S-100 – Part 7

Spatial Schema

Page intentionally left blank

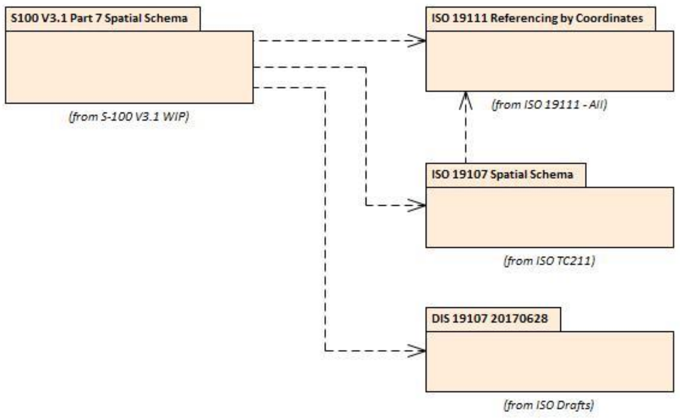
Contents

7-1 Scope

The spatial requirements of S-100 are less comprehensive than the requirements of ISO 19107 “Geographical Information - Spatial schema” which contains all the information necessary for describing and manipulating the spatial characteristics of geographical features and on which this Part is based. Hence this Part contains only the subset of ISO 19107 classes required for S-100. This version only contains geometry, if there is a future requirement for topology then this Part will be extended to meet these requirements.

This Part specifies:

1) A subset of ISO 19107 classes (clause 6) which is the minimum required to support 0, 1, 2 and 2.5 dimensional spatial schemas. As such it is restricted to specifying only data and does not include operations; and
2) Additional constraints (omitted optional elements or constrained cardinalities) which are imposed on these classes by this profile.
3) Additional classes for certain kinds of curvilinear geometry. These additional classes are based on specifications that are expected to be in the next edition of ISO 19107.



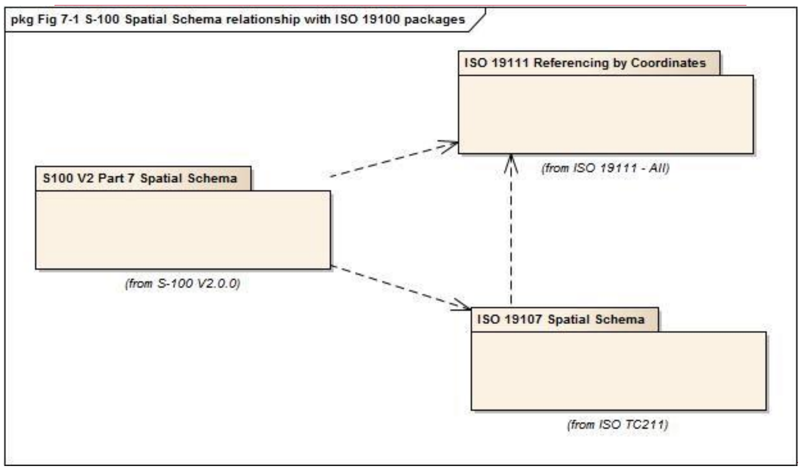

pkg Fig 7-1 S-100 Spatial Schema relationship with ISO 19100 packages
ISO 19111 Referencing by Coordinates
(from ISO 19111 - All)
S100 V2 Part 7 Spatial Schema
(from S-100 V2.0.0)
ISO 19107 Spatial Schema
(from ISO TC211)

Figure 7-1 — S-100 Spatial Schema relationship with ISO 19100 Packages

7-2 Conformance

This profile consists of simple geometry based on three criteria – complexity, dimensionality and functional complexity. The first two criteria (complexity and dimensionality) determine the types defined in this profile that shall be implemented according to an application schema that conforms to a given conformance option.

There are:

Two levels of complexity:

1) Geometric Primitives
2) Geometric Complexes;

Four levels of dimensionality:

1) 0-dimensional objects
2) 0- and 1-dimensional objects
3) 0-, 1- and 2-dimensional objects
4) 0-, 1-, 2- and 2.5 -dimensional objects; and

One level of functional complexity:

1) Data types only (operations are not included).

This profile satisfies the conformance classes A.1.1.1, A.1.1.2, A.1.1.3, A.2.1.1 and A.2.1.2 in ISO 19107. This profile conforms to conformance class 2 of ISO 19106:2004.

7-3 Normative references

The following referenced documents are required for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including amendments) applies.

ISO 19107:2003, Geographic information — Spatial schema.

ISO TS 19103:2005, Geographic information — Conceptual schema language.

ISO 19111, Geographic information — Spatial referencing by coordinates.

7-3.1 Non-normative references

ISO/DIS 19107, Geographic information – Spatial schema (2017-06-28)

Formatted: Heading 2, Left, Space After: 0 pt

7-4 Geometry

7-4.1 Introduction

This profile consists of simple geometry which can be expressed in multiple configurations as described in ISO 19107:2003 clause 6.1.3.

