GM_Curve.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic.xsd
XML Schema Component	<pre><element name="AbstractCurve" type="gml:AbstractCurveType" abstract="true" substitutionGroup="gml:AbstractGeometricPrimitive"/> <complexType name="AbstractCurveType" abstract="true"> <annotation> <documentation>gml:AbstractCurveType is an abstraction of a curve to support the different levels of complexity. The curve may always be viewed as a geometric primitive, i.e. is continuous.</documentation> </annotation> <complexContent> <extension base="gml:AbstractGeometricPrimitiveType"/> </complexContent> </complexType></pre>
Example	<i>not applicable because the type is abstract</i>

GM_Curve / gml:Curve – Documentation

The XML Schema implementation of the *GM_Curve* type (see ISO 19107:2003, section 6.3.16) is given by the *gml:Curve* XSD element and *gml:CurveType* XSD complex type (see ISO 19136:2007, section 10.4.5), documented in the following table.

XSD Element	gml:Curve
Type	gml:CurveType
BaseType	gml:AbstractCurveType
Restriction	<i>none</i>
Usage	To represent spatial properties of aeronautical features with 1D shape (e.g. the centerline of an airspace corridor) but also the boundaries of a 2D shape (e.g. the exterior of an airspace).
Definition	<p>A curve is a 1-dimensional primitive. Curves are continuous, connected, and have a measurable length in terms of the coordinate system.</p> <p>A curve is composed of one or more curve segments. Each curve segment within a curve may be defined using a different</p>

	<p>interpolation method. The curve segments are connected to one another, with the end point of each segment except the last being the start point of the next segment in the segment list.</p> <p>The orientation of the curve is positive.</p> <p>The element “segments” encapsulates the segments of the curve.</p>
Comments	The orientation of a curve can be inverted using an <code>OrientableCurve</code> as wrapper.
Used in	Used as parent type for AIXM Curve. Can be used in the definition of a surface boundary.
XML Schema File	(./ISO_19136_Schemas/) <code>geometryPrimitives.xsd</code>
XML Schema Component	<pre> <element name="Curve" type="gml:CurveType" substitutionGroup="gml:AbstractCurve"/> <complexType name="CurveType"> <complexContent> <extension base="gml:AbstractCurveType"> <sequence> <element ref="gml:segments"/> </sequence> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:Curve srsName="urn:ogc:def:crs:EPSG::4326" ... gml:id="IDX"> <gml:segments> <gml:GeodesicString> <gml:posList>lat_P1 long_P1 lat_P2 long_P2</gml:posList> </gml:GeodesicString> <gml:ArcByCenterPoint gml:id="A01"> <gml:pos>lat_P3 long_P3</gml:pos> <gml:radius uom="m">radius</gml:radius> <gml:startAngle uom="deg">calculated_start_angle</gml:startAngle> <gml:endAngle uom="deg">calculated_end_angle</gml:endAngle> </gml:ArcByCenterPoint> <gml:GeodesicString> <gml:posList>lat_P4 long_P4 lat_P5 long_P5</gml:posList> </gml:GeodesicString> </gml:segments> </gml:Curve> </pre>

GM_CurveSegment / gml:AbstractCurveSegment – Documentation

The XML Schema implementation of the *GM_CurveSegment* type (see ISO 19107:2003, section 6.4.9) is given by the *gml:AbstractCurveSegment* XSD element and

gml:AbstractCurveSegmentType XSD complex type (see ISO 19136:2007, section 10.4.7.1), documented in the following table.

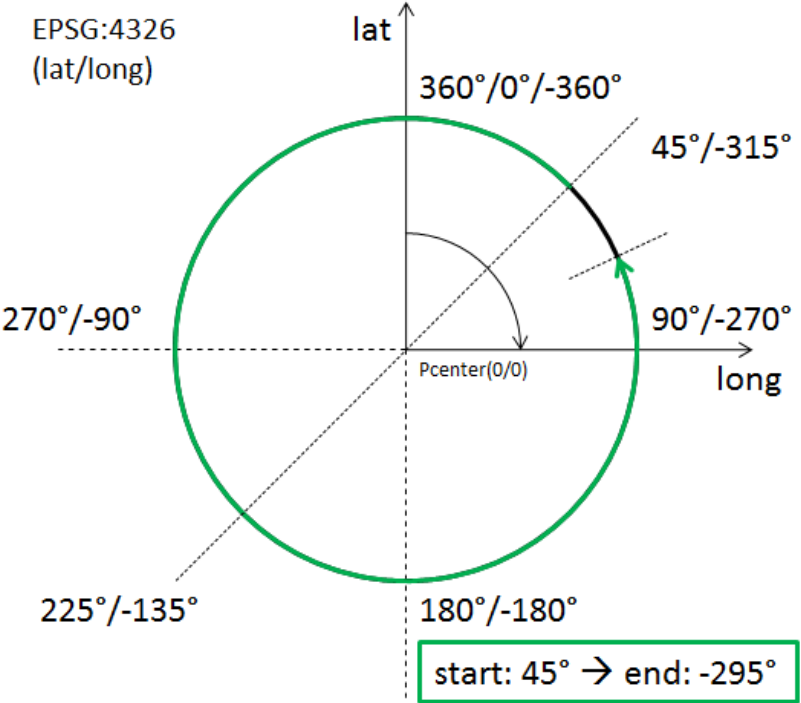
NOTE: *GM_CurveSegment* is the abstract parent of a number of curve segment types.

