

Open Geospatial Consortium

Approval Date: 2012-03-23

Publication Date: 2012-05-15

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

Reference number of this document: OGC 12-028

Editors: **OGC Aviation Domain Working Group**

Guidance and Profile of GML for use with Aviation Data

Copyright © 2012 Open Geospatial Consortium, Inc.

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

Warning

This document is not an OGC Standard. This is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It 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.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Abstract

The ISO 19107 spatial schema, which is implemented by GML, is very complex. ISO 19107 defines an extensive list of geometries, geometric properties and operations – many of which are not necessary for aeronautical information applications. In addition, the ISO 19107 contains an exhaustive 3D geometry model that is probably not needed in its entirety for AIXM either. Therefore, a GML profile for AIXM needs to be defined. The objective of this document is to identify the elements of the AIXM-GML profile and also to provide guidelines for the use of GML constructs in AIXM data sets.

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.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable

1	Introduction.....	1
1.1	Objective	1
1.2	Scope – Applicability	1
1.3	References	1
1.4	Interpretation	1
2	Geographical data in Aeronautical Information	2
2.1	Requirements.....	2
2.2	Geographic vs geometric data	2
2.3	Digital data encoding using AIXM	3
3	WGS-84	4
3.1	Use of srsName	4
3.2	Use of global srsName	5
4	Positions.....	8
4.1	Background	8
4.2	GML encoding	8
5	Lines and Surfaces	10
5.1	Background	10
5.2	GML encoding	11
5.2.1	Straight lines	11
5.2.2	Parallels.....	12
5.2.3	Arc by edge	13
5.2.4	Arc by centre point.....	15
5.2.5	Circle by center point.....	20
5.2.6	Corridor.....	21
5.2.7	Perimeter encoding direction	22
5.2.8	Other geometries.....	24
6	Airspace aggregation	26
6.1	Background	26
6.2	GML encoding	26
6.2.1	By reference.....	27
6.2.2	By copying the geometry	29
6.2.3	Combined method.....	31
7	Point references and annotations	32
7.1	Background	32
7.2	GML encoding	33
7.2.1	Using aixm:Point annotations	34
7.2.2	Using xlink:href	35
8	Geographical border references	38
8.1	Background	38
8.2	GML encoding	39

8.2.1	Using aixm:Curve annotations.....	39
8.2.2	Using xlink:href.....	41
9	AIXM GML Profile.....	45
9.1	Introduction.....	45
9.2	General Information.....	45
9.2.1	XML Namespaces.....	45
9.2.2	General Comments.....	45
9.2.3	AIXM GML Profile Types tables.....	46
9.3	AIXM Types.....	48
9.3.1	Overview.....	48
9.3.2	AIXM Point – Documentation.....	49
9.3.3	AIXM ElevatedPoint – Documentation.....	51
9.3.4	AIXM Curve – Documentation.....	52
9.3.5	AIXM ElevatedCurve – Documentation.....	54
9.3.6	AIXM Surface – Documentation.....	56
9.3.7	AIXM ElevatedSurface – Documentation.....	57
9.4	ISO 19107 and ISO 19136 (GML) Types.....	59
9.4.1	Overview.....	59
9.4.2	DirectPosition / gml:pos – Documentation.....	64
9.4.3	GM_Point / gml:Point – Documentation.....	66
9.4.4	GM_Envelope / gml:Envelope – Documentation.....	67
9.4.5	GM_PointRef / gml:pointProperty – Documentation.....	68
9.4.6	GM_Position / gml:geometricPositionGroup – Documentation.....	70
9.4.7	GM_PointArray / gml:posList – Documentation.....	71
9.4.8	gml:AbstractCurve – Documentation.....	72
9.4.9	GM_Curve / gml:Curve – Documentation.....	74
9.4.10	GM_CurveSegment / gml:AbstractCurveSegment – Documentation.....	75
9.4.11	gml:ArcByCenterPoint – Documentation.....	77
9.4.12	gml:CircleByCenterPoint – Documentation.....	79
9.4.13	GM_Arc / gml:Arc – Documentation.....	81
9.4.14	GM_Circle / gml:Circle – Documentation.....	82
9.4.15	GM_GeodesicString / gml:GeodesicString – Documentation.....	83
9.4.16	GM_Geodesic / gml:Geodesic – Documentation.....	85
9.4.17	GM_LineString / gml:LineStringSegment – Documentation.....	86
9.4.18	GM_Surface / gml:Surface – Documentation.....	88
9.4.19	GM_SurfacePatch / gml:AbstractSurfacePatch – Documentation.....	89
9.4.20	GM_Polygon / gml:PolygonPatch – Documentation.....	90
9.4.21	gml:AbstractRing – Documentation.....	92
9.4.22	GM_Ring / gml:Ring – Documentation.....	93
9.4.23	GM_OrientableCurve / gml:OrientableCurve – Documentation.....	95
9.4.24	GM_CompositeCurve / gml:CompositeCurve – Documentation.....	97
10	Interpolation and Densification Considerations.....	99
10.1	Background.....	99

10.1.1	Comparison of models	99
10.2	Mapping of GML to Simple Geometry	100
10.2.1	Flattening of geometry structure	100
10.2.2	Densification of curves	100
10.2.3	Loss of data structure	102
Annex A - ArcByCenterPoint Interpretation Summary.....		103
Annex B - Mapping for ArcByCenterPoint		113
Annex C – GML change request – support arbitrary rhumb lines.....		116
Annex D – Considerations about interpolations		117
Annex E – Offset Curve and Airspace Corridor encoding issues.....		120

i. Preface

This paper provides guidelines for some aspects of the GML encoding that are specific to the aviation data domain.

ii. Document terms and definitions

This document uses the standard terms defined in Subclause 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 standard.

iii. Document contributor contact points

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

Name	Role	Organization
David Burggraf	Contributor	Galdos, Inc.
Eddie Curtis	Contributor	Snowflake Software
Warwick Dufour	Contributor	Avitech AG
Johannes Echterhoff	Editor	iGSI
Davide Castagni Fabbri	Contributor	IDS Ingegneria Dei Sistemi S.p.A
Benoit Geffroy	Contributor	EgisAvia
Francois Germain	Contributor	Thales
Razvan Guleac	Contributor	EUROCONTROL
Alain Kabamba	Contributor	Erdas
Michal Kadlec	Contributor	Avitech AG
Antonio Locandro	Contributor	COCESNA
Eduard Porosnicu	Editor	EUROCONTROL
Bert Robben	Contributor	Luciad
Timo Thomas	Contributor	Comsoft
Scott Wilson	Contributor	EUROCONTROL

1 Introduction

1.1 Objective

The AIXM 5.1 schema uses 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 by GML, is very complex. The standard contains an extensive list of geometries, geometric properties and operations – many of which are not necessary for aeronautical information applications. In addition, the ISO 19107 contains an exhaustive 3D geometry model that is probably not needed in its entirety for AIXM either. Therefore, a GML profile for AIXM needs to be defined. The objective of this document is to identify the elements of the AIXM-GML profile and also to provide guidelines for the use of GML constructs in AIXM data sets.

1.2 Scope – Applicability

The discussion paper focuses on guidelines for a aeronautical data encoding using the Aeronautical Information Exchange Model (AIXM).

1.3 References

- [1] ICAO Annex 15 (13th Edition) – Aeronautical Information Services
- [2] ISO 19107:2003 – Geographic information — Spatial schema
- [3] ISO 19136:2007 – Geography Markup Language
- [4] OGC 07-092r3 - [Definition identifier URNs in OGC namespace](#)
- [5] OGC 06-042 - [OpenGIS® Web Map Server Implementation Specification](#)
- [6] EAD AIXM4.5-to-5.1_Mapping.doc
- [7] EAD AIXM 5.1 Conceptual Manual 0.2.doc

1.4 Interpretation

The following definitions shall apply:

- ‘data set’ means identifiable collection of data

2 Geographical data in Aeronautical Information

2.1 Requirements

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

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

2.2 Geographic vs geometric data

Such as many other classes of applications, aeronautical applications deal with entities which have a geographic extent. By definition, hence, software systems supporting aeronautical applications are Geographic Information Systems (GIS). It's worth to distinguish between the adjectives "geometric" and "geographic".

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. The most known example of metric space, familiar to everyone, is the n-dimensional Euclidean space E^n , namely the real vector space R^n endowed with the usual Euclidean metric (where the distance between any couple of 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 (e.g. by comparison with a rigid sample), once a length unit of measure has been defined, e.g. meter. This metric is the Euclidean-Pythagorean one for Earth Centered Earth Fixed (ECEF) coordinates.

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

It also happens, in many applications, that the horizontal aspect of geographic entities is by far more relevant than the vertical one. Hence geographic entities are simply described as 2D geometric objects: the orthogonal projection of 3D entities onto the Earth surface.

Here comes the subtlety: 2D geographic entities live in a curved world, the Earth surface. Curved means that the metric we adopt to measure distances, angles and areas on that surface cannot be brought back to the Euclidean metric on \mathbb{R}^2 (mathematicians say that “a curved surface is not isomorphic to the flat space \mathbb{E}^2 ”): we cannot find any CRS parameterizing the surface, such that the distance between any couple of points is given by 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).

2.3 Digital data encoding using AIXM

The Aeronautical Information Exchange Model (AIXM) is used since 2003 for the provision of AI in digital format. Initially developed for the European AIS Database, AIXM is being progressively adopted by other States world-wide, also encouraged by the ICAO work towards the adoption of AIXM as ICAO Guidance Material, through the AIS-AIM Sub Group (<http://www2.icao.int/en/ais-aimsg/Lists/Meetings/AllItems.aspx>).

The AIXM versions up to 4.5 used a custom XML encoding, not based on OGC standards. Since 2008, with the publication of version 5, the AIXM specification has embraced GML as data encoding format.

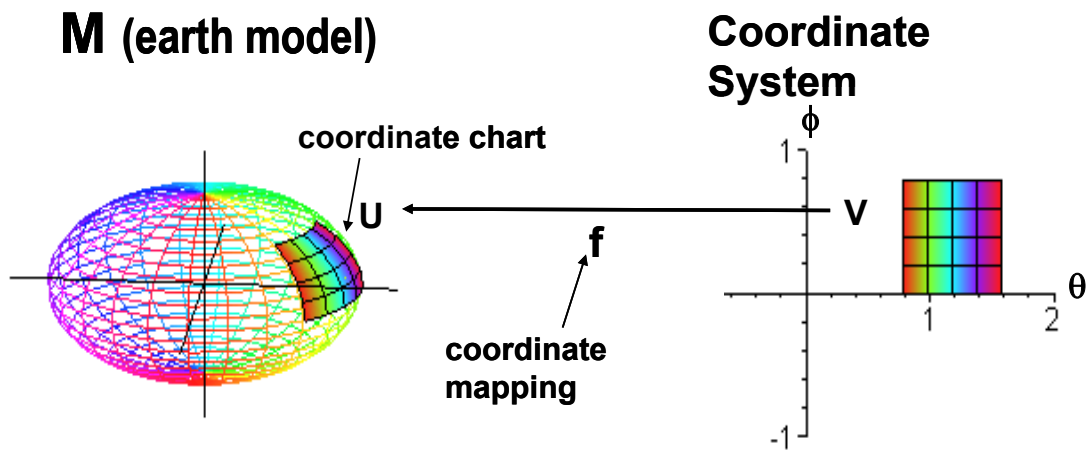
There are a number of specificities in the AI Domain that concern the provision of geographical and geometrical information. These are discussed in this document, which also includes recommendations for data encoding in GML.

3 WGS-84

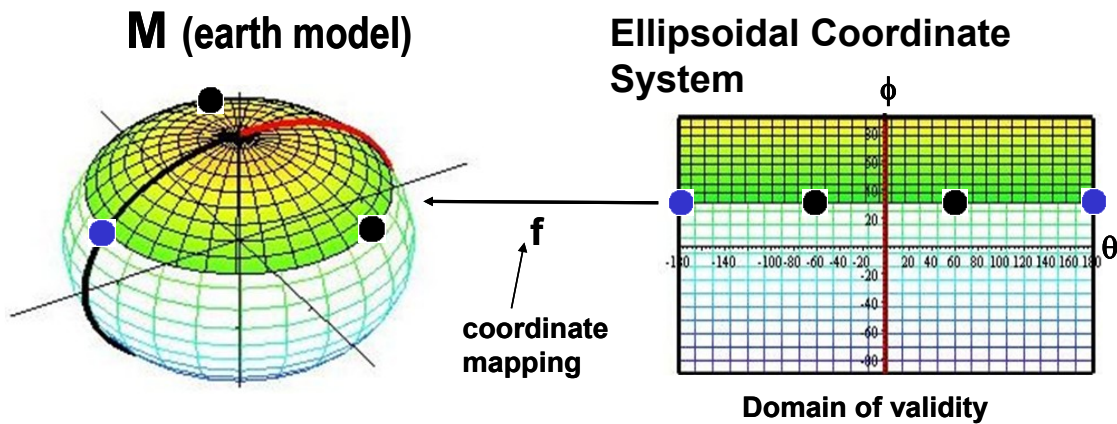
According to ICAO Annex 15, all “*published aeronautical geographical coordinates (indicating latitude and longitude) shall be expressed in terms of the WGS-84 geodetic reference datum*”.

3.1 Use of 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.



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



The CRS reference is critical for the correct encoding and processing of the geographical data contained in AIXM/GML files. It 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 OGC themselves. 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].

Because of the way that angle directions are measured in the AI domain (North corresponds to 0°, East to 90°, etc.), the OGC:CRS 84 is not appropriate for AIXM 5.1/GML data sets that contain arcs of circle defined by start angle/end angle. This will be explained in section 5.2.4.1 of this document. Therefore, it is generally recommended that the EPSG:4326 CRS is used in AIXM 5.1 data sets, which use the WGS-84 reference datum required by ICAO. However, this does not exclude the use of other CRS when appropriate.

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 URN], which provides the following URN identifier for the EPSG:4326 CRS: *urn:ogc:def:crs:EPSG::4326*.

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">
```

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

3.2 Use of global srsName

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

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, unless overruled by the presence of a local `srsName`. The `gml:Envelope` is a child element of the `gml:boundedBy` property of the feature, as indicated in Figure 1. Thus it is not necessary for a geometry to carry a `srsName` attribute, if it uses the same coordinate reference system as given on the `gml:boundedBy` property of its parent feature.

Inheritance of the coordinate reference system continues to any depth of nesting, but 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, in order to indicate the reference system used locally, even if they are the same as the parent. A geometry without `srsName` derives its CRS from its closest ancestor that has an `srsName`. Because of the way that AIXM is built, this can only be one of the following:

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

If no such ancestor is found, it 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.

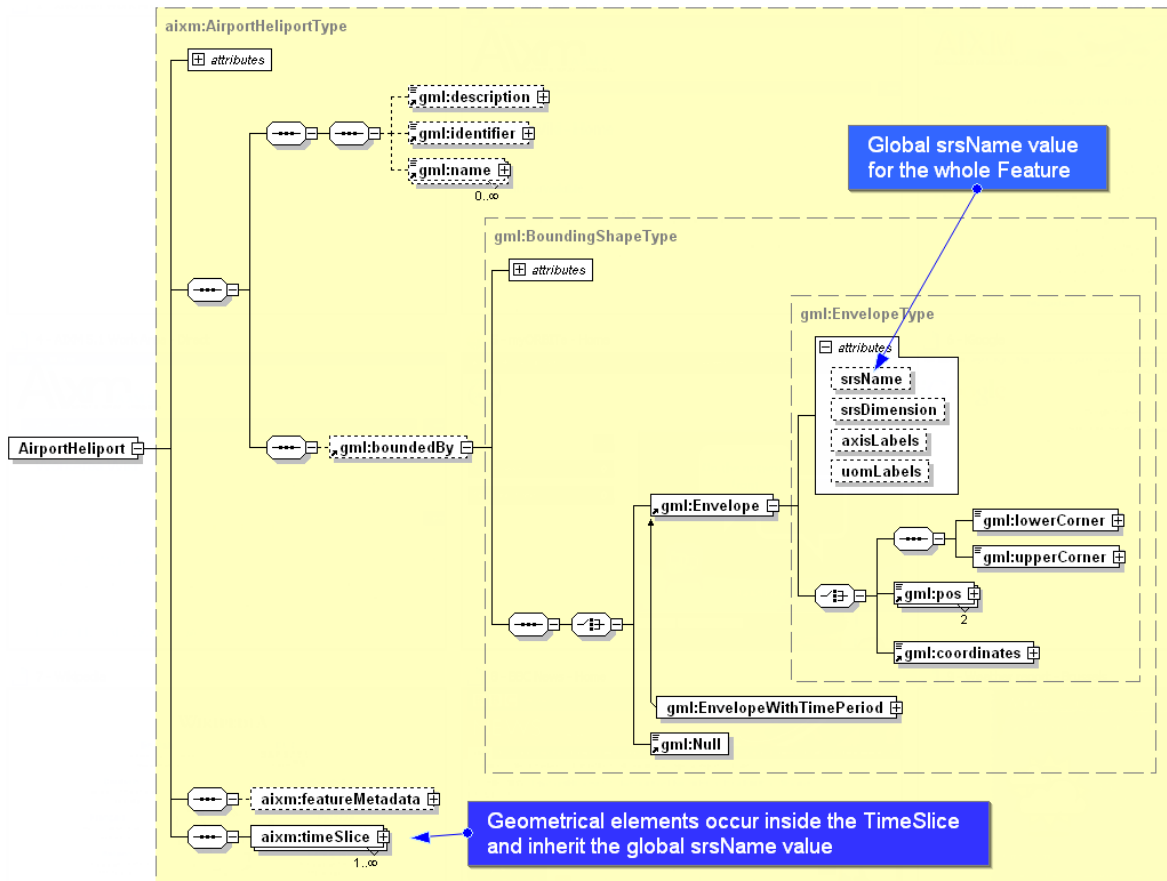


Figure 1 - Global srsName specified using gml:Envelope

4 Positions

4.1 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, they are expressed as a pair of latitude/longitude coordinates. The information about the geodetic reference datum is usually not provided for each individual position, being specified once for the whole AIP.

4.2 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, as in the following example:

```

...
<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 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, as it can be seen in the example of the prohibited area definition from section 5.1. Not surprisingly, the WMS specification 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.