XSD Element	gml:AbstractCurveSegment
Type	gml:AbstractCurveSegmentType
BaseType	-
Restriction	<i>none</i>
Usage	Abstract type that is the supertype for all curve segment types.
Definition	<p>A curve segment defines a homogeneous segment of a curve. The attributes <i>numDerivativesAtStart</i>, <i>numDerivativesAtEnd</i> and <i>numDerivativesInterior</i> specify the type of continuity as specified in ISO 19107:2003, 6.4.9.3.</p> <p>The <i>AbstractCurveSegment</i> element is the abstract head of the substitution group for all curve segment elements, i.e. continuous segments of the same interpolation mechanism.</p> <p>All curve segments shall have an attribute <i>interpolation</i> with type <i>gml:CurveInterpolationType</i> specifying the curve interpolation mechanism used for this segment. This mechanism uses the control points and control parameters to determine the position of this curve segment.</p>
Comments	<i>none</i>
Used in	Curve
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="AbstractCurveSegment" type="gml:AbstractCurveSegmentType" abstract="true" substitutionGroup="gml:AbstractObject"/> <complexType name="AbstractCurveSegmentType" abstract="true"> <attribute name="numDerivativesAtStart" type="integer" default="0"/> <attribute name="numDerivativesAtEnd" type="integer" default="0"/> <attribute name="numDerivativeInterior" type="integer" default="0"/> </complexType> </pre>
Example	<i>not applicable, because GM_CurveSegment is an abstract type</i>

gml:ArcByCenterPoint – Documentation

The XML Schema implementation of the *GML ArcByCenterPoint* type (see ISO 19136:2007, section D.3.7) is given by the *gml:ArcByCenterPoint* XSD element and *gml:ArcByCenterPointType* XSD complex type (see ISO 19136:2007, section 10.4.7.10), documented in the following table.

XSD Element	gml:ArcByCenterPoint
Type	gml:ArcByCenterPointType
BaseType	gml:AbstractCurveSegmentType
Restriction	The use of the child elements “gml:pointRep” as well as “gml:coordinates” is deprecated. The child element “gml:posList” is allowed by ISO 19136, but if used it shall contain only one position that represents the center of the arc.
Usage	Typical construct for arcs used in the definition of airspace borders in the AI domain.
Definition	This variant of the arc requires that the points on the arc shall be computed instead of storing the coordinates directly. The single control point is the center point of the arc plus the radius and the bearing at start and end. This representation can be used only in 2D. The element radius specifies the radius of the arc. The element startAngle specifies the bearing of the arc at the start. The element endAngle specifies the bearing of the arc at the end. The interpolation is fixed as "circularArcCenterPointWithRadius". Since this type describes always a single arc, the attribute "numArc" is fixed to "1". The content model follows the general pattern for the encoding of curve segments.
Comments	<i>See the main body of the document for further details about the angle measurement convention, angle value ranges, arc direction, arc interpolation, mapping rules as well as recommended units of measurements.</i>
Used in	As child of GM_CurveSegment to represent a segment of a GM_Curve .
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="ArcByCenterPoint" type="gml:ArcByCenterPointType" substitutionGroup="gml:AbstractCurveSegment"/> <complexType name="ArcByCenterPointType"> <complexContent> <extension base="gml:AbstractCurveSegmentType"> <sequence> <choice> <choice> <element ref="gml:pos"/> <element ref="gml:pointProperty"/> element ref="gml:pointRep"/> </choice> <element ref="gml:posList"/> element ref="gml:coordinates"/> </choice> <element name="radius" type="gml:LengthType"/> <element name="startAngle" type="gml:AngleType" minOccurs="0"/> </pre>

	<pre> <element name="endAngle" type="gml:AngleType" minOccurs="0"/> </sequence> <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="circularArcCenterPointWithRadius"/> <attribute name="numArc" type="integer" use="required" fixed="1"/> </extension> </complexContent> </complexType> </pre>
Example	<p>Assuming an arc defined by its center point as well as start and end angle as shown in the following diagram:</p>  <p>Then the GML encoding in an <code>ArcByCenterPoint</code> is as follows (in a left-handed CRS):</p> <pre> <gml:ArcByCenterPoint ... numArc="1"> <gml:pos>0 0</gml:pos> <gml:radius uom="m">1000</gml:radius> <gml:startAngle uom="deg">45.0</gml:startAngle> <gml:endAngle uom="deg">-295.0</gml:endAngle> </gml:ArcByCenterPoint> </pre>

gml:CircleByCenterPoint – Documentation

The XML Schema implementation of the *GML CircleByCenterPoint* type (see ISO 19136:2007, section D.3.7) is given by the *gml:CircleByCenterPoint* XSD element and *gml:CircleByCenterPointType* XSD complex type (see ISO 19136:2007, section 10.4.7.11), documented in the following table.