For information, note that in the previous AIXM 4.5 version (not based on GML) positional data was encoded using AIXM-defined XML elements, where WGE designates the WGS-84 reference datum:

```
...  
<geoLat>52.2889</geoLat>  
<geoLong>-32.0350</geoLong>  
<codeDatum>WGE</codeDatum>  
...
```

5 Lines and Surfaces

5.1 Background

Certain features in the AI domain have polygonal shape (such as Airspace) or linear shape (such as a power line VerticalStructure). 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)
 52°11'08.00"N 005°12'30.00"E;
 52°12'22.00"N 005°17'15.00"E;
 52°11'21.00"N 005°17'56.00"E;
 52°10'09.00"N 005°17'56.00"E;
 (then along the parallel to) 52°10'09.00"N 005°13'11.00"E;
 to point of origin.

Usually, it is not indicated in the AI source documents what kind of interpolation is used for the curve between the consecutive points, 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 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
 53°10'12.59"N 004°46'21.14"E; *along clockwise arc (radius 8 NM, centre 53°15'00.00"N 004°57'00.00"E) to*
53°07'01.98"N 004°56'02.41"E; 53°11'00.00"N 004°51'24.00"E; to point of origin.

EHR 4B (VLIEHORS)
 53°09'43.06"N 005°06'58.79"E; 53°02'40.00"N 005°15'00.00"E; 52°58'09.00"N 005°06'22.00"E; 53°07'01.98"N
 004°56'02.41"E; *along anti-clockwise arc (radius 8 NM, centre 53°15'00.00"N 004°57'00.00"E) 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).

5.2 GML encoding

5.2.1 Straight lines

It is recommended that `gml:GeodesicString` is used as default encoding for straight lines, for the following reasons:

- 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).

5.2.2 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 can 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 2.

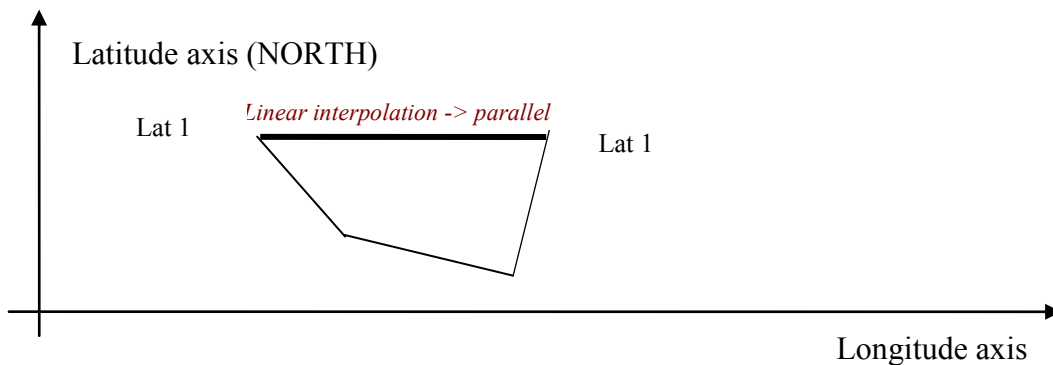


Figure 2 - Linear interpolation in a geodetic CRS

It shall be noted 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). Note that a change request has been raised with OGC (see Annex C) in order to enable specifying additional interpolations, such as rhumbline.

In conclusion:

- in the classical aeronautical information case of two consecutive points having the same latitude, a `gml:LineStringSegment` with "default" CRS EPSG:4326 should be used for encoding.
- 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 can be used.

5.2.3 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 3 - Arc by edge point:

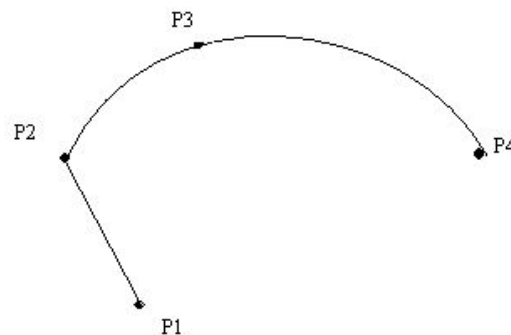


Figure 3 - Arc by edge point

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>
...

```

In fact, this is a particular case of the more general GML concept of “ArcString”¹, 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 numArc is the property defining the number of the arcs in the string, such as in Figure 4 - ArcString in ISO 19107

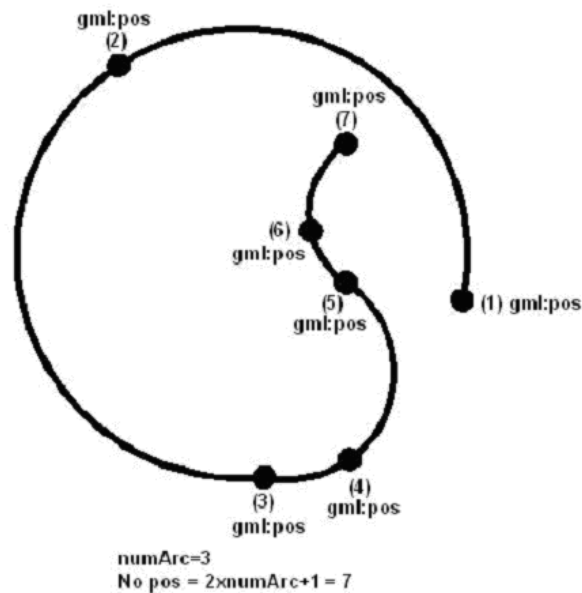


Figure 4 - ArcString in ISO 19107

For information, the AIXM 4.5 way of encoding the arc by edge point using AIXM-defined elements is provided below:

```

...
<Avx>
  <codeType>ABE</codeType>
  <geoLat>lat-P2</geoLat>
  <geoLong>long-P2</geoLong>
  <codeDatum>WGE</codeDatum>
  <geoLatArc>lat-P3</geoLatArc>

```

¹ Note that ArcString is not foreseen by the AIXM GML Profile, as defined in Chapter 9 of this document.

```
<geoLongArc>long-P3</geoLongArc>
</Avx>
...
```

5.2.4 Arc by centre point

Currently, this is the typical construct for arcs used in the definition of airspace borders in the AI domain. 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.

5.2.4.1 Measuring angles in GML

GML explicitly implements² 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 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.*”

² 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 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).

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 the EPSG:4326 geodetic CRS actually corresponds to a clockwise rotation in the AI domain.

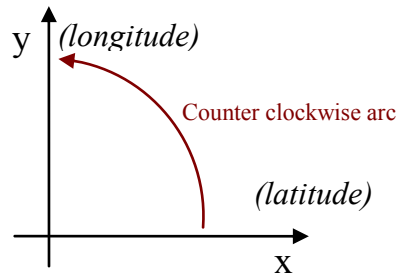


Figure 5 - Angle measured in a 2D geodetic CRS that has latitude as first axis

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 6. Practically, the x/y reference system is mirrored and rotated to be aligned with the meridians and the parallels.

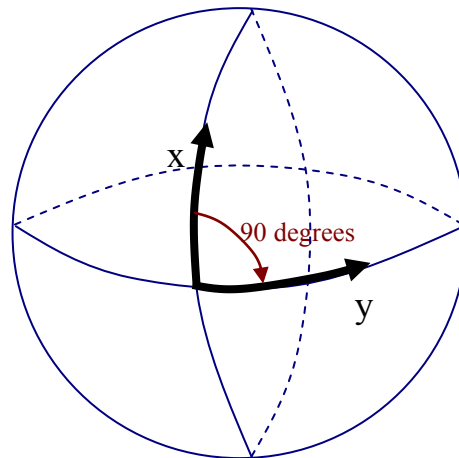
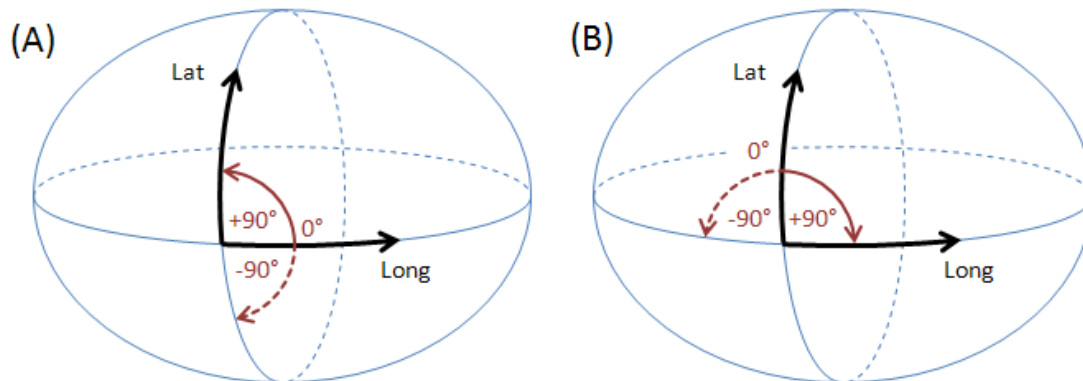


Figure 6 - Coordinate System Axes of EPSG:4326 CRS mapped on the Earth's surface

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:



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

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

(B) GeodeticCRS: urn:ogc:def:crs:EPSG::4326

- 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 (on the positive part of the coordinate system's first axis).

5.2.4.2 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 A.

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

- *clockwise on the surface of the Earth when the startAngle is lower than the endAngle;*
- *counter-clockwise on the surface of the Earth when the startAngle is higher than the endAngle*

The same convention applies to any other geodetic CRS that has latitude as first (x) axis. This is exemplified in Figure 7. 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.

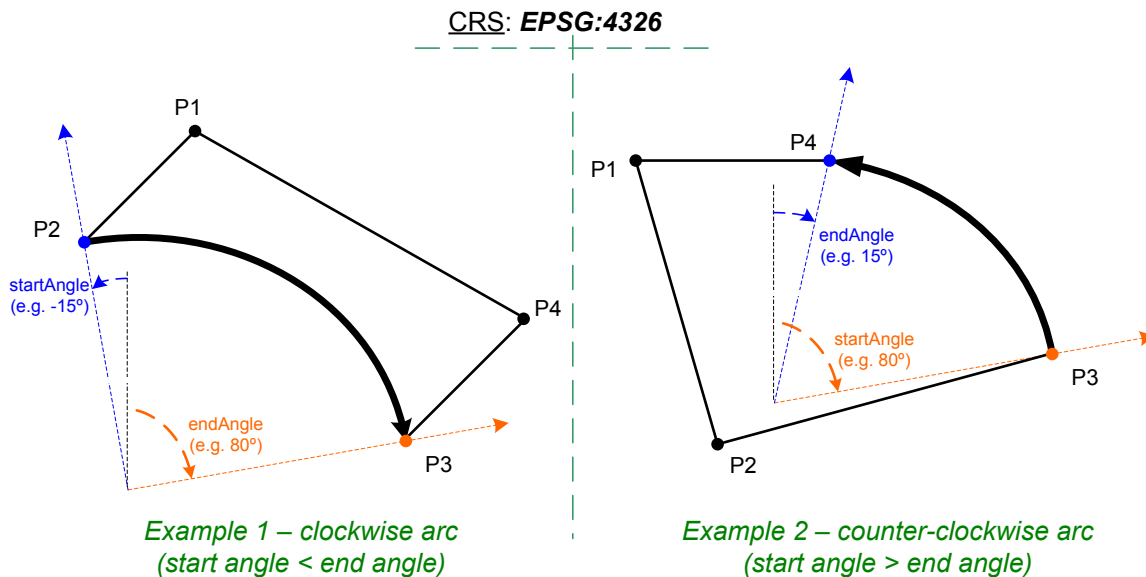


Figure 7 - Arcs are drawn from startAngle to endAngle

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>
...

```

5.2.4.3 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, it is recommended that conventional units of measure symbols are 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) (<http://aurora.regenstrief.org/UCUM>). 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 are recommended to 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**” is recommended to be used for the `uom` attribute of `gml:startAngle` and `gml:endAngle` elements.

5.2.4.4 Old AIXM 4.5 encoding

For information, the same border using a clockwise arc was encoded in AIXM 4.5 as in the example below:

```

...
<Avx>
  <codeType>GRC</codeType>
  <geoLat>lat-P1</geoLat>
  <geoLong>long-P1</geoLong>
  <codeDatum>WGE</codeDatum>
</Avx>
<Avx>
  <codeType>CWA</codeType>
  <geoLat>lat-P2</geoLat>
  <geoLong>long-P2</geoLong>
  <codeDatum>WGE</codeDatum>
  <geoLatArc>lat-P3</geoLatArc>
  <geoLongArc>long-P3</geoLongArc>
  <valRadiusArc>Radius</valRadiusArc>
  <uomRadiusArc>NM</uomRadiusArc>
</Avx>
<Avx>
  <codeType>GRC</codeType>
  <geoLat>lat-P4</geoLat>
  <geoLong>long-P4</geoLong>
  <codeDatum>WGE</codeDatum>
</Avx>
...

```

5.2.5 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:polygonPatches>
      <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>
...