XSD Element	gml:CircleByCenterPoint
Type	gml:CircleByCenterPointType
BaseType	gml:ArcByCenterPointType
Restriction	Only the circle center position as well as the radius can be defined. The use of the child elements “gml:pointRep” as well as “gml:coordinates” is deprecated. The child element “gml:posList” is allowed by ISO 19136, but if used it shall contain only one position that represents the center of the circle.
Usage	To define the geometry of a circular airspace in the AI domain.
Definition	A gml:CircleByCenterPoint is a gml:ArcByCenterPoint with identical start and end angle to form a full circle. Again, this representation can be used only in 2D.
Comments	Use of CircleByCenterPoint is recommended only in the definition of an interior and/or exterior surface boundary – like an airspace boundary – with circular shape. In that case, the direction of the boundary is implied and automatically complies with ISO 19107 rules (exterior boundary is encoded counter clockwise while any interior boundary is encoded clockwise; for further information. If gml:CircleByCenterPoint was used in conjunction with other curve segments to define a curve then the direction (clockwise / counter-clockwise) of the CircleByCenterPoint is not well-defined by GML.
Used in	As child of ArcByCenterPoint to represent the segment of a GM_Curve that forms a circle.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="CircleByCenterPoint" type="gml:CircleByCenterPointType" substitutionGroup="gml:ArcByCenterPoint"/> <complexType name="CircleByCenterPointType"> <complexContent> <restriction base="gml:ArcByCenterPointType"> <sequence> <choice> <choice> <element ref="gml:pos"/> <element ref="gml:pointProperty"/> element ref="gml:pointRep"/> </choice> <element ref="gml:posList"/> element ref="gml:coordinates"/> </choice> <element name="radius" type="gml:LengthType"/> </sequence> </restriction> </complexContent> </complexType> </pre>
Example	<gml:CircleByCenterPoint ... numArc="1">

	<pre><gml:pos>0 0</gml:pos> <gml:radius uom="m">1000</gml:radius> </gml:CircleByCenterPoint></pre>
--	--

GM_Arc / gml:Arc – Documentation

The XML Schema implementation of the *GM_Arc* type (see ISO 19107:2003, section 6.4.15) is given by the *gml:Arc* XSD element and *gml:ArcType* XSD complex type (see ISO 19136:2007, section 10.4.7.6), documented in the following table.

XSD Element	gml:Arc
Type	gml:ArcType
BaseType	gml:ArcStringType (not documented in this profile because use of ArcString is not foreseen by this profile)
Restriction	The value domain of attribute "numArc" is fixed to "1". Use of the child elements "gml:pointRep" and "gml:coordinates" is deprecated.
Usage	To define an arc that is given via three control points – usually to define a curve segment.
Definition	An Arc is an arc string with only one arc unit, i.e. three control points including the start and end point. As arc is an arc string consisting of a single arc, the attribute "numArc" is fixed to "1".
Comments	The direction of the Arc is implicitly provided by the order of its control points. Use of GM_Arc is also known as "arc by edge". Although not extensively used at present in the aviation domain, it is expected that its usage will increase, with the corresponding diminishing of ArcByCenterPoint. GM_Arc is simpler and less open to interpretation than ArcByCenterPoint.
Used in	As child of GM_ArcString, usually to represent a segment of a GM_Curve.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre><element name="Arc" type="gml:ArcType" substitutionGroup="gml:ArcString"/> <complexType name="ArcType"> <complexContent> <restriction base="gml:ArcStringType"> <sequence> <choice> <choice minOccurs="3" maxOccurs="3"> <element ref="gml:pos"/> <element ref="gml:pointProperty"/> element ref="gml:pointRep"/> </choice> <element ref="gml:posList"/> element ref="gml:coordinates"/> </sequence> </restriction> </complexContent> </complexType></pre>

	<pre> </choice> </sequence> <attribute name="numArc" type="integer" fixed="1"/> </restriction> </complexContent> </complexType> </pre>
Example	<pre> <gml:Arc ...> <gml:posList>0 0 1 1 0 2</gml:posList> </gml:Arc> </pre>

GM_Circle / gml:Circle – Documentation

The XML Schema implementation of the *GM_Circle* type (see ISO 19107:2003, section 6.4.16) is given by the *gml:Circle* XSD element and *gml:CircleType* XSD complex type (see ISO 19136:2007, section 10.4.7.7), documented in the following table.

XSD Element	gml:Circle
Type	gml:CircleType
BaseType	gml:ArcType
Restriction	Use of the child elements “gml:pointRep” and “gml:coordinates” is deprecated.
Usage	To define a circle that is given via three control points – usually to define the boundary of a circular airspace.
Definition	A Circle is an arc whose ends coincide to form a simple closed loop. The three control points shall be distinct non-co-linear points for the circle to be unambiguously defined. The arc is simply extended past the third control point until the first control point is encountered.
Comments	Other than CircleByCenterPoint, Circle has a well defined direction.
Used in	As child of GM_Arc, usually to represent a segment of a GM_Curve that forms a circle.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="Circle" type="gml:CircleType" substitutionGroup="gml:Arc"/> <complexType name="CircleType"> <complexContent> <extension base="gml:ArcType"/> </complexContent> </complexType> </pre>
Example	<pre> <gml:Circle ...> <gml:posList>0 1 -1 0 0 -1</gml:posList> </gml:Circle> </pre>

GM_GeodesicString / gml:GeodesicString – Documentation

The XML Schema implementation of the *GM_GeodesicString* type (see ISO 19107:2003, section 6.4.12) is given by the *gml:GeodesicString* XSD element and *gml:GeodesicStringType* XSD complex type (see ISO 19136:2007, section 10.4.7.20), documented in the following table.