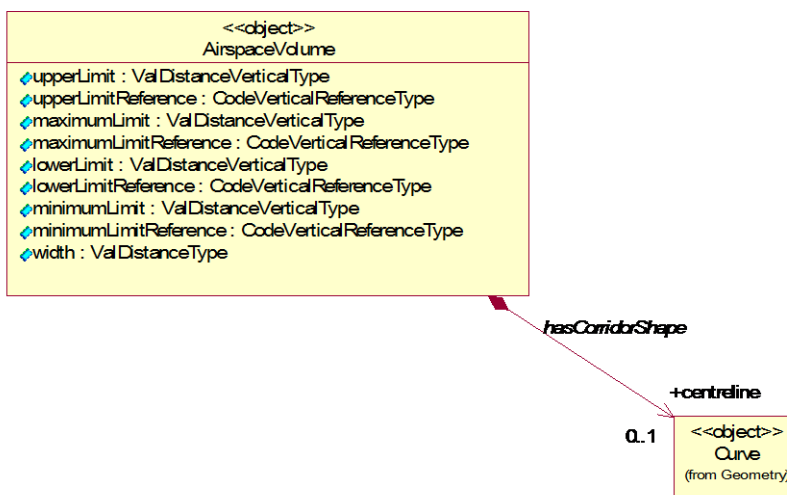
```

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.

5.2.6 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



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`, it is currently not recommended to use `OffsetCurve`. Further details are presented in Annex E.

5.2.7 Perimeter encoding direction

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

For AI Domain applications, this implies that the outside perimeter and any eventual holes shall be encoded as shown in Figure 8. It is unlikely that more than one level of internal patches (rings) would exist for features in the AI Domain.

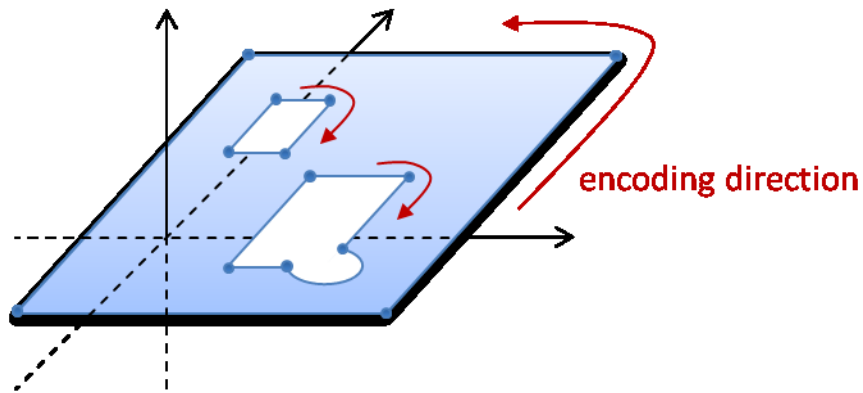


Figure 8 - Encoding direction for surface boundaries

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.

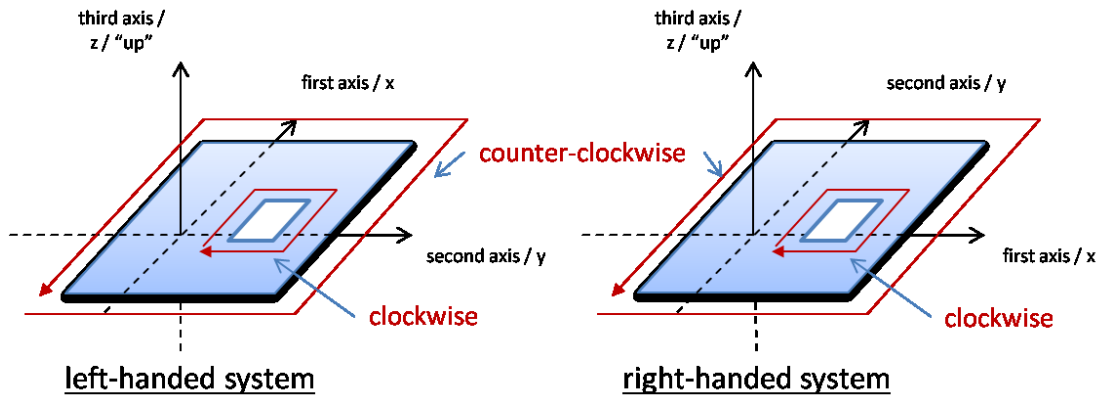


Figure 9 - The boundary encoding is the same in both left- and right-handed systems

Figure 8 shows a surface with two holes, one of which has an arc as part of its boundary. The direction of this arc needs to be in line with the overall orientation of the boundary. This can unambiguously be achieved following the rules laid out in section 5.2.4 and Annex A. The following figure shows some examples where arcs are part of a surface's boundary.

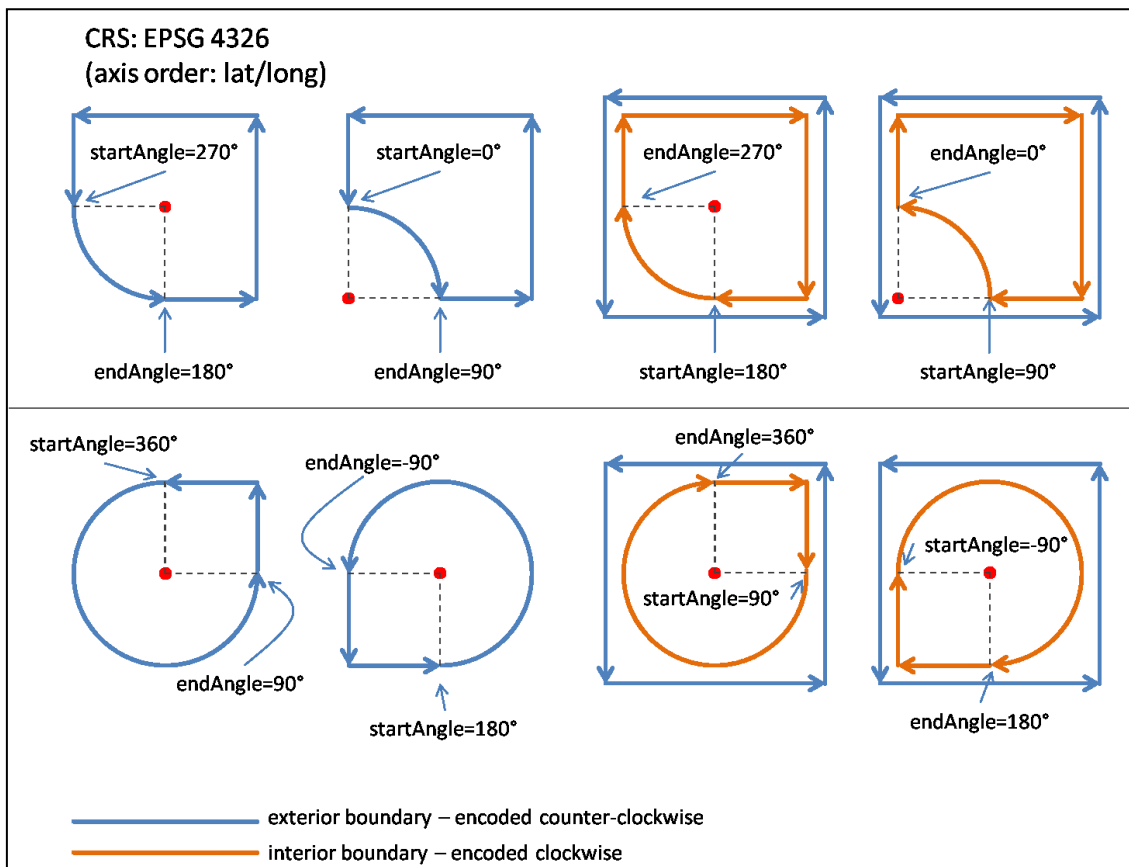
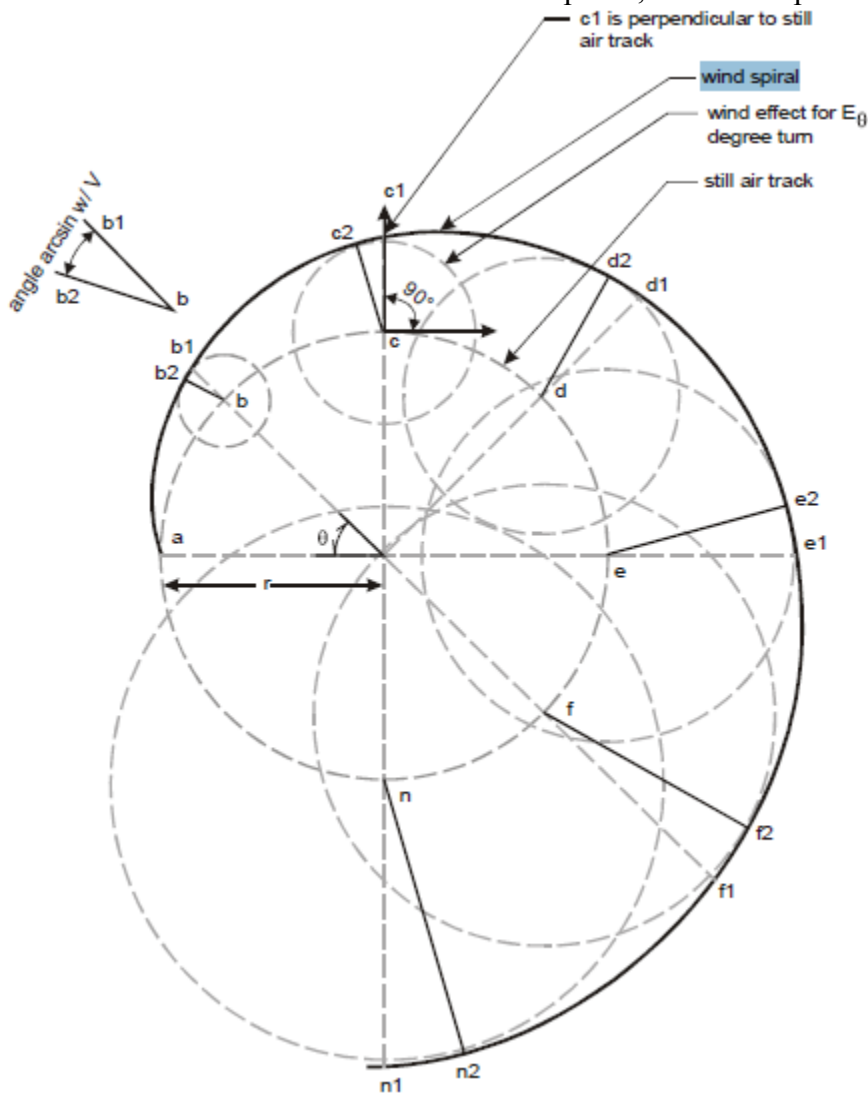


Figure 10 - arcs and their direction when being part of a surface boundary

5.2.8 Other geometries

Some specific geometrical constructs are used in the design of instrument approach/departure produres 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



11

Figure 11 - Wind Spiral

Other such geometrical constructs are the “segment locus” (Figure 12) and “arc locus” (Figure 13).

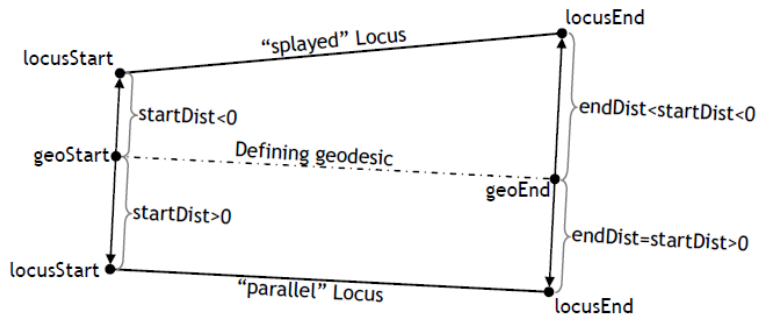


Figure 12 - Segment Locus (a “parallel” and a “splaying” locus with respect to a given geodesic line.)

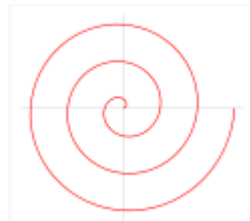


Figure 13 - Arc Locus

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.

6 Airspace aggregation

6.1 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.

6.2 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.

6.2.1 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 14 indicates which elements of the AIXM 5.1 model are used or not used when applying this method.

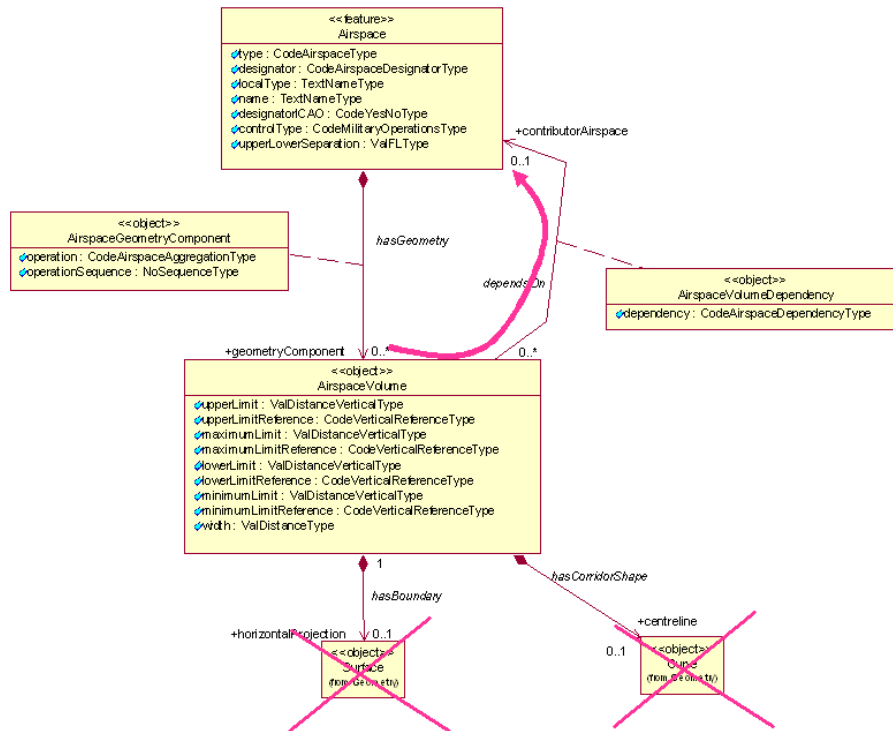


Figure 14 - Model use for airspace aggregation by reference

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>
...

```

6.2.2 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.

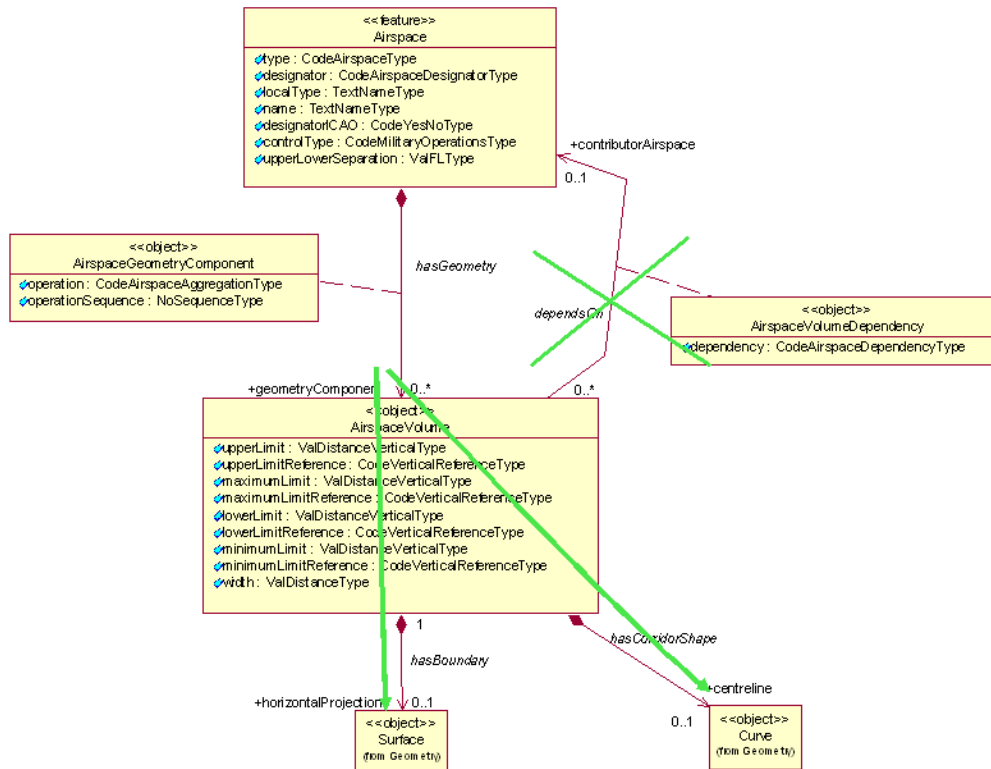


Figure 15 - Model use for airspace aggregations when copying referenced geometries

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:

```

...
<airm:type>CTA</airm:type>
<airm:designator>EADD</airm:designator>
<airm:name>CTA DONLON</airm:name>
<airm:geometryComponent>
  <airm:AirspaceGeometryComponent gml:id="AV001">
    <airm:operation>BASE</airm:operation>
    <airm:operationSequence>1</airm:operationSequence>
    <airm:theAirspaceVolume>
      <airm:AirspaceVolume gml:id="V001">
        <airm:upperLimit uom="FL">245</airm:upperLimit>
        <airm:upperLimitReference>STD</airm:upperLimitReference>
        <airm:lowerLimit uom="FL">30</airm:lowerLimit>
        <airm:lowerLimitReference>STD</airm:lowerLimitReference>
        <airm:horizontalProjection>
          <airm:Surface gml:id="S001">
            <gml:patches>
              <gml:PolygonPatch>
                <gml:exterior>
                  <gml:Ring>
                    <gml:curveMember>
                      <airm: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>
                      </airm:Curve>
                    </gml:curveMember>
                  </gml:Ring>
                </gml:exterior>
              </gml:PolygonPatch>
            </gml:patches>
          </airm:Surface>
        </airm:horizontalProjection>
      </airm:AirspaceVolume>
    </airm:theAirspaceVolume>
  </airm:AirspaceGeometryComponent>
</airm:geometryComponent>
<airm:geometryComponent>
  <airm:AirspaceGeometryComponent gml:id="AV002">
    <airm:operation>UNION</airm:operation>
    <airm:operationSequence>2</airm:operationSequence>
    <airm:theAirspaceVolume>
      <airm:AirspaceVolume gml:id="V002">
        <airm:upperLimit uom="FL">245</airm:upperLimit>
        <airm:upperLimitReference>STD</airm:upperLimitReference>
        <airm:lowerLimit uom="FL">50</airm:lowerLimit>
        <airm:lowerLimitReference>STD</airm:lowerLimitReference>
        <airm: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>
...

```

6.2.3 Combined method

The two methods can also be combined. However, as for the method described in 6.2.2, this is only appropriate for SNAPSHOT TimeSlices.

7 Point references and annotations

7.1 Background

Positions of Nav aids 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:

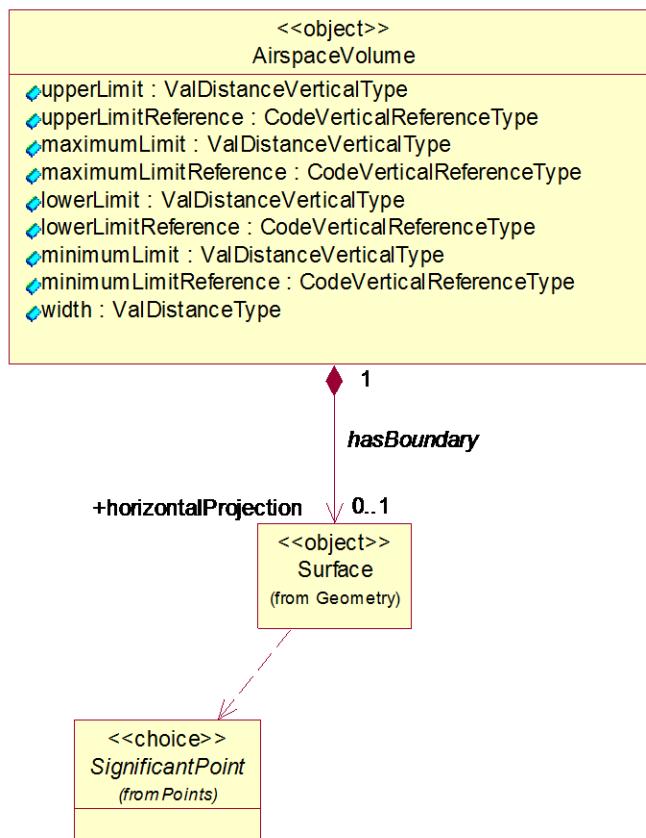


Figure 16 - Point references model in AIXM

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.

7.2 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.**”

7.2.1 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>
          <!-- 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>
...

```

7.2.2 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. 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.

7.2.2.1 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:theAirspaceVolume>
  ...
</aixm:Airspace>

```

```

        </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 7.2.2.2). Only for data export/import purpose the reference would be towards the `gml:Point` directly.

7.2.2.2 Abstract reference to remote feature

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

```

<aixm:Airspace gml:id="urn:uuid:c4241c79-d7a8-4b4b-a997-88a802557138">
  ...
  <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="urn:uuid:791fb712-6c7a-46bb-8e98-49d76942573e"
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:polygonsPatches>
            </aixm:Surface>
          </aixm:horizontalProjection>
        </aixm:AirspaceVolume>
      </gml:Curve>
    </gml:segments>
  </gml:CircleByCenterPoint>
  ...
</aixm:Airspace>

```

The `xlink:href` value is a remote reference to the `gml:identifier` of the Navaid that has the lat/long position used as circle centre as its location. A native GML tool might identify this as an error, because the `aixm:Navaid` is not in the substitution group of `gml:Point`. 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.

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

- as a simple annotation (see 7.2.1)
- as a local concrete `xlink:href` reference using `gml:id` (see 7.2.2.1)
- as a remote abstract `xlink:href` using `gml:identifier` (see 7.2.2.2)

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.

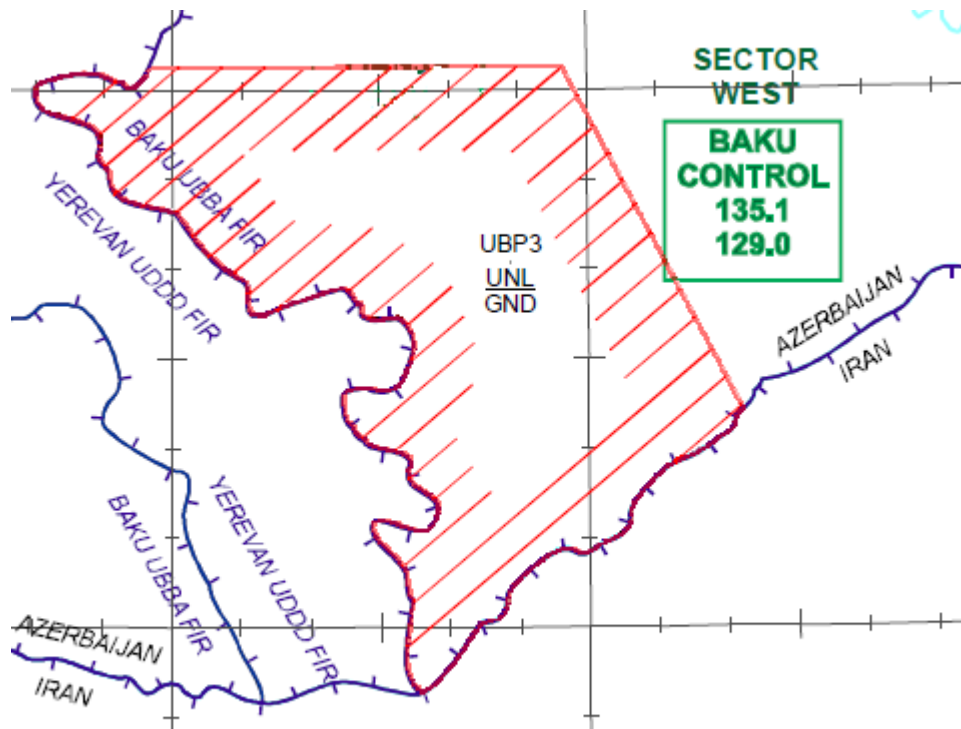
8 Geographical border references

8.1 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”*



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:

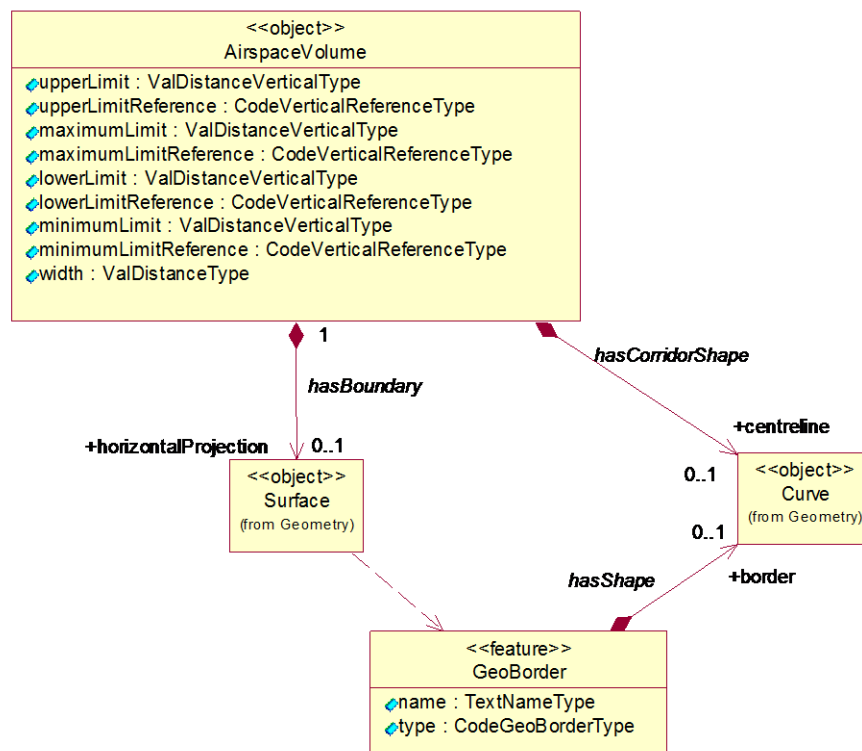


Figure 17 - GeoBorder references model in AIXM

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.

8.2 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.

8.2.1 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>
                <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:curveMember>
            </gml:Ring>
          </gml:PolygonPatch>
        </gml:polygonPatches>
      </aixm:Surface>
    </aixm:horizontalProjection>
  </aixm:Airspace>

```



```

    </aixm:Curve>
    </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.”*

8.2.2 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.

8.2.2.1 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-hoc³” 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">