XSD Element	gml:GeodesicString
Type	gml:GeodesicStringType
BaseType	gml:AbstractCurveSegmentType
Restriction	<i>none</i>
Usage	Used for the encoding of straight lines on the earth surface.
Definition	A sequence of geodesic segments. The number of control points shall be at least two. interpolation is fixed as "geodesic". The content model follows the general pattern for the encoding of curve segments.
Comments	GeodesicString is the default encoding for straight lines recommended by this document. The more compact form of representing the control points of a GeodesicString is achieved via the gml:posList element.
Used in	As child of GM_CurveSegment to represent a segment of a GM_Curve.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="GeodesicString" type="gml:GeodesicStringType" substitutionGroup="gml:AbstractCurveSegment"/> <complexType name="GeodesicStringType"> <complexContent> <extension base="gml:AbstractCurveSegmentType"> <choice> <element ref="gml:posList"/> <group ref="gml:geometricPositionGroup" minOccurs="2" maxOccurs="unbounded"/> </choice> <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="geodesic"/> </complexContent> </complexType> </pre>
Example	<pre> <gml:GeodesicString ...> <gml:posList>39.42916667 47.36333334 39.426818 47.353277 38.87277778 46.54722222</gml:posList> </gml:GeodesicString> </pre>

GM_Geodesic / gml:Geodesic – Documentation

The XML Schema implementation of the *GM_Geodesic* type (see ISO 19107:2003, section 6.4.13) is given by the *gml:Geodesic* XSD element and *gml:GeodesicType* XSD complex type (see ISO 19136:2007, section 10.4.7.21), documented in the following table.

XSD Element	gml:Geodesic
Type	gml:GeodesicType
BaseType	gml:GeodesicStringType
Restriction	<i>none</i>
Usage	To define a straight line on the earth surface – usually to define a curve segment.
Definition	A GM_Geodesic consists of two distinct positions joined by a geodesic curve. The control points of a GM_Geodesic shall all lie on the geodesic between its start point and end point. Between these two points, a geodesic curve defined from the ellipsoid or geoid model used by the coordinate reference system may be used to interpolate other positions. Any other point in the controlPoint array must fall on this geodesic.
Comments	Geodesic is a particular case of the more general concept of “GeodesicString”.
Used in	As child of GM_GeodesicString, usually to represent a segment of a GM_Curve.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="Geodesic" type="gml:GeodesicType" substitutionGroup="gml:GeodesicString"/> <complexType name="GeodesicType"> <complexContent> <extension base="gml:GeodesicStringType"/> </complexContent> </complexType> <element name="Geodesic" type="gml:GeodesicType" substitutionGroup="gml:GeodesicString"/> <complexType name="SurfaceType"> <complexContent> <extension base="gml:AbstractSurfaceType"> <sequence> <element ref="gml:patches"/> </sequence> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:Geodesic ...> <gml:posList>39.42916667 47.36333334 38.87277778 46.54722222</gml:posList> </gml:Geodesic> </pre>

GM_LineString / gml:LineStringSegment – Documentation

The XML Schema implementation of the *GM_LineString* type (see ISO 19107:2003, section 6.4.10) is given by the *gml:LineStringSegment* XSD element and *gml:LineStringSegmentType* XSD complex type (see ISO 19136:2007, section 10.4.7.4), documented in the following table.

XSD Element	gml:LineStringSegment
Type	gml:LineStringSegmentType
BaseType	gml:AbstractCurveSegmentType
Restriction	Use of the child elements “gml:pointRep” and “gml:coordinates” is deprecated.
Usage	Use to represent a curve segment that is composed of line segments with linear interpolation.
Definition	A LineStringSegment is a curve segment that is defined by two or more control points including the start and end point, with linear interpolation between them. The content model follows the general pattern for the encoding of curve segments.
Comments	In order to represent a parallel (line between locations with same latitude), a GM_LineString with CRS EPSG:4326 should be used for encoding. In order to represent a rhumbline whereas the two consecutive latitudes are different, a GM_LineString with a “Mercator” projected CRS like EPSG:3395 should be used. The more compact form of representing the control points of a GM_LineString is achieved via the gml:posList element.
Used in	As child of GM_CurveSegment to represent a segment of a GM_Curve.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="LineStringSegment" type="gml:LineStringSegmentType" substitutionGroup="gml:AbstractCurveSegment"/> <complexType name="LineStringSegmentType"> <complexContent> <extension base="gml:AbstractCurveSegmentType"> <sequence> <choice> <choice minOccurs="2" maxOccurs="unbounded"> <element ref="gml:pos"/> <element ref="gml:pointProperty"/> element ref="gml:pointRep"/> </choice> <element ref="gml:posList"/> element ref="gml:coordinates"/> </choice> </sequence> <attribute name="interpolation" type="gml:CurveInterpolationType" fixed="linear"/> </extension> </complexContent> </pre>

	</complexType>
Example	<pre><gml:LineStringSegment ...> <gml:posList> 40.05 45.88972222 40.06 46.93333333</gml:posList> </gml:LineStringSegment></pre>

GM_Surface / gml:Surface – Documentation

The XML Schema implementation of the *GM_Surface* type (see ISO 19107:2003, section 6.3.17) is given by the *gml:Surface* XSD element and *gml:SurfaceType* XSD complex type (see ISO 19136:2007, section 10.5.10), documented in the following table.