```

³ 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.

```

        <!-- 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>
        <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.

8.2.2.2 Abstract reference to remote feature

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>
                  </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 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.

9 AIXM GML Profile

9.1 Introduction

The use of the models and encodings defined by ISO 19107 as well as ISO 19136 facilitates the interoperable exchange of spatial information in and between various application domains. However, to satisfy the needs of multiple domains regarding the representation of spatial information, the amount of concepts and functionality covered by the two standards is rather wide. Only a specific subset of the functionality defined by these standards is relevant to the Aviation domain.

This chapter thus defines a profile of the ISO 19107 / ISO 19136 types, describing the XML Schema (XSD) implementation of the types that are expected to be used within AIXM encoded information. This subset of ISO 19107 / ISO 19136 types needs to be supported by any application that intends to process AIXM data.

9.2 General Information

9.2.1 XML Namespaces

This chapter uses a number of XML namespace prefixes; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 1: Prefixes and Namespaces used in the AIXM GML Profile chapter

Prefix	Namespace
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

9.2.2 General Comments

The following comments apply to all types documented in this profile:

- The Coordinate Reference System (CRS) reference is critical for the correct encoding and processing of AIXM/GML geometries, as it 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.

It is generally recommended that the EPSG:4326 Coordinate Reference System (CRS) is used in AIXM 5.1 data sets, which use the WGS-84 reference datum required by ICAO and has latitude/longitude axis order. However, this does not exclude the use of other CRS when appropriate.

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

The srsDimension attribute should not be used in AIXM geometries.

See section 3 for further details.

- Geographic coordinates, i.e. latitude and longitude, shall be expressed in decimal degrees, not in degrees minutes seconds. More detailed information on conversion of the coordinate format is provided in section 4.2.

9.2.3 AIXM GML Profile Types tables

The following tables provide an overview of the conceptual types and their XML Schema implementation that is documented in this profile.

Table 2 – AIXM Conceptual Types and the relevant XSD Implementation to document

AIXM Conceptual Type	AIXM XSD Implementation (Element and Type)	Section Reference
Point	aixm:Point, -Type	9.3.2
ElevatedPoint	aixm:ElevatedPoint, -Type	9.3.3
Curve	aixm:Curve, -Type	9.3.4
ElevatedCurve	aixm:ElevatedCurve, -Type	9.3.5
Surface	aixm:Surface, -Type	9.3.6
ElevatedSurface	aixm:ElevatedSurface, -Type	9.3.7

Table 3 – ISO conceptual types and the relevant XSD implementation to document

ISO 19107 Type	ISO 19136 / GML Implementation (Element and Type)	Section Reference
DirectPosition	gml:pos, gml:DirectPositionType	9.4.2
GM_Point	gml:Point, -Type	9.4.3
GM_Envelope	gml:Envelope, -Type	9.4.4
GM_PointRef	gml:pointProperty, gml:PointPropertyType	9.4.5
GM_Position	<i>Defined in GML as a group, not as an element/type:</i> gml:geometricPositionGroup	9.4.6
GM_PointArray	gml:posList, gml:DirectPositionListType	9.4.7
-	gml:AbstractCurve, -Type	9.4.8
GM_Curve	gml:Curve, -Type	9.4.9
GM_CurveSegment	gml:AbstractCurveSegment, -Type	9.4.10
-	gml:ArcByCenterPoint, -Type	9.4.11
-	gml:CircleByCenterPoint, -Type	9.4.12
GM_Arc	gml:Arc, -Type	9.4.13
GM_Circle	gml:Circle, -Type	9.4.14
GM_GeodesicString	gml:GeodesicString, -Type	9.4.15
GM_Geodesic	gml:Geodesic, -Type	9.4.16
GM_LineString	gml:LineStringSegment, -Type	9.4.17
GM_Surface	gml:Surface, -Type	9.4.18
GM_SurfacePatch	gml:AbstractSurfacePatch, -Type	9.4.19

ISO 19107 Type	ISO 19136 / GML Implementation (Element and Type)	Section Reference
GM_Polygon	gml:PolygonPatch, -Type	9.4.20
-	gml:AbstractRing, -Type	9.4.21
GM_Ring	gml:Ring, -Type	9.4.22
GM_OrientableCurve	gml:OrientableCurve, -Type	9.4.23
GM_CompositeCurve	gml:CompositeCurve, -Type	9.4.24

9.3 AIXM Types

9.3.1 Overview

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.

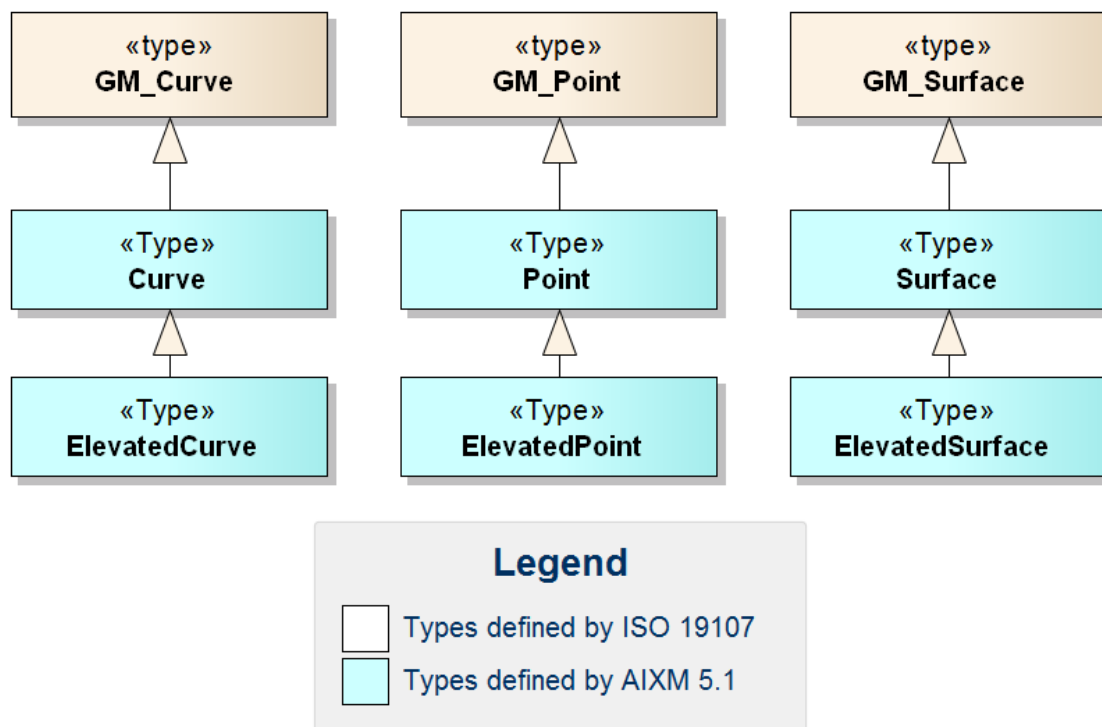


Figure 18 – Overview of AIXM Geometry Types

The XML Schema implementation of these AIXM types is documented in the following sections. The XSD implementation of the types from ISO 19107 is documented in section 9.4.

9.3.2 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 (see section 9.4.3)
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 – see section 7.2.1 for further details
Used in	Used in a number of AIXM features. Can also be used as a substitute for GM_Point / gml:Point (see section 9.4.3).
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> <aixm:LinguisticNote gml:id="N002"> <aixm:note lang="ENG">VILLAGE JAKOVO</aixm:note> </aixm:LinguisticNote> </aixm:translatedNote> </aixm:Note> </aixm:annotation> </aixm:Point> </pre>

→ back to *AIXM GML Profile Types tables*

9.3.3 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 (see section 9.3.2)
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 (see section 9.3.2).
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> </complexType> </element> </sequence> </extension> </complexContent> </complexType></pre>

	<pre> <attributeGroup ref="gml:OwnershipAttributeGroup"/> </complexType> </element> </sequence> </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>
<p>Example</p>	<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>

→ back to *AIXM GML Profile Types tables*

9.3.4 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 (see section 9.4.9)
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 – see section 8.2.1 for further details
Used in	Used in some AIXM features. Can also be used as a substitute for GM_Curve / gml:Curve (see section 9.4.9).
XML Schema File	AIXM_Features.xsd
XML Schema Component	<pre> <element name="Curve" type="aixm:CurveType" substitutionGroup="gml:Curve"/> <complexType name="CurveType"> <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>
Example	<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"> </pre>

	<pre> <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>
--	---

→ back to *AIXM GML Profile Types tables*

9.3.5 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 (see section 9.3.4)
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 (see section 9.3.4).
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> </pre>

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

→ back to *AIXM GML Profile Types tables*

9.3.6 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 (see section 9.4.18)
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 (see section 9.4.18) 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 (see section 9.4.18).
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> <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>
---------	--

→ back to *AIXM GML Profile Types tables*

9.3.7 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 (see section 9.3.6)
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 (see section 9.3.6).
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"> <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"/> </pre>

	<pre> </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>

→ back to *AIXM GML Profile Types tables*

9.4 ISO 19107 and ISO 19136 (GML) Types

9.4.1 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.

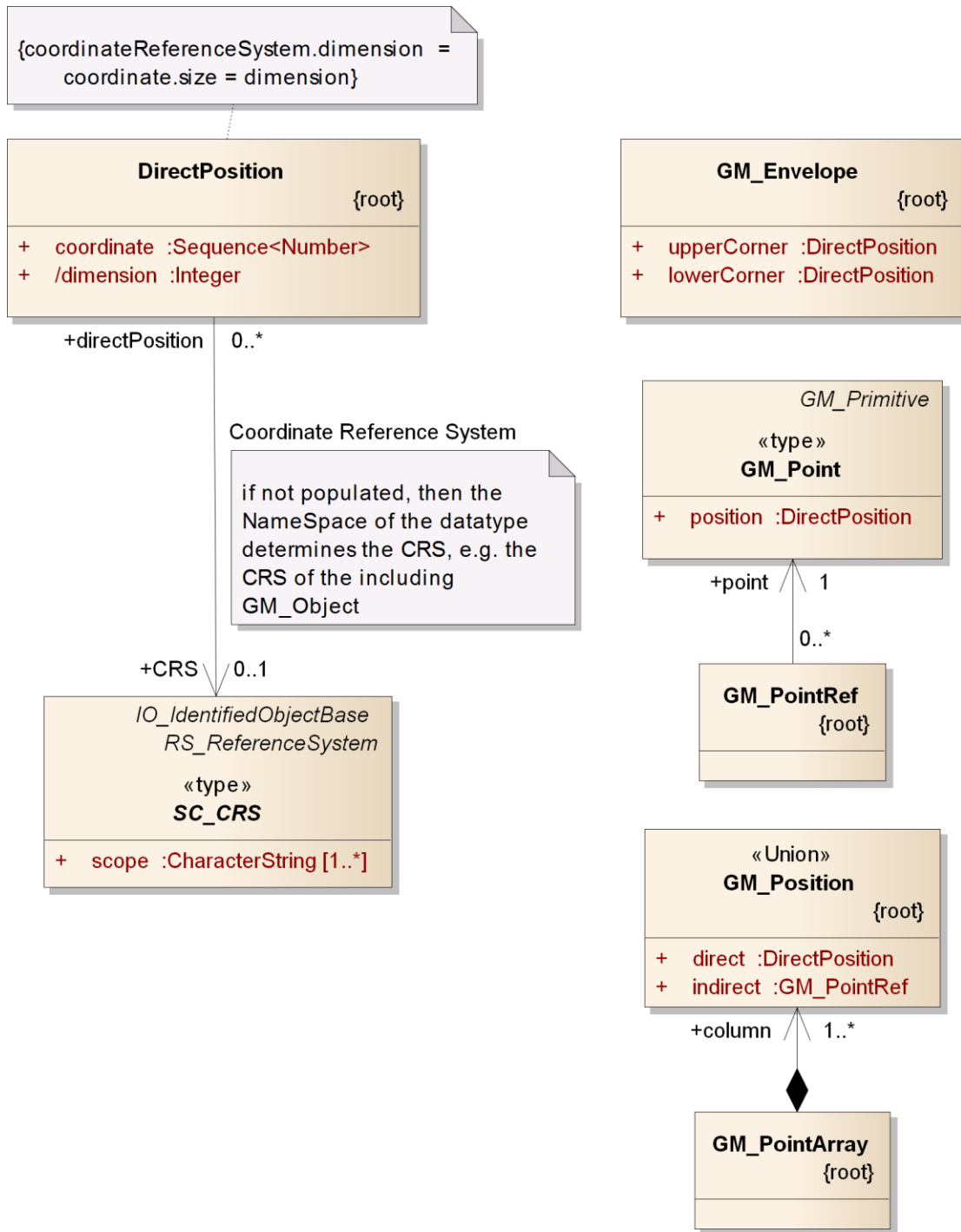


Figure 19: Overview of basic types from ISO 19107 for the definition of position information

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_Point` is the parent of the AIXM Point type (see section 9.3.1). `GM_PointRef` is used to include (the position of) an existing point by reference (in XML usually via `xlinks`). `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 9.3.1). 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.

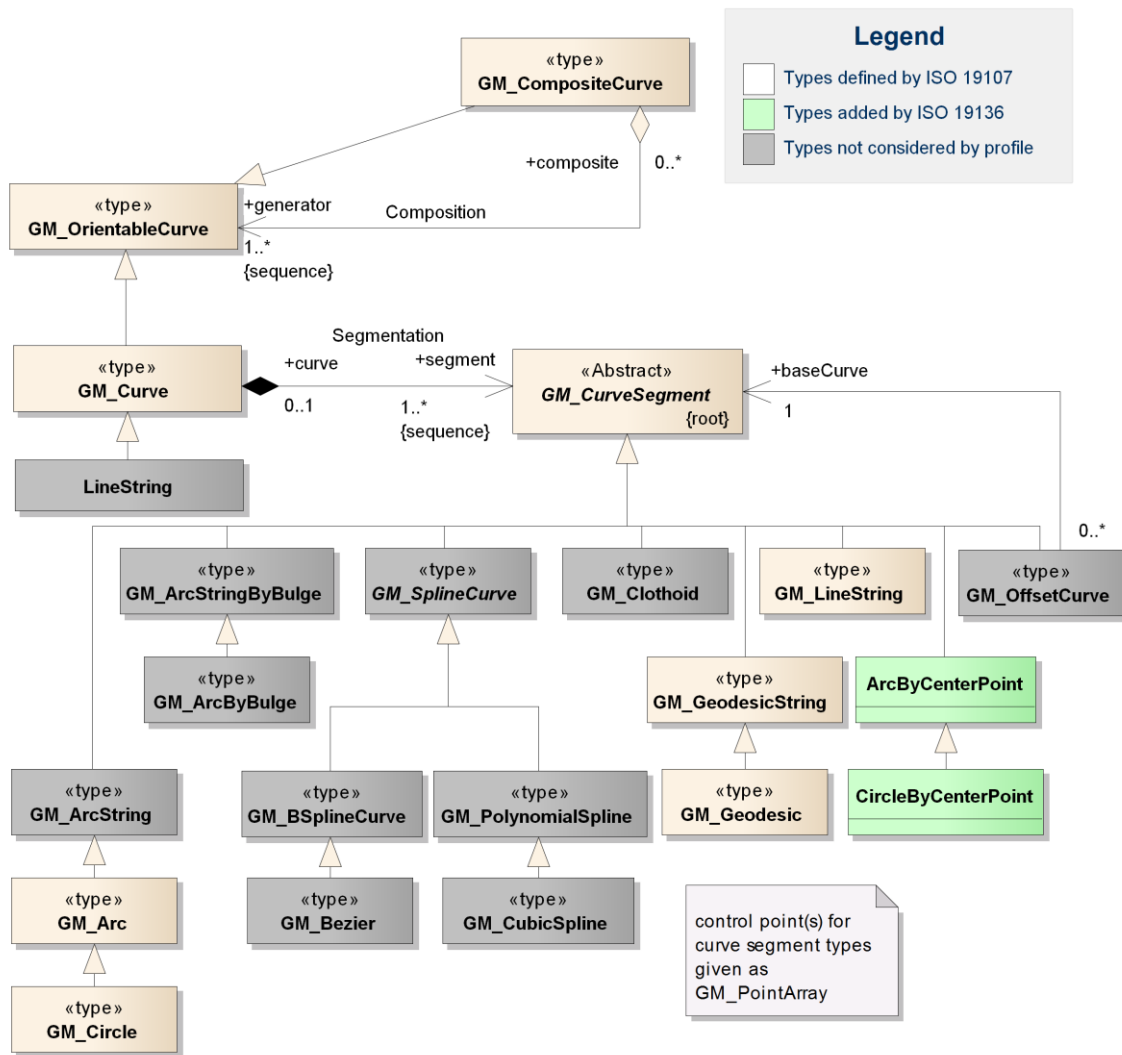


Figure 20: Overview of types from ISO 19107 / ISO 19136 used for the definition of a curve in spatial properties of AIXM features – types irrelevant for the profile marked with grey background

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.

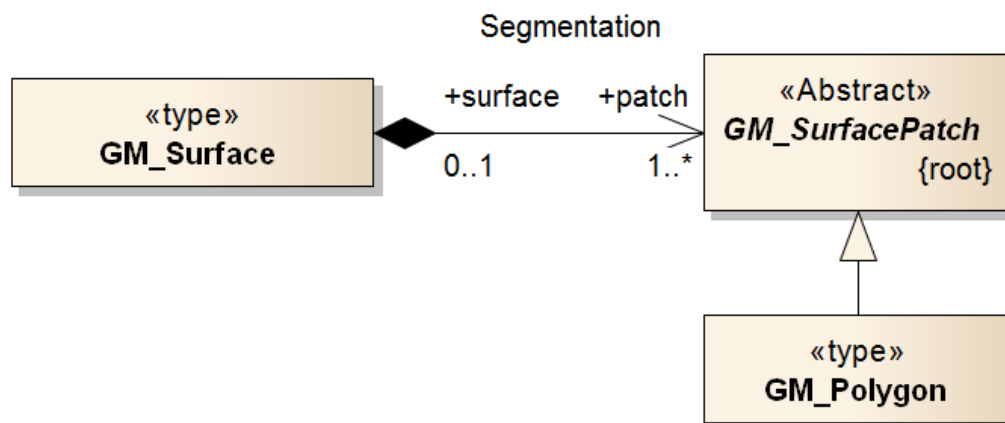


Figure 21: Overview of types from ISO 19107 that are relevant for the definition of a surface in spatial properties of AIXM features

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 20 and the types shown in Figure 21 and Figure 22.

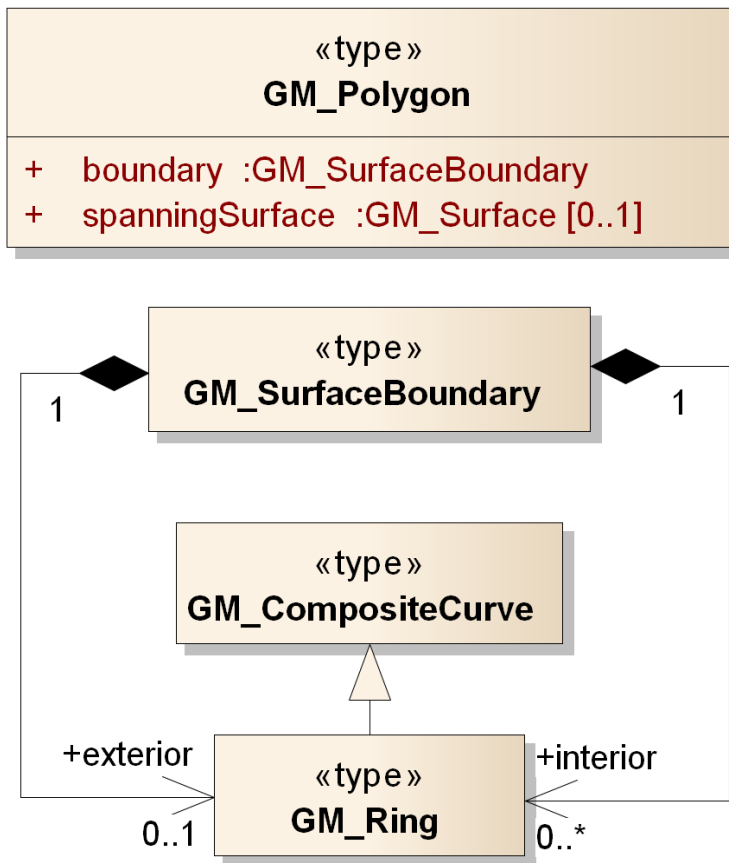


Figure 22: Overview of types from ISO 19107 that are relevant for the definition of a GM_Polygon in spatial properties of AIXM features

Note: the *LinearRing* type (defined by GML) is a child of *GM_Ring* (see section 9.4.22) 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 rhumb lines (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.

The XML Schema implementation of the types from ISO 19107 and ISO 19136 (GML) that are relevant for AIXM is documented in the following sections.

9.4.2 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>none</i>
Usage	Defines the coordinates of a position in a given CRS.
Definition	<p>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.</p> <p>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.</p>
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 (see section 9.4.3), GM_Envelope (see section 9.4.4), GM_Position (see section 9.4.6), GM_LineString (see section 9.4.17), GM_Arc (see section 9.4.13), GM_Circle (see section 9.4.14), ArcByCenterPoint (see section 9.4.11) and CircleByCenterPoint (see section 9.4.12).
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.xsd
XML Schema Component	<pre><complexType name="DirectPositionType"> <simpleContent> <extension base="gml:doubleList"> <attributeGroup ref="gml:SRSReferenceGroup"/> </extension> </simpleContent> </complexType> <element name="pos" type="gml:DirectPositionType"/></pre>
Example	<gml:pos>46.1 3.2</gml:pos>

→ back to *AIXM GML Profile Types tables*

9.4.3 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) – see section 7.2.2 for further details.
Used in	Used as parent type for AIXM Point (see section 9.3.2). Also used in a number of geometry types.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.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"/> </pre>

	<pre> <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>

→ back to *AIXM GML Profile Types tables*

9.4.4 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 (see section for 3.2 further details).
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	<code>gml:boundedBy</code> property of AIXM feature or feature collection
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.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>

→ back to *AIXM GML Profile Types tables*

9.4.5 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	<code>gml:pointProperty</code>
Type	<code>gml:PointPropertyType</code>
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. pointProperty 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 gml:Point.</p>
Comments	The GML implementation of GM_PointRef supports not only the pure reference to a point (see section 9.4.3) but also including a point directly.
Used in	Used in the definition of various geometry types.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.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>

→ back to *AIXM GML Profile Types tables*

9.4.6 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	gml:geometricPositionGroup
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 GM_Point.
Definition	<p>GML supports two different ways to specify a geometric position: either by a direct position (a data type) or a point (a geometric object).</p> <ul style="list-style-type: none"> • <i>gml:pos</i> elements are positions that are —owned by the geometric primitive encapsulating this geometric position. • <i>gml:pointProperty</i> 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 GM_GeodesicString.
Used in	The XML Schema Implementation of the GM_GeodesicString type (see section 9.4.15).
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.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="#P0XYZ"/></pre>

	<pre><!-- end of group --> </gml:GeodesicString></pre>
--	--

→ back to *AIXM GML Profile Types tables*

9.4.7 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	gml:posList
Type	<i>gml:DirectPositionListType</i>
BaseType	<i>none</i>
Restriction	<i>none</i>
Usage	To encode a sequence of direct positions.
Definition	<p><i>gml:posList</i> instances (and other instances with the content model specified by <i>DirectPositionListType</i>) hold the coordinates for a sequence of direct positions within the same coordinate reference system (CRS).</p> <p>If no <i>srsName</i> 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 <i>srsName</i> will be specified at the direct position level only in rare cases.</p> <p>The optional attribute <i>count</i> specifies the number of direct positions in the list. If the attribute <i>count</i> is present then the attribute <i>srsDimension</i> shall be present, too.</p> <p>The number of entries in the list is equal to the product of the</p>

	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.
Comments	<i>none</i>
Used in	Used in the XML Schema Implementation of GM_LineString (see section 9.4.17), GM_Arc (see section 9.4.13), GM_Circle (see section 9.4.14), ArcByCenterPoint (see section 9.4.11), CircleByCenterPoint (see section 9.4.12), GM_GeodesicString (see section 9.4.15) and GM_Geodesic (see section 9.4.16).
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.xsd
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>

→ back to *AIXM GML Profile Types tables*

9.4.8 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	gml:AbstractCurveType
BaseType	gml:AbstractGeometricPrimitiveType
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 (see section 9.4.9), gml:OrientableCurve (see section 9.4.23) and gml:CompositeCurve (see section 9.4.24). Also used in the content model of gml:OrientableCurve, gml:CompositeCurve and gml:Ring (see section 9.4.22). Consequently, the GML encoding is not a direct mapping of the UML model shown in Figure 20.
XML Schema File	(./ISO_19136_Schemas/) geometryBasic0d1d.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>

→ back to *AIXM GML Profile Types tables*

9.4.9 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 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 <i>OrientableCurve</i> as wrapper – see section 9.4.23 for further details.
Used in	Used as parent type for AIXM Curve (see section 9.3.4). Can be used in the definition of a surface boundary (see Figure 22 and section 9.4.22).
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
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>

	<pre> </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>

→ back to *AIXM GML Profile Types tables*

9.4.10 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, as depicted in Figure 20.

XSD Element	gml:AbstractCurveSegment
Type	gml:AbstractCurveSegmentType
BaseType	-

Restriction	<i>none</i>
Usage	Abstract type that is the supertype for all curve segment types, see Figure 20.
Definition	<p>A curve segment defines a homogeneous segment of a curve.</p> <p>The attributes numDerivativesAtStart, numDerivativesAtEnd and numDerivativesInterior specify the type of continuity as specified in ISO 19107:2003, 6.4.9.3.</p> <p>The AbstractCurveSegment 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 interpolation with type gml:CurveInterpolationType 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 – see section 9.4.9.
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>

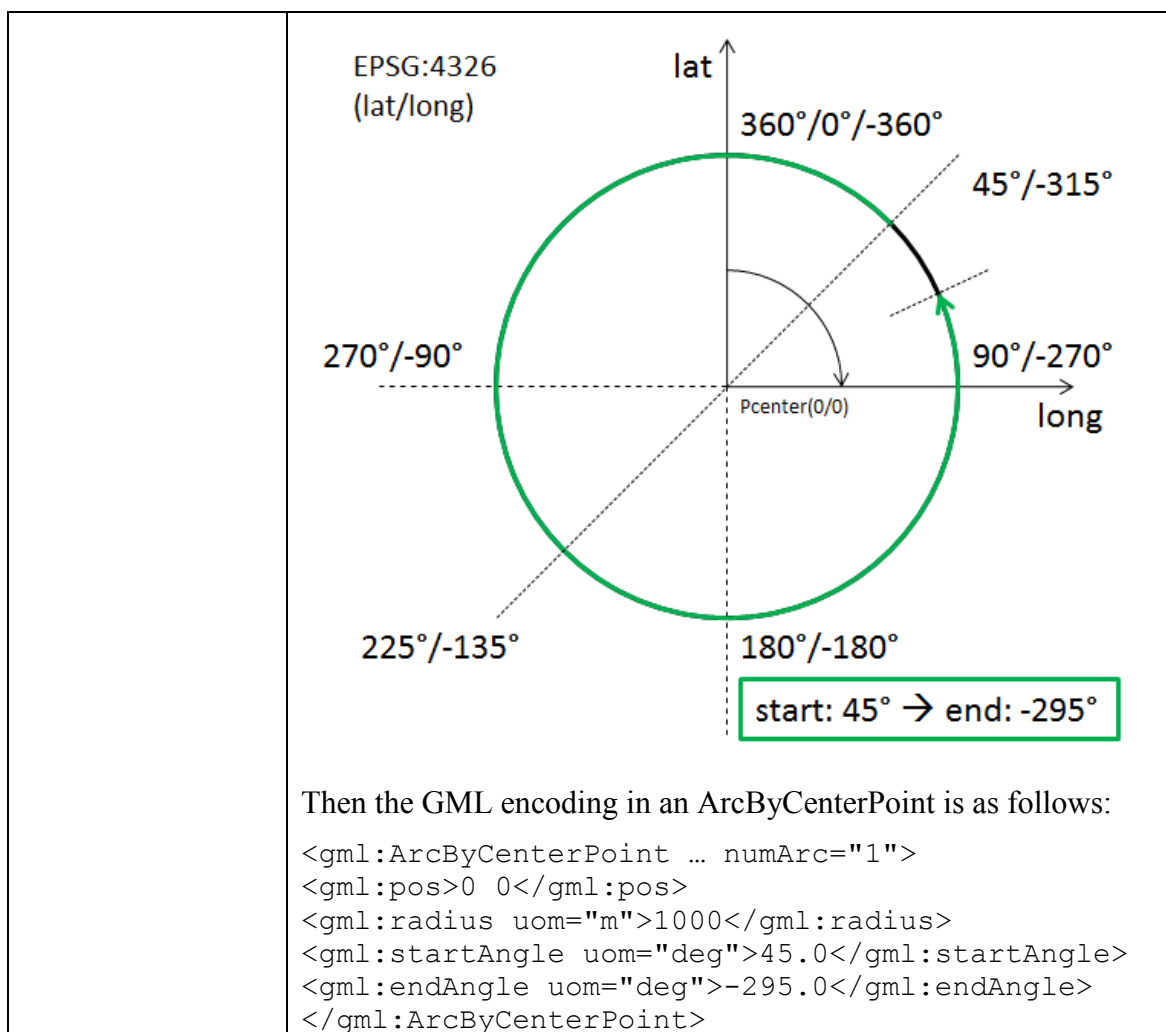
→ back to *AIXM GML Profile Types tables*

9.4.11 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 (see section 9.4.10)
Restriction	<p>In order to comply with common aviation rules (angles are measured clockwise starting from the True North), ArcByCenterPoint should be used only in combination with a CRS that has latitude/longitude axis order – see section 5.2.4.1 for further details.</p> <p>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.</p>