XSD Element	gml:Surface
Type	gml:SurfaceType
BaseType	gml:AbstractSurfaceType
Restriction	<i>none</i>
Usage	Defines the basic structure of a surface, which is a composition of one or more surface patches.
Definition	A Surface is a 2-dimensional primitive and is composed of one or more surface patches as specified in ISO 19107:2003, 6.3.17.1. The surface patches are connected to one another.
Comments	The direction of interior and exterior boundaries has to comply with ISO 19107 rules. More specifically, the exterior must be encoded counter clockwise while any interior boundary must be encoded clockwise; for further information.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre><element name="Surface" type="gml:SurfaceType" substitutionGroup="gml:AbstractSurface"/> <complexType name="SurfaceType"> <complexContent> <extension base="gml:AbstractSurfaceType"> <sequence> <element ref="gml:patches"/> </sequence> </extension> </complexContent> </complexType></pre>
Used in	Used as supertype for AIXM Surface.
Example	<pre><gml:Surface ... gml:id="IDX" srsName="urn:ogc:def:crs:EPSG::4326"> <gml:patches> <gml:PolygonPatch> <gml:exterior> <gml:Ring> <gml:curveMember> <gml:Curve gml:id="C002"> <gml:segments> <gml:GeodesicString></pre>

	<pre> <gml:posList>52.20611 5.2875 52.18917 5.29889 52.19117 5.3289 52.20611 5.2875</gml:posList> </gml:GeodesicString> </gml:segments> </gml:Curve> </gml:curveMember> </gml:Ring> </gml:exterior> </gml:PolygonPatch> </gml:patches> </gml:Surface> </pre>
--	--

GM_SurfacePatch / gml:AbstractSurfacePatch – Documentation

The XML Schema implementation of the *GM_SurfacePatch* type (see ISO 19107:2003, section 6.4.34) is given by the *gml:AbstractSurfacePatch* XSD element and *gml:AbstractSurfacePatchType* XSD complex type (see ISO 19136:2007, section 10.5.12.1), documented in the following table.

XSD Element	gml:AbstractSurfacePatch
Type	gml:AbstractSurfacePatchType
BaseType	<i>none</i>
Restriction	<i>none</i>
Usage	Represents the parent of all surface patch types – only the GM_Polygon child type is currently used in the encoding of AIXM surface patches.
Definition	<p>A surface patch defines a homogenous portion of a surface.</p> <p>The AbstractSurfacePatch element is the abstract head of the substitution group for all surface patch elements describing a continuous portion of a surface.</p> <p>All surface patches shall have an attribute interpolation (declared in the types derived from gml:AbstractSurfacePatchType) specifying the interpolation mechanism used for the patch using gml:SurfaceInterpolationType.</p>
Comments	<i>none</i>
Used in	GM_Surface
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="AbstractSurfacePatch" type="gml:AbstractSurfacePatchType" abstract="true"/> <complexType name="AbstractSurfacePatchType" abstract="true"/> </pre>
Example	<i>not applicable because the type is abstract</i>

GM_Polygon / gml:PolygonPatch – Documentation

The XML Schema implementation of the *GM_Polygon* type (see ISO 19107:2003, section 6.4.36) is given by the *gml:PolygonPatch* XSD element and *gml:PolygonPatchType* XSD complex type (see ISO 19136:2007, section 10.5.12.4), documented in the following table.

XSD Element	gml:PolygonPatch
Type	gml:PolygonPatchType
BaseType	gml:AbstractSurfacePatchType
Restriction	<i>Interior patches are not allowed for aeronautical data</i>
Usage	Used to represent the patch(es) of an AIXM Surface.
Definition	A gml:PolygonPatch is a surface patch that is defined by a set of boundary curves and an underlying surface to which these curves adhere. The curves shall be coplanar and the polygon uses planar interpolation in its interior. interpolation is fixed to "planar", i.e. an interpolation shall return points on a single plane. The boundary of the patch shall be contained within that plane.
Comments	The direction of interior and exterior boundaries has to comply with ISO 19107 rules. More specifically, the exterior must be encoded counter clockwise while any interior boundary must be encoded clockwise; for further information. GM_Ring should be used to represent and encode the exterior/interior of a GM_Polygon.
Used in	As child of GM_SurfacePatch to represent the surface patch(es) of a GM_Surface.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="PolygonPatch" type="gml:PolygonPatchType" substitutionGroup="gml:AbstractSurfacePatch"/> <complexType name="PolygonPatchType"> <complexContent> <extension base="gml:AbstractSurfacePatchType"> <sequence> <element ref="gml:exterior" minOccurs="0"/> <element ref="gml:interior" minOccurs="0" maxOccurs="unbounded"/> </sequence> <attribute name="interpolation" type="gml:SurfaceInterpolationType" fixed="planar"/> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:PolygonPatch ...> <gml:exterior> <gml:Ring> <gml:curveMember> <gml:Curve gml:id="C002"> <gml:segments> </pre>

	<pre> <gml:GeodesicString> <gml:posList>52.20611 5.2875 52.18917 5.29889 52.19117 5.3289 52.20611 5.2875</gml:posList> </gml:GeodesicString> </gml:segments> </gml:Curve> </gml:curveMember> </gml:Ring> </gml:exterior> </gml:PolygonPatch> </pre>
--	---

gml:AbstractRing – Documentation

The XML Schema implementation of the *GML AbstractRing* type is given by the *gml:AbstractRing* XSD element and *gml:AbstractRingType* XSD complex type (see ISO 19136:2007, section 10.5.6), documented in the following table.

XSD Element	gml:AbstractRing
Type	gml:AbstractRingType
BaseType	<i>none</i>
Restriction	<i>none</i>
Usage	Represents the supertype for the GML encoding of the GM_Ring type and its subtypes.
Definition	An abstraction of a ring to support surface boundaries of different complexity. The AbstractRing element is the abstract head of the substitution group for all closed boundaries of a surface patch.
Comments	At the moment, the only substitute for gml:AbstractRing foreseen by this profile is GM_Ring / gml:Ring.
Used in	The definition of the exterior/interior of a surface boundary
XML Schema File	(./ISO_19136_Schemas/) geometryBasic2d.xsd
XML Schema Component	<pre> <element name="AbstractRing" type="gml:AbstractRingType" abstract="true" substitutionGroup="gml:AbstractObject"/> <complexType name="AbstractRingType" abstract="true"> <sequence/> </complexType> </pre>
Example	<i>not applicable because the type is abstract</i>

GM_Ring / gml:Ring – Documentation

The XML Schema implementation of the *GM_Ring* type (see ISO 19107:2003, section 6.3.6) is given by the *gml:Ring* XSD element and *gml:RingType* XSD complex type (see ISO 19136:2007, section 10.5.11.1), documented in the following table.

XSD Element	gml:Ring
--------------------	-----------------

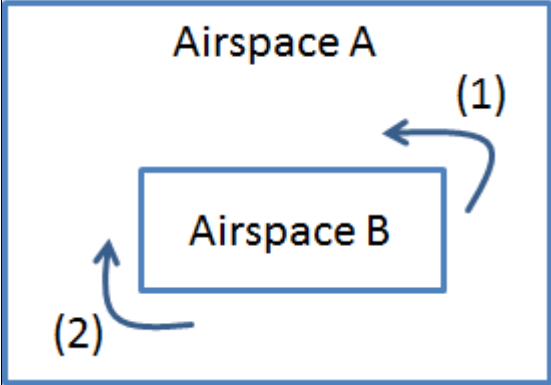
Type	<code>gml:RingType</code>
BaseType	<code>gml:AbstractRingType</code>
Restriction	<i>none</i>
Usage	Used to represent the exterior/interior of a surface boundary.
Definition	<p>A ring is used to represent a single connected component of a surface boundary as specified in ISO 19107:2003, 6.3.6.</p> <p>Every <code>gml:curveMember</code> references or contains one curve, i.e. any element which is substitutable for <code>gml:AbstractCurve</code>. In the context of a ring, the curves describe the boundary of the surface. The sequence of curves shall be contiguous and connected in a cycle. If provided, the <code>aggregationType</code> attribute shall have the value "sequence".</p>
Comments	<p>In the special case that there is only one curve member in a Ring, the curve member itself needs to form a cycle. For example, if a Ring is formed by a <code>GeodesicString</code>, then the first and last position element of that <code>GeodesicString</code> must be equal in order to form a cycle. The <code>CircleByCenterPoint</code> type automatically forms a cycle.</p> <p>In AIXM applications airspace boundaries may be based on national borders or on other geographical features, such as shorelines, rivers etc. The encoding of such <code>GeoBorders</code> can be achieved using annotations (for applications where a text remark is sufficient) or using references (for applications where a true reference needs to be preserved). The former approach depends on the use of <code>aixm:Curve</code> as curve member, the latter requires either a local reference to a curve or an abstract reference to a remote feature.</p>
Used in	The definition of the exterior/interior(s) of a Polygon.
XML Schema File	(./ISO_19136_Schemas/) <code>geometryPrimitives.xsd</code>
XML Schema Component	<pre> <element name="Ring" type="gml:RingType" substitutionGroup="gml:AbstractRing"/> <complexType name="RingType"> <complexContent> <extension base="gml:AbstractRingType"> <sequence> <element ref="gml:curveMember" maxOccurs="unbounded"/> </sequence> <attributeGroup ref="gml:AggregationAttributeGroup"/> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:Ring ...> <gml:curveMember> <gml:Curve gml:id="CUR001"> <gml:segments> <gml:LineStringSegment interpolation="linear"> <gml:posList> 40.05 45.88972222 40.05 </pre>

	<pre> 46.93333333</gml:posList> </gml:LineStringSegment> <gml:GeodesicString interpolation="geodesic"> <gml:posList>40.05 46.93333333 39.42916667 47.36333334</gml:posList> </gml:GeodesicString> </gml:segments> </gml:Curve> </gml:curveMember> <gml:curveMember xlink:href="#CRV002" xlink:title="along the state border with Islamic Republic of Iran"/> <gml:curveMember xlink:href="#CRV003" xlink:title="along the state border with Armenia"/> </gml:Ring> </pre>
--	---

GM_OrientableCurve / gml:OrientableCurve – Documentation

The XML Schema implementation of the *GM_OrientableCurve* type (see ISO 19107:2003, section 6.3.14) is given by the *gml:OrientableCurve* XSD element and *gml:OrientableCurveType* XSD complex type (see ISO 19136:2007, section 10.4.6), documented in the following table.

XSD Element	gml:OrientableCurve
Type	gml:OrientableCurveType
BaseType	gml:AbstractCurveType
Restriction	<i>none</i>
Usage	To invert the orientation of another curve.
Definition	<p>OrientableCurve consists of a curve and an orientation. If the orientation is "+", then the OrientableCurve is identical to the baseCurve. If the orientation is "-", then the OrientableCurve is related to another AbstractCurve with a parameterization that reverses the sense of the curve traversal.</p> <p>The property gml:baseCurve references or contains the base curve, i.e. it either references the base curve via the XLink-attributes or contains the curve element. A curve element is any element which is substitutable for gml:AbstractCurve. The base curve has positive orientation.</p> <p>NOTE This definition allows for a nested structure, i.e. an gml:OrientableCurve may use another gml:OrientableCurve as its base curve.</p>
Comments	<p>OrientableCurve can be used to invert the direction of a curve inside a gml:Ring. OrientableCurve is not foreseen to be used often. However, there is one use case that would require OrientableCurve: If an already established curve – for example from the border of an existing airspace – is used by reference in the definition of another airspace then there may be a need to invert the direction of that</p>