Usage	Typical construct for arcs used in the definition of airspace borders in the AI domain.
Definition	<p>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.</p> <p>The element radius specifies the radius of the arc.</p> <p>The element startAngle specifies the bearing of the arc at the start.</p> <p>The element endAngle specifies the bearing of the arc at the end.</p> <p>The interpolation is fixed as "circularArcCenterPointWithRadius".</p> <p>Since this type describes always a single arc, the attribute "numArc" is fixed to "1".</p> <p>The content model follows the general pattern for the encoding of curve segments.</p>

Comments	See section 5.2.4 for further details about the angle measurement convention, angle value ranges, arc direction, arc interpolation, mapping rules as well as recommended units of measurements.
Used in	As child of GM_CurveSegment (see section 9.4.10) to represent a segment of a GM_Curve (see section 9.4.9).
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"/> <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	Assuming an arc defined by its center point as well as start and end angle as shown in the following diagram:



→ back to *AIXM GML Profile Types tables*

9.4.12 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	<code>gml:ArcByCenterPointType</code> (see section 9.4.11)
Restriction	<p>Only the circle center position as well as the radius can be defined.</p> <p>The use of the child elements “<code>gml:pointRep</code>” as well as “<code>gml:coordinates</code>” is deprecated. The child element “<code>gml:posList</code>” is allowed by ISO 19136, but if used it shall contain only one position that represents the center of the circle.</p>
Usage	To define the geometry of a circular airspace in the AI domain, see section 5.2.5 for further details.
Definition	A <code>gml:CircleByCenterPoint</code> is a <code>gml:ArcByCenterPoint</code> with identical start and end angle to form a full circle. Again, this representation can be used only in 2D.
Comments	<p>Use of <code>CircleByCenterPoint</code> 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, see section 5.2.7).</p> <p>If <code>gml:CircleByCenterPoint</code> was used in conjunction with other curve segments to define a curve then the direction (clockwise / counter-clockwise) of the <code>CircleByCenterPoint</code> is not well-defined by GML.</p>
Used in	As child of <code>ArcByCenterPoint</code> (see section 9.4.11) to represent the segment of a <code>GM_Curve</code> (see section 9.4.9) that forms a circle.
XML Schema File	(./ISO_19136_Schemas/) <code>geometryPrimitives.xsd</code>
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> </choice> <element ref="gml:posList"/> </pre>

	<pre> <element ref="gml:coordinates"/> </choice> <element name="radius" type="gml:LengthType"/> </sequence> </restriction> </complexContent> </complexType> </pre>
Example	<pre> <gml:CircleByCenterPoint ... numArc="1"> <gml:pos>0 0</gml:pos> <gml:radius uom="m">1000</gml:radius> </gml:CircleByCenterPoint> </pre>

→ back to *AIXM GML Profile Types tables*

9.4.13 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” – see section 5.2.3. Although not extensively used at present in the aviation domain, it is expected that its usage will increase, with the corresponding diminishing of ArcByCenterPoint (see section 9.4.11). 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 (see section 9.4.9).
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"/> </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>

→ back to *AIXM GML Profile Types tables*

9.4.14 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	<code>gml:Circle</code>
Type	<code>gml:CircleType</code>
BaseType	<code>gml:ArcType</code> (see section 9.4.13)
Restriction	Use of the child elements “ <code>gml:pointRep</code> ” and “ <code>gml:coordinates</code> ” 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 <code>CircleByCenterPoint</code> (see section 9.4.12), <code>Circle</code> has a well defined direction.
Used in	As child of <code>GM_Arc</code> (see section 9.4.13), usually to represent a segment of a <code>GM_Curve</code> (see section 9.4.9) that forms a circle.
XML Schema File	(./ISO_19136_Schemas/) <code>geometryPrimitives.xsd</code>
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>

→ back to *AIXM GML Profile Types tables*

9.4.15 `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 (see section 9.4.10)
Restriction	<i>none</i>
Usage	Used for the encoding of straight lines on the earth surface.
Definition	<p>A sequence of geodesic segments.</p> <p>The number of control points shall be at least two.</p> <p>interpolation is fixed as "geodesic".</p> <p>The content model follows the general pattern for the encoding of curve segments.</p>
Comments	<p>GeodesicString is the default encoding for straight lines recommended by this document, see section 5.2.1 for details.</p> <p>The more compact form of representing the control points of a GeodesicString is achieved via the gml:posList element.</p>
Used in	As child of GM_CurveSegment (see section 9.4.10) to represent a segment of a GM_Curve (see section 9.4.9).
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"/> </extension> </pre>

	<pre></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>

→ back to *AIXM GML Profile Types tables*

9.4.16 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 (see section 9.4.15)
Restriction	<i>none</i>
Usage	To define a straight line on the earth surface – usually to define a curve segment.
Definition	A <i>GM_Geodesic</i> consists of two distinct positions joined by a geodesic curve. The control points of a <i>GM_Geodesic</i> 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” – see section 9.4.15.
Used in	As child of <i>GM_GeodesicString</i> (see section 9.4.15), usually to represent a segment of a <i>GM_Curve</i> (see section 9.4.9).
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>

→ back to *AIXM GML Profile Types tables*

9.4.17 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 (see section 9.4.10)
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	<p>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.</p> <p>The content model follows the general pattern for the encoding of curve segments.</p>
Comments	<p>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. See section 5.2.2 for further details.</p> <p>The more compact form of representing the control points of a GM_LineString is achieved via the gml:posList element.</p>
Used in	As child of GM_CurveSegment (see section 9.4.10) to represent a segment of a GM_Curve (see section 9.4.9).
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> </complexType> </pre>

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

→ back to *AIXM GML Profile Types tables*

9.4.18 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, see section 5.2.7.
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>

	</complexType>
Used in	Used as supertype for AIXM Surface (see section 9.3.6).
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> <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>

→ back to *AIXM GML Profile Types tables*

9.4.19 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 (see section 9.4.20) 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 – see section 9.4.18.
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>

→ back to *AIXM GML Profile Types tables*

9.4.20 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 (see section 9.4.19)
Restriction	<i>none</i>

Usage	Used to represent the patch(es) of an AIXM Surface.
Definition	<p>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.</p> <p>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.</p>
Comments	<p>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, see section 5.2.7.</p> <p>GM_Ring – see section 9.4.22 – should be used to represent and encode the exterior/interior of a GM_Polygon.</p>
Used in	As child of GM_SurfacePatch (see section 9.4.19) to represent the surface patch(es) of a GM_Surface (see section 9.4.18).
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"></pre>

	<pre> <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> </gml:Curve> </gml:curveMember> </gml:Ring> </gml:exterior> </gml:PolygonPatch> </pre>
--	--

→ back to *AIXM GML Profile Types tables*

9.4.21 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 (see section 9.4.22) and its subtypes.
Definition	<p>An abstraction of a ring to support surface boundaries of different complexity.</p> <p>The AbstractRing element is the abstract head of the substitution group for all closed boundaries of a surface patch.</p>
Comments	At the moment, the only substitute for gml:AbstractRing foreseen by this profile is GM_Ring / gml:Ring – see section 9.4.22.
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>

→ back to *AIXM GML Profile Types tables*

9.4.22 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	gml:RingType
BaseType	gml:AbstractRingType (see section 9.4.21)
Restriction	<i>none</i>
Usage	Used to represent the exterior/interior of a surface boundary (see Figure 22).
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 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.</p> <p>If provided, the aggregationType attribute shall have the value "sequence".</p>
Comments	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

	<p>Ring is formed by a GeodesicString, then the first and last position element of that GeodesicString must be equal in order to form a cycle. The CircleByCenterPoint type (see section 9.4.12) 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 GeoBorders 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 aixm:Curve as curve member, the latter requires either a local reference to a curve or an abstract reference to a remote feature. For further details, see section 8.</p>
Used in	The definition of the exterior/interior(s) of a Polygon (see section 9.4.20).
XML Schema File	(/ISO_19136_Schemas/) geometryPrimitives.xsd
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 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></pre>

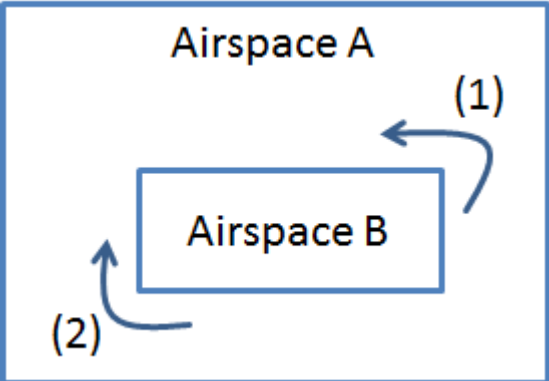
	<pre> </gml:curveMember> <gml:curveMember xlink:href="#CRV002" xlink:title="along the state border with Islamic Republic of Iran"/> <gml:curveMember xlink:href="#CRV003" xlink:title="along the state border with Armenia"/> </gml:Ring> </pre>
--	--

→ back to *AIXM GML Profile Types tables*

9.4.23 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</p>

	base curve.
Comments	<p>OrientableCurve can be used to invert the direction of a curve inside a gml:Ring (see section 9.4.22). OrientableCurve is not foreseen to be used often. However, there is one use case that would require OrientableCurve:</p> <p>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 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 (see section 5.2.7). 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 (see section 9.4.22), CompositeCurve (see section 9.4.24) or another OrientableCurve.
XML Schema File	(./ISO_19136_Schemas/) geometryPrimitives.xsd
XML Schema Component	<element name="OrientableCurve" type="gml:OrientableCurveType" substitutionGroup="gml:AbstractCurve"/>

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

→ back to *AIXM GML Profile Types tables*

9.4.24 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
BaseType	gml:AbstractCurveType
Restriction	<i>none</i>
Usage	To represent a curve as a combination of other curves.
Definition	A <i>gml:CompositeCurve</i> is represented by a sequence of (orientable) curves such that each curve in the sequence terminates

	<p>at the start point of the subsequent curve in the list.</p> <p><code>gml:curveMember</code> 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 <code>aggregationType</code> attribute shall have the value "sequence".</p> <p>NOTE This definition allows for a nested structure, i.e. a <code>gml:CompositeCurve</code> may use, for example, another <code>gml:CompositeCurve</code> as a curve member.</p>
Comments	<p><code>CompositeCurve</code> supports a simple aggregation of curves. This can be used to combine existing curves by reference (<code>xlink:href</code>), for example to ensure consistency in use of common boundaries (or segments thereof).</p>
Used in	<p>As member of a Ring (see section 9.4.22), <code>OrientableCurve</code> (see section 9.4.23) or <code>CompositeCurve</code>.</p>
XML Schema File	<p>(./ISO_19136_Schemas/) <code>geometryPrimitives.xsd</code></p>
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>
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>

→ back to *AIXM GML Profile Types tables*

10 Interpolation and Densification Considerations

10.1 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.

10.1.1 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.

10.2 Mapping of GML to Simple Geometry

10.2.1 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.

10.2.2 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.

ISO curve segment type	CRS type	Transformation	Simple representation
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 ⁴	Any	Densified	LineString *
GML CircleByCenterPoint ⁵	Any	Densified	LineString *
GM_Geodesic	Geodetic	Direct	Loaded as a linestring
GM_Geodesic	Non-geodetic	Densified	Densified

⁴ defined by GML only, not by ISO 19107

⁵ defined by GML only, not by ISO 19107

GM_GeodesicString	Geodetic	Direct	Loaded as a linestring **
GM_GeodesicString	Non-geodetic	Densified	LineString

* Arcs and circles are interpreted as being the set of points at an equal 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

** In a geodetic coordinate system a linestring and a geodesic between two control points follow the same path.

*** 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.

10.2.3 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.

Annex A - ArcByCenterPoint Interpretation Summary

Introduction

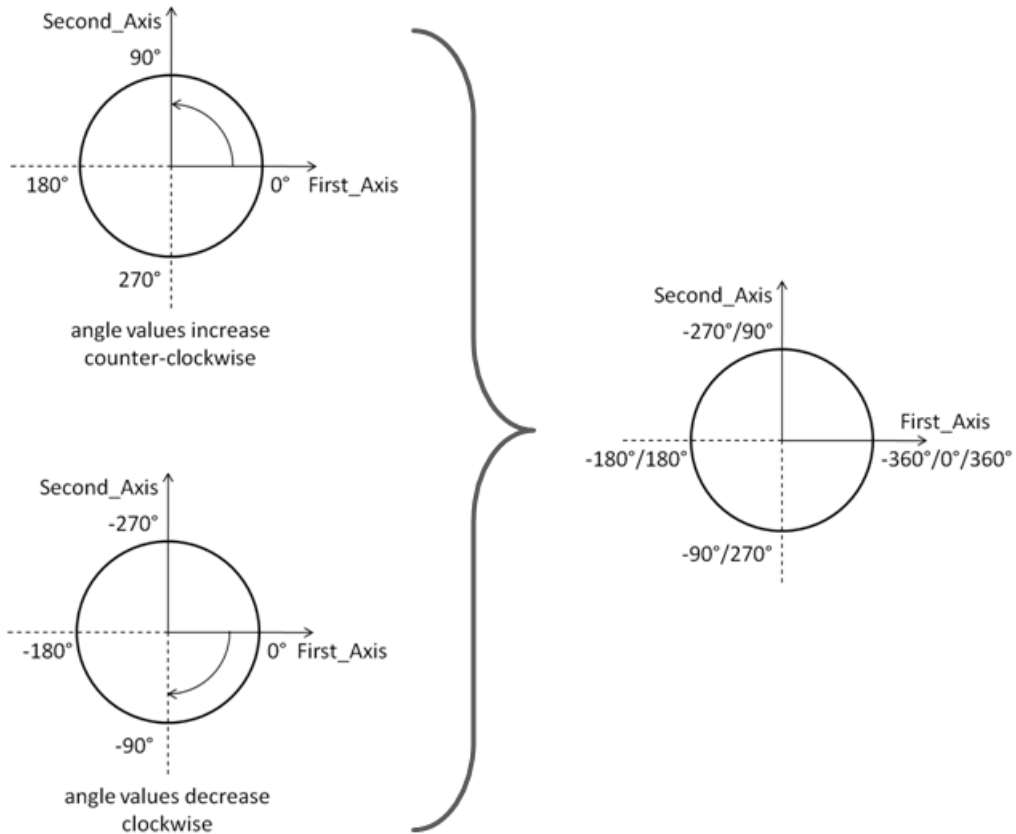
This annex contains the summary of the discussions on the correct interpretation of ArcByCenterPoint.

Angle Measuring Convention in GML

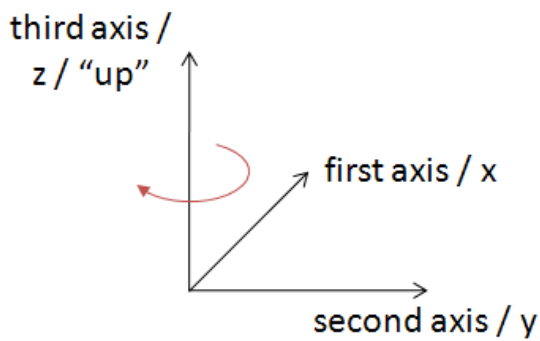
GML implements the semantics of ISO 19107. As such, the semantics for an angle encoded in GML is given by ISO 19107 clause 6.3.12.2: *"In this variant of Bearing usually used for 2D coordinate systems, the first angle (azimuth) is measured from the first coordinate axis (usually north) in a counterclockwise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

General Direction of Increasing and Decreasing Angle Values

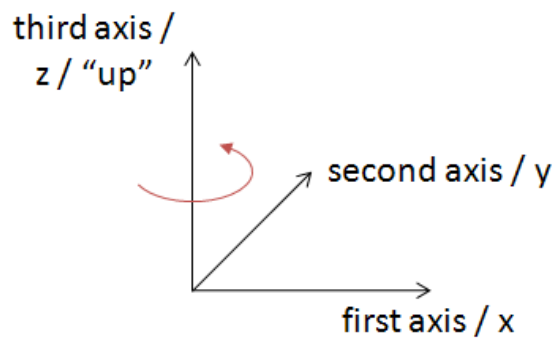
ISO 19107 thus defines in which direction angle values increase and in which direction they decrease - see the following diagram:



Depending upon the "up" and the orientation of the first two axes, we have a left-handed or right handed coordinate system (see wikipedia and the following figure).



left-handed system



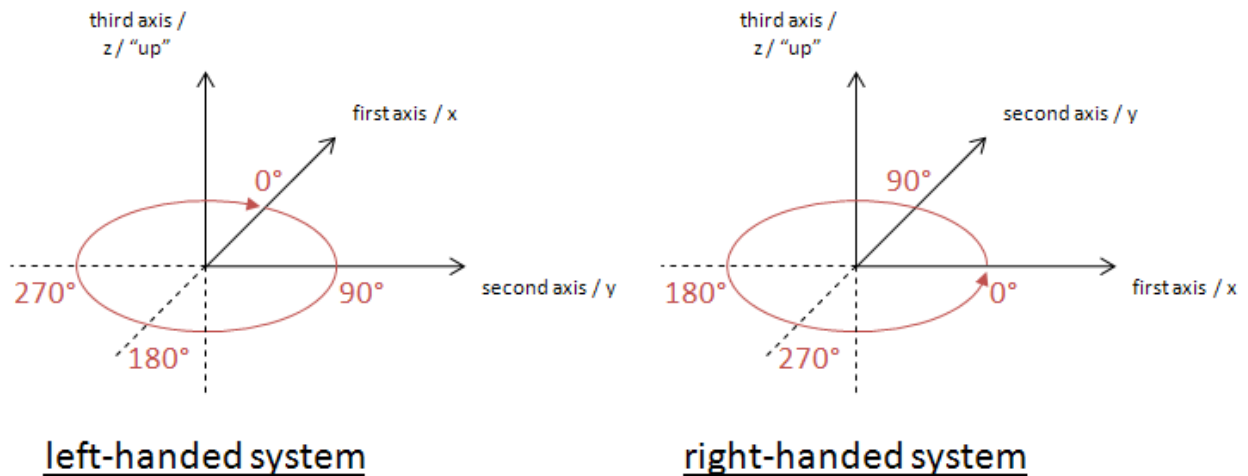
right-handed system

(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.

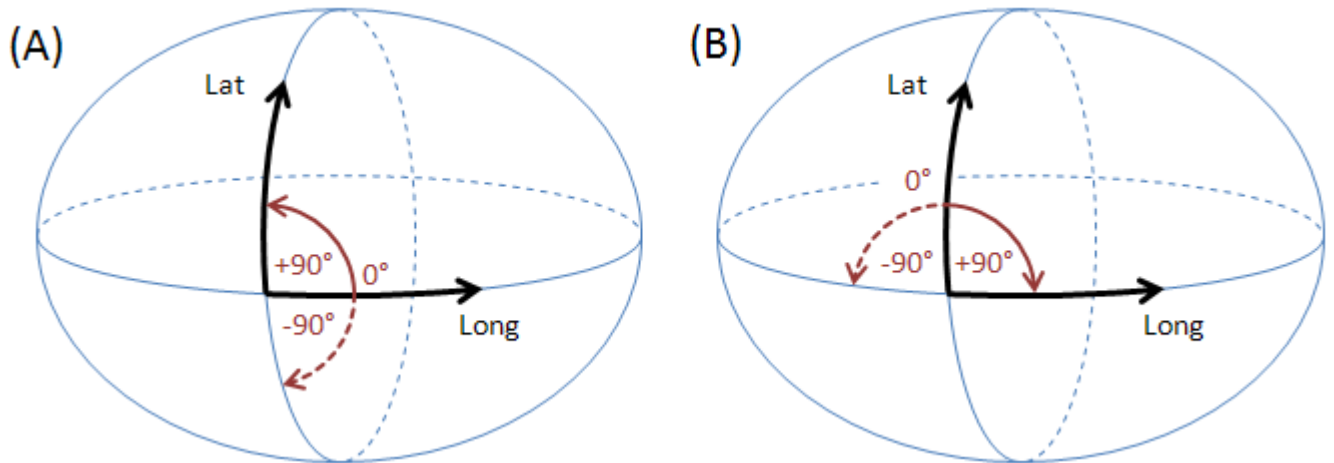


Note: The planes in geographic information are representations of the ground, and as such have a "natural" up direction (usually represented by a normal vector). In a 3D system, "up" is given by the definition of the third axis in the chosen CRS. In a 2D system "up" is usually implied.

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

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

(B) GeodeticCRS: urn:ogc:def:crs:EPSG::4326

- 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 in both cases. 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.

Examples

The following diagrams exemplify this definition. 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)

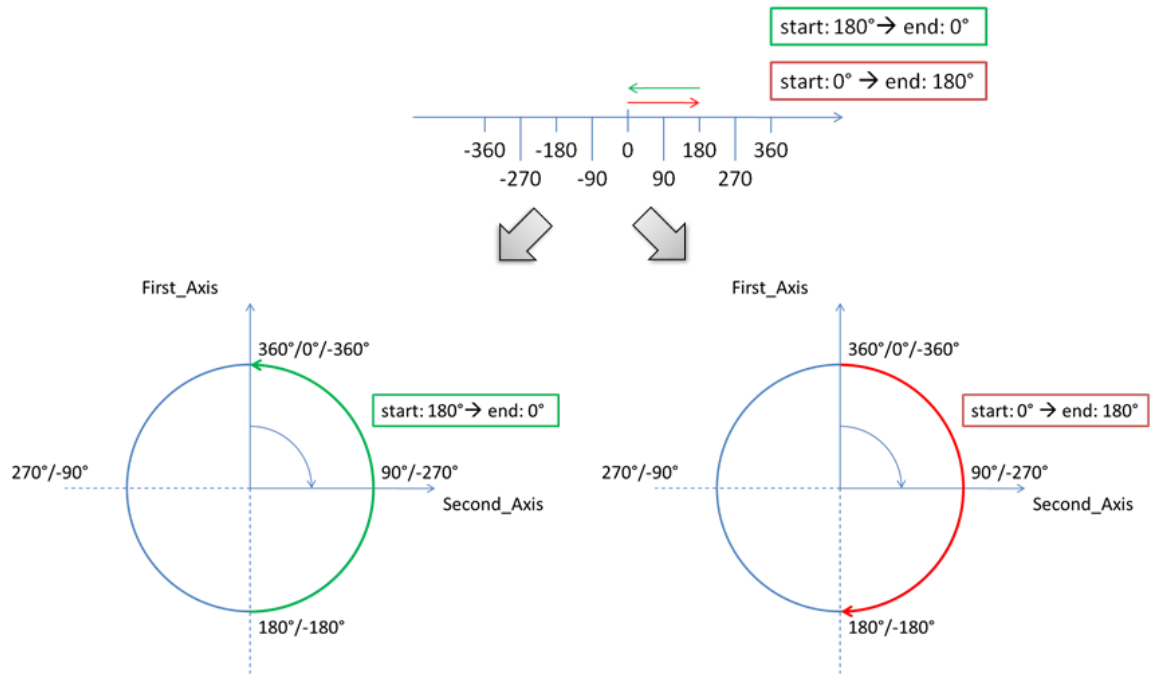
- computing the length of an arc can be performed as follows: $|startAngle - endAngle|/180^\circ * radius * pi$ - the definition conforms to this formula

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

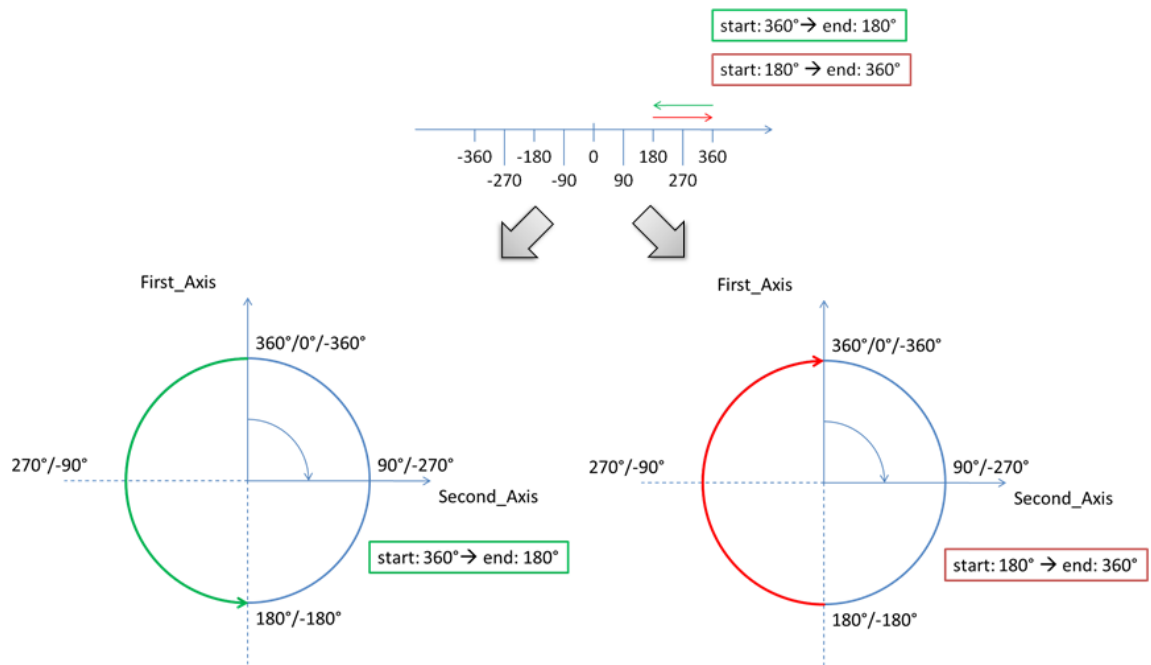
```
<gml:ArcByCenterPoint numArc="1"
xsi:schemaLocation="http://www.opengis.net/gml/3.2
http://schemas.opengis.net/gml/3.2.1/gml.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml/3.2">
  <gml:pointProperty>
    <gml:Point gml:id="ID1">
      <gml:pos>0 0</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.

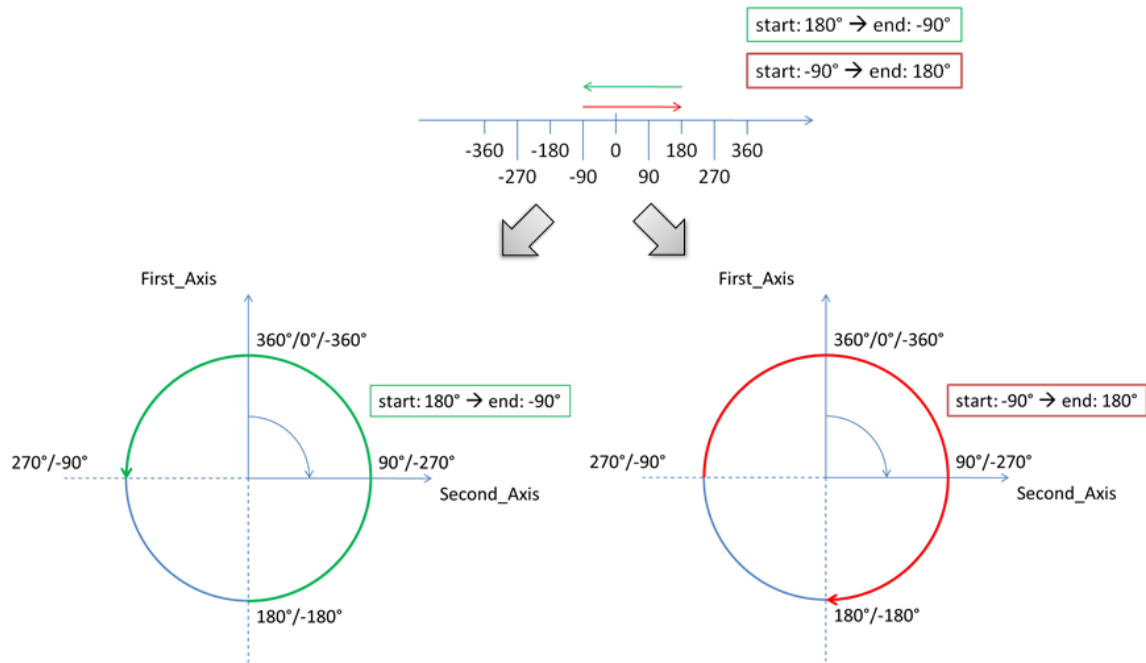
example - one



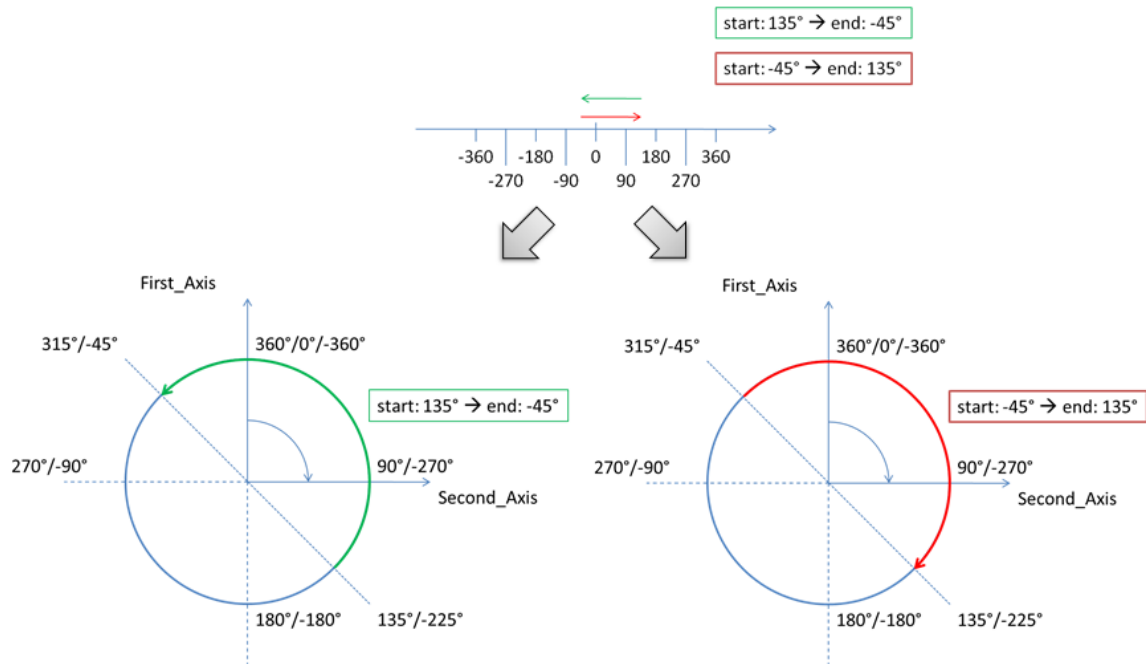
example - two



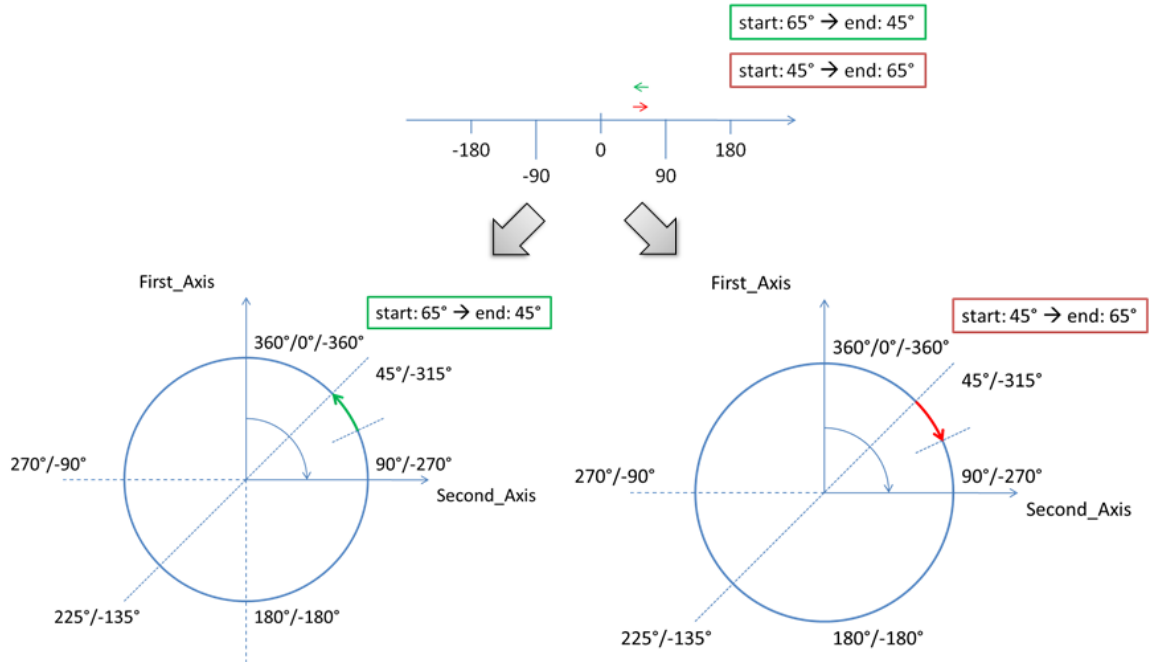
example - three



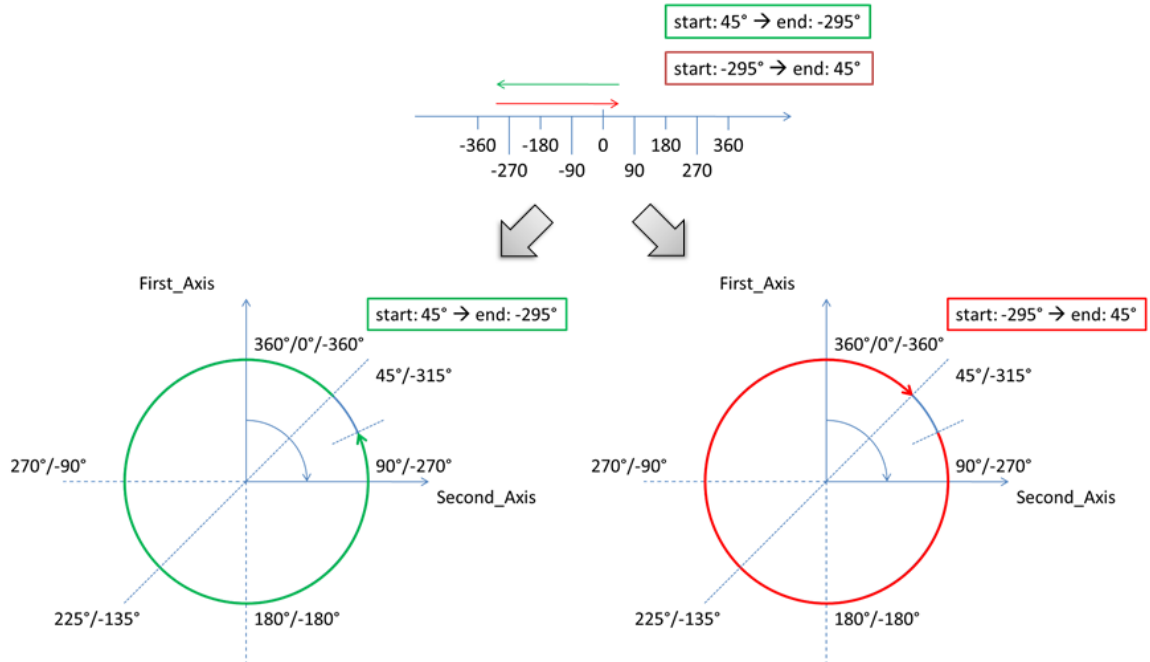
example - four



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 center 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 B - 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 23 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.