	<p>referenced curve. For example, one airspace (A) has a hole which is another airspace (B). In that particular case, the boundary of B is encoded counter-clockwise. However, to be included as an interior boundary in A that direction would need to be inverted (to ensure that the interior of A is encoded clockwise) which is not possible without an OrientableCurve wrapper if the curve members of the boundary of B are to be referenced. The following diagram illustrates the setup:</p>  <p>(1) Exterior of Airspace B, encoded counter-clockwise (2) Interior of Airspace A, encoded clockwise.</p> <p>Even though in AIXM the AirspaceVolume element is used to aggregate an airspace made of parts, a client may like to see the horizontal projection of the airspace aggregation described in GML, for which the OrientableCurve may then be used.</p>
Used in	As member of a Ring, CompositeCurve or another OrientableCurve.
XML Schema File	(/ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="OrientableCurve" type="gml:OrientableCurveType" substitutionGroup="gml:AbstractCurve"/> <complexType name="OrientableCurveType"> <complexContent> <extension base="gml:AbstractCurveType"> <sequence> <element ref="gml:baseCurve"/> </sequence> <attribute name="orientation" type="gml:SignType" default="+"/> </extension> </complexContent> </complexType> <element name="baseCurve" type="gml:CurvePropertyType"/> </pre>
Example	<pre> <gml:Ring ...> <gml:curveMember> <gml:OrientableCurve gml:id="IDX" orientation="-"> </pre>

	<pre> <gml:baseCurve xlink:href="#CUR001"/> </gml:OrientableCurve> </gml:curveMember> </gml:Ring> </pre>
--	--

GM_CompositeCurve / gml:CompositeCurve – Documentation

The XML Schema implementation of the *GM_CompositeCurve* type (see ISO 19107:2003, section 6.6.5) is given by the *gml:CompositeCurve* XSD element and *gml:CompositeCurveType* XSD complex type (see ISO 19136:2007, section 11.2.2.2), documented in the following table.

XSD Element	gml:CompositeCurve
Type	gml:CompositeCurveType
Base Type	gml:AbstractCurveType
Restriction	<i>none</i>
Usage	To represent a curve as a combination of other curves.
Definition	<p>A gml:CompositeCurve is represented by a sequence of (orientable) curves such that each curve in the sequence terminates at the start point of the subsequent curve in the list.</p> <p>gml:curveMember references or contains inline one curve in the composite curve.</p> <p>The curves are contiguous, the collection of curves is ordered. Therefore, if provided, the aggregationType attribute shall have the value "sequence".</p> <p>NOTE This definition allows for a nested structure, i.e. a gml:CompositeCurve may use, for example, another gml:CompositeCurve as a curve member.</p>
Comments	CompositeCurve supports a simple aggregation of curves. This can be used to combine existing curves by reference (xlink:href), for example to ensure consistency in use of common boundaries (or segments thereof).
Used in	As member of a Ring, OrientableCurve or CompositeCurve.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<pre> <element name="CompositeCurve" type="gml:CompositeCurveType" substitutionGroup="gml:AbstractCurve"/> <complexType name="CompositeCurveType"> <complexContent> <extension base="gml:AbstractCurveType"> <sequence> <element ref="gml:curveMember" maxOccurs="unbounded"/> </sequence> <attributeGroup ref="gml:AggregationAttributeGroup"/> </extension> </complexContent> </complexType> </pre>

	<pre> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:CompositeCurve ... gml:id="IDX"> <gml:curveMember xlink:href="#CRV001"/> <gml:curveMember xlink:href="#CRV002"/> <gml:curveMember xlink:href="#CRV003"/> </gml:CompositeCurve> </pre>

TM_GeometricPrimitive / gml:AbstractTimeGeometricPrimitive – Documentation

The XML Schema implementation of the *TM_GeometricPrimitive* type (see ISO 19108:2002, section 5.2.3) is given by the *gml:AbstractTimeGeometricPrimitive* XSD element and *gml:AbstractTimeGeometricPrimitiveType* XSD complex type, documented in the following table.

XSD Element	gml:AbstractTimeGeometricPrimitive
Type	gml:AbstractTimeGeometricPrimitiveType
BaseType	gml:AbstractTimePrimitiveType
Restriction	<i>Only the Gregorian calendar with UTC times is allowed in the aeronautical domain.</i>
Usage	Represents the supertype of geometric primitives in the temporal dimension: the instant and the period. Allows the definition of the temporal reference system, if the default is not used.
Definition	A temporal geometry shall be associated with a temporal reference system through the frame attribute that provides a URI reference that identifies a description of the reference system. Following ISO 19108, the Gregorian calendar with UTC is the default reference system, but others may also be used.
Comments	<i>none</i>
Used in	Used as direct parent type for TM_Instant and TM_Period.
XML Schema File	temporal.xsd
XML Schema Component	<pre> <element name="AbstractTimeGeometricPrimitive" type="gml:AbstractTimeGeometricPrimitiveType" abstract="true" substitutionGroup="gml:AbstractTimePrimitive"/> <complexType name="AbstractTimeGeometricPrimitiveType" abstract="true"> <complexContent> <extension base="gml:AbstractTimePrimitiveType"> <attribute name="frame" type="anyURI" default="fixed"="#ISO-8601"/> </extension> </complexContent> </complexType> <element name="AbstractTimePrimitive" </pre>