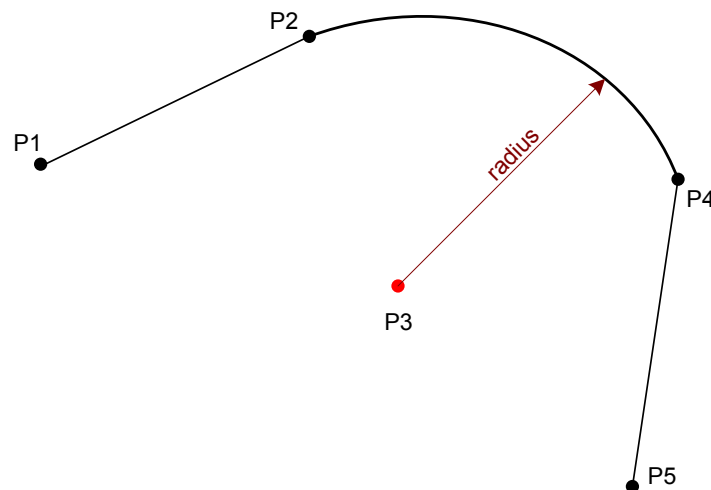


Figure 23 - Mapping arc by start/end point into ArcByCenterPoint

Sometimes, trying to draw the arc using exactly the 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 different 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 **Error! Reference source not found.** 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.

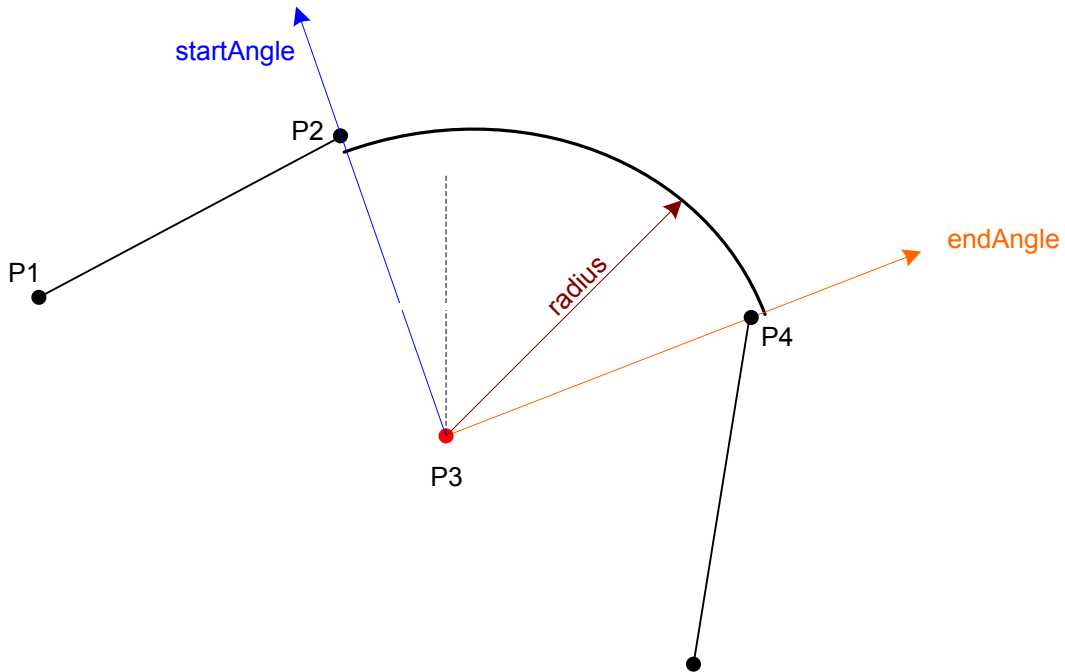


Figure 24 - Calculating startAngle and endAngle for ArcByCenterPoint

If the potential misalignment of the arc (as indicated in **Error! Reference source not found.**) 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 **Error! Reference source not found.**

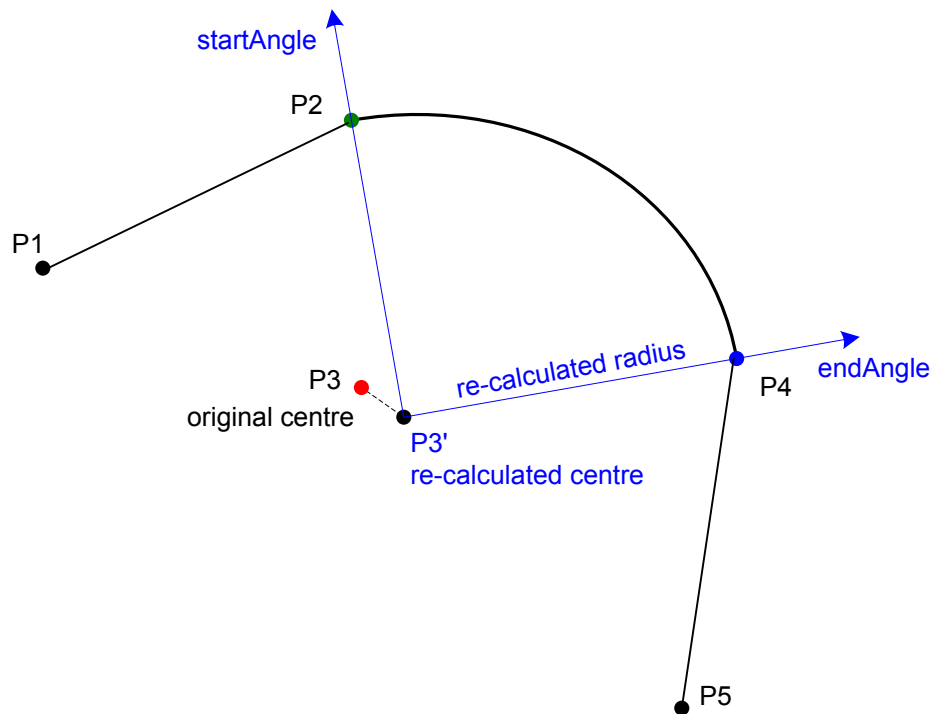


Figure 25 – Re-calculating the whole arc so that it generates a closed surface

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 is left for decision by application developers.

Annex C – GML change request – support arbitrary rhumb lines

The following change request has been raised within OGC in order to enable specifying additional interpolations, such as rhumb line.

Title:	⌘ GML: Make CurveInterpolationType and SurfaceInterpolationType expandable codelists.
Source:	⌘
Work item code:	⌘ Date: ⌘ 2008-11-11
Category:	⌘ B
	<p>Use <u>one</u> of the following categories:</p> <p>F (Critical correction) A (corresponds to a correction in an earlier release) B (Addition of feature), C (Functional modification of feature) D (Editorial modification)</p> <p>Detailed explanations of the above categories can be found in the TC Policies and Procedures.</p>
Reason for change:	⌘ (1.) Other curve interpolation types needed in applications (e.g. rhumb lines in Aeronautical Information (AIXM), often used in (aerial and naval) navigation). (2.) ISO 19107 GM_CurveInterpolation and GM_SurfaceInterpolation are <<CodeList>>s that are expandable per definition.
Summary of change:	⌘ Change the definition of CurveInterpolationType (10.4.7.3) and SurfaceInterpolationType (10.5.12.3) such that both become extendable codelists in either of the two ways specified in E.2.4.9 to allow the use of other values than the predefined ones. This also brings the implementation in line with ISO 19107, where both types are <<CodeList>>s, which are supposed to be implemented as expandable types.

Annex D – 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 it for granted that such a kind of structure exists. This is the case for a “flat” surface, isomorphic to \mathbb{R}^2 , because \mathbb{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**⁶: 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

⁶ a linear structure is locally implicitly defined by the coordinate system itself, namely by the local mapping of the surface on \mathbb{R}^2

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
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 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 broadly vary, 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 troubles 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 treat geodesic objects as linear in many respects (for computation and visualization purposes). One of the most used tricks to reduce approximation errors of extended objects is their “densification”. Densification has anyway its drawbacks:

- the original geometric data (the extended curve) is altered, with **integrity loss**;

- the subsampling introduces a given amount of **arbitriness and irreversibility** (from the subsampled geometry it is not possible to recover the original data);
- the approximated representation implies anyway 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.

Annex E – Offset Curve and Airspace Corridor encoding issues

Introduction

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.

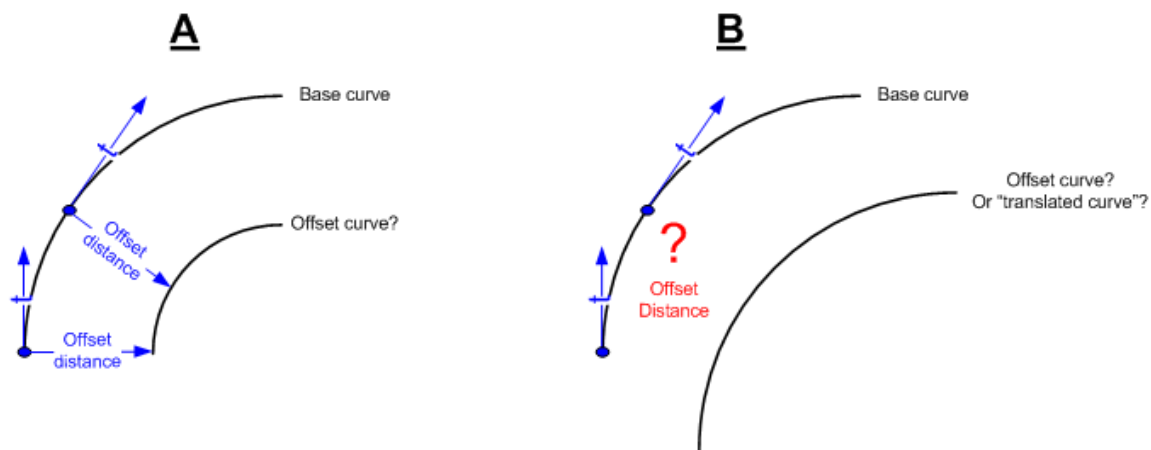


Figure 26 – Example “A” is the correct interpretation of an offsetCurve

The second issue is constructing an offset curve when the base curve is composed of segments that sudden changes of 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 27.

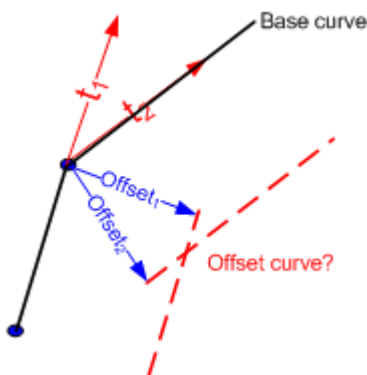


Figure 27 - What about sudden changes in the tangent direction?

In fact, the encoding of an airspace corridor requires the use of two `gml:OffsetCurve` elements as shown in Figure 28. 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 $P1'$, $P2''$, $P1'''$ need to be calculated in order to define the closing arc, and similar for the points $P2'$, $P2''$, $P2'''$.

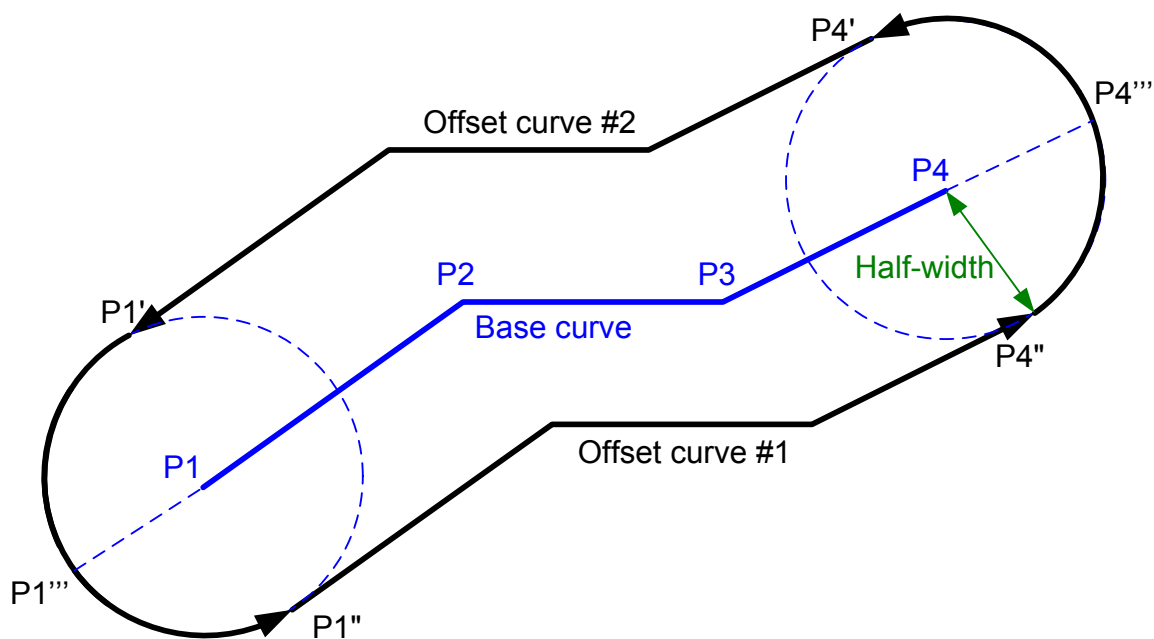


Figure 28 - Airspace corridor encoding

What we actually need is a “buffer” around the base curve, as shown in Figure 29. This does not seem to exist in GML as a specific construct.

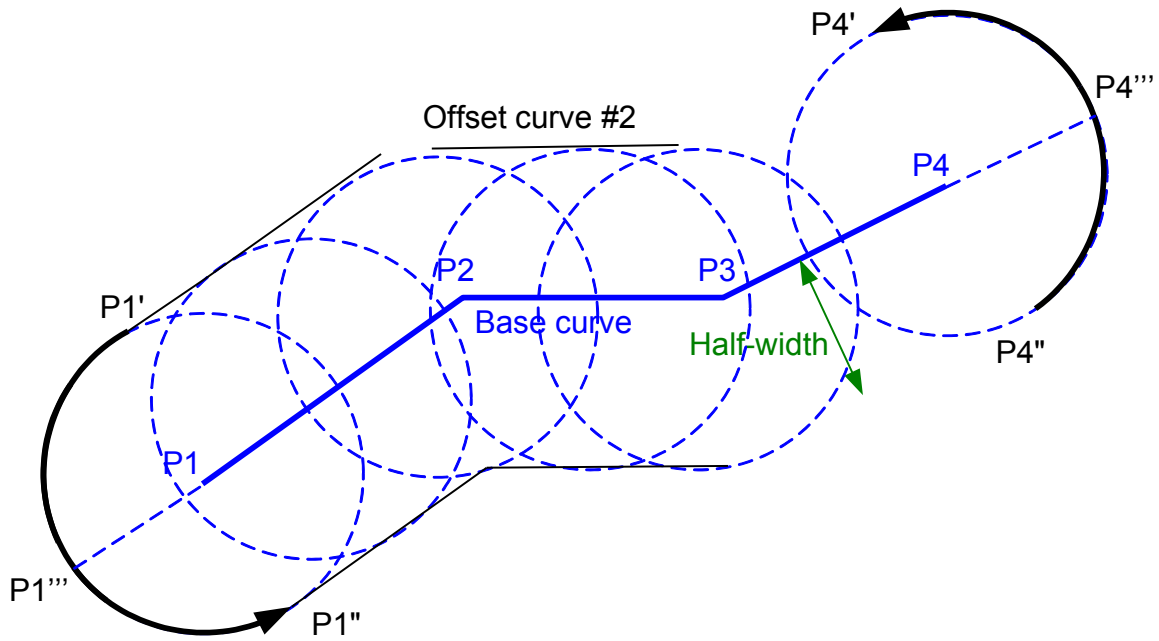


Figure 29 - Airspace corridor as buffer around a centreline