	<pre> type="gml:AbstractTimePrimitiveType" abstract="true" substitutionGroup="gml:AbstractTimeObject"> </element> <complexType name="AbstractTimePrimitiveType" abstract="true"> <complexContent> <extension base="gml:AbstractTimeObjectType"> <sequence> <element name="relatedTime" type="gml:RelatedTimeType" minOccurs="0" maxOccurs="unbounded"/> </sequence> </extension> </complexContent> </complexType> <complexType name="AbstractTimeObjectType" abstract="true"> <complexContent> <extension base="gml:AbstractGMLType"/> </complexContent> </complexType> </pre>
Example	<i>not applicable, because the GML realization of TM_GeometricPrimitive is an abstract type</i>

TM_Instant / gml:TimeInstant – Documentation

The XML Schema implementation of the TM_Instant type (see ISO 19108:2002, section 5.2.3.2) is given by the gml:TimeInstant XSD element and gml:TimeInstantType XSD complex type, documented in the following table.

XSD Element	gml:TimeInstant
Type	gml:TimeInstantType
BaseType	gml:AbstractTimeGeometricPrimitiveType
Restriction	<i>none</i>
Usage	Used to represent an identifiable position in time.
Definition	An instant is a zero-dimensional geometric primitive that represents position in time. It is equivalent to a point in space. In practice, an instant is an interval whose duration is less than the resolution of the time scale.
Comments	Each TimeInstant shall contain one gml:timePosition
Used in	Used as a child of gml:validTime and aixm:featureLifetime. Also used in gml:TimePeriod element, as a child element of gml:begin and gml:end.
XML Schema File	temporal.xsd
XML Schema Component	<pre> <complexType name="TimeInstantType" final="#all"> <complexContent> <extension </pre>

	<pre> base="gml:AbstractTimeGeometricPrimitiveType"> <sequence> <element ref="gml:timePosition"/> </sequence> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:TimeInstant gml:id="IDX"> <gml:timePosition>2010-01- 23T14:00:00Z</gml:timePosition> </gml:TimeInstant> </pre>

TM_Period / gml:TimePeriod – Documentation

The XML Schema implementation of the TM_Period type (see ISO 19108:2002, section 5.2.3.3) is given by the gml:TimePeriod XSD element and gml:TimePeriodType XSD complex type, documented in the following table.

XSD Element	gml:TimePeriod
Type	gml:TimePeriodType
BaseType	gml:AbstractTimeGeometricPrimitiveType
Restriction	The use of child element gml:begin and gml:end are out of scope for AIXM. Use gml:beginPosition and gml:endPosition instead.
Usage	Used to represent an extent in time. <i>In this profile, the beginPosition is considered included in the time interval and the endPosition is considered excluded from the time interval. This is particularly important for temporal queries.</i>
Definition	The period is a one-dimensional geometric primitive that represents extent in time. The period is equivalent to a curve in space. Like a curve, it is an open interval bounded by beginning and end points (instants), and has length (duration). Its location in time is described by the temporal positions of the instants at which it begins and ends; its duration equals the temporal distance between those two temporal positions.
Comments	none
Used in	Used as a child element of gml:validTime and aixm:featureLifetime
XML Schema File	temporal.xsd
XML Schema Component	<pre> <complexType name="TimePeriodType"> <complexContent> <extension base="gml:AbstractTimeGeometricPrimitiveType"> <sequence> <choice> <element name="beginPosition" type="gml:TimePositionType"/> <del element name="begin" type="gml:TimeInstantPropertyType"/> </pre>

	<pre> </choice> <choice> <element name="endPosition" type="gml:TimePositionType"/> <element name="end" type="gml:TimeInstantPropertyType"/> </choice> <group ref="gml:timeLength" minOccurs="0"/> </sequence> </extension> </complexContent> </complexType> </pre>
Example	<pre> <gml:TimePeriod gml:id="IDX"> <gml:beginPosition>2010-01- 23T14:00:00Z</gml:beginPosition> <gml:endPosition>2010-12-23T14:00:00Z</gml:endPosition> </gml:TimePeriod> </pre>

GML Simple Types

The GML profile for aeronautical data also includes a number of simple types, such as NilReasonType, SignType, CodeType, etc. These are copied as such from the corresponding GML schema files in order to have a valid gml321forAIXM.xsd schema file.

Deprecated GML 3.2.1 items

The elements and attributes that are deprecated in GML 3.2.1 (such as gml:coordinates, gml:remoteSchema attribute of the gml:AssociationPropertyGroup, etc. are not included in the profile either.

Annex G - Bibliography

The following documents contain information that was considered in the writing of this document:

[ANNEX 15] ICAO Annex 15 (14th Edition) – Aeronautical Information Services

[AIXM 4.5-5.1] [AIXM 4.5 to AIXM 5.1 Mapping Guidelines \(version 1.1\)](#)

[EPSG CRS] [European Petroleum Survey Group Geodesy Parameters](#)

[UCUM] [Unified Code of Units of Measure](#)

Annex H – Revision History

Date	Release	Author	Paragraph modified	Description
2012-03-22	1.0	Aviation Domain Working Group members	All	Initial version
2015-02-12	1.1	Aviation Domain Working Group members	All	Reorganisation of the document, some content was moved in Annexes. Further work on the GML profile. Added missing ISO 19108 elements. Guidelines for the use of other CRS than EPSG:4326. Removed the possibility of using abstract remote references for Point, as there is no real use case for that. Added some considerations about TimeSlice synchronisation in relation with abstract feature references. Explicit forbid the dynamic redefinition (with xsi:type) for geometrical elements.