7-4.1.1 S-100 spatial schema Geometry classes and their ISO 19107:2003 reference

Table 7-1 Spatial types

Coordinate Geometry	Geometry Primitive	Geometry Complex	Geometry Aggregate
DirectPosition (6.4.1)	GM_Curve (6.3.16)	GM_Complex (6.6.2)	GM_Aggregate (6.5.2)
CurveInterpolation (6.4.8)	GM_CurveBoundary (6.3.5)	GM_Composite (6.6.3)	GM_MultiPoint (6.5.4)
GM_CurveSegment (6.4.9)	GM_OrientableCurve (6.3.14)	GM_CompositeCurve (6.6.5)	
GM_Position (6.4.5)	GM_OrientableSurface (6.3.15)		
GM_Polygon (6.4.36)	GM_Point (6.3.11)		
GM_SurfacePatch (6.4.34)	GM_Primitive (6.3.10)		
SurfaceInterpolation (6.4.32)	GM_Ring (6.3.6)		
S100_ArcByCenterPoint (none)	GM_Surface (6.3.17)		
S100_CircleByCenterPoint (none)	GM_SurfaceBoundary (6.3.7)		
S100_GM_SplineCurve			
S100_GM_PolynomialSpline			

Formatted: Left

Formatted: Left

7-4.1.1.1 Splines model (Informative)

Formatted: Heading 4

The spline classes S100_GM_SplineCurve and S100_GM_PolynomialSpline in this revision of S-100 bridge the curves and splines model in ISO 19107:2003 and the draft revision of ISO 19107, which is under development as this update to S-100 is being developed. Considerations in this bridging are:

- The new draft ISO model removes the concept of curve segment; "…Curve, CurveSegment, GenericCurve and CompositeCurve [are] implemented by a single class. For the same reason, there are no separate 'segments' or patches." A strict integration of this concept into S-100 would require a comprehensive overhaul of Part 7, and potentially of the S-100 data formats as well.
- The model in ISO 19107:2003 is flawed in its modelling of knots, and this propagated into the GML schema in ISO 19136.
- Finalization of the new edition of ISO 19107 is still some time away – the current draft is not yet an ISO International Standard and is subject to change.
- Some spline classes (or interfaces) merely add constraints and/or change a fixed attribute value compared to their generalizations, without defining any new attributes.

Formatted: English (United Kingdom)

The cross-references for the S100 spline classes to ISO 19107:2003 and the draft ISO 19107 classes as of August 2017 are given in the table below:

Table 7-2 ISO references for S-100 spline classes

S-100 class	ISO 19107:2003 reference	Draft ISO 19107 model reference

S100_GM_SplineCurve	GM_SplineCurve (6.4.26); GM_BSplineCurve (6.4.30)	<interface>SplineCurve; <interface>BSplineCurve <datatype>BSplineData
S100_GM_PolynomialSpline	GM_PolynomialSpline (6.4.27); GM_CubicSpline (6.4.28)	<interface>PolynomialSpline; <interface>CubicSpline

All the "classes" in the draft revision of ISO 19107 are "interfaces", and the representation of the coordinates is an implementation decision. The new classes are therefore given an “S100_” prefix.

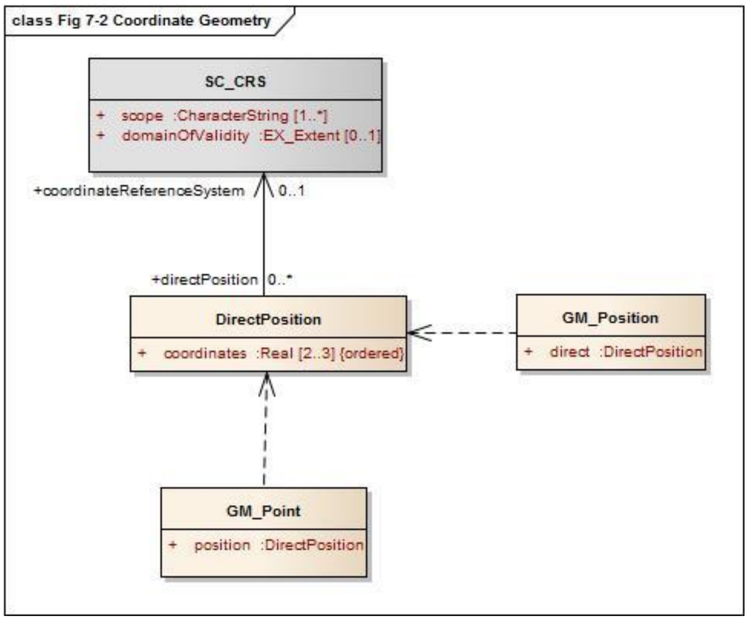

class Fig 7-2 Coordinate Geometry
SC_CRS
+ scope :CharacterString [1..*]
+ domainOfValidity :EX_Extent [0..1]
+coordinateReferenceSystem
0..1
+directPosition
0..*
DirectPosition
+ coordinates :Real [2..3] {ordered}
GM_Position
+ direct :DirectPosition
GM_Point
+ position :DirectPosition

Figure 7-2 — Coordinate Geometry

7-4.1.2 DirectPosition

7-4.1.2.1 Semantics

DirectPosition holds the coordinates for a position within a particular coordinate reference system. In this profile, the associated *SC_CRS* must be linked at the *GM_Aggregate* level and not directly to a *DirectPosition*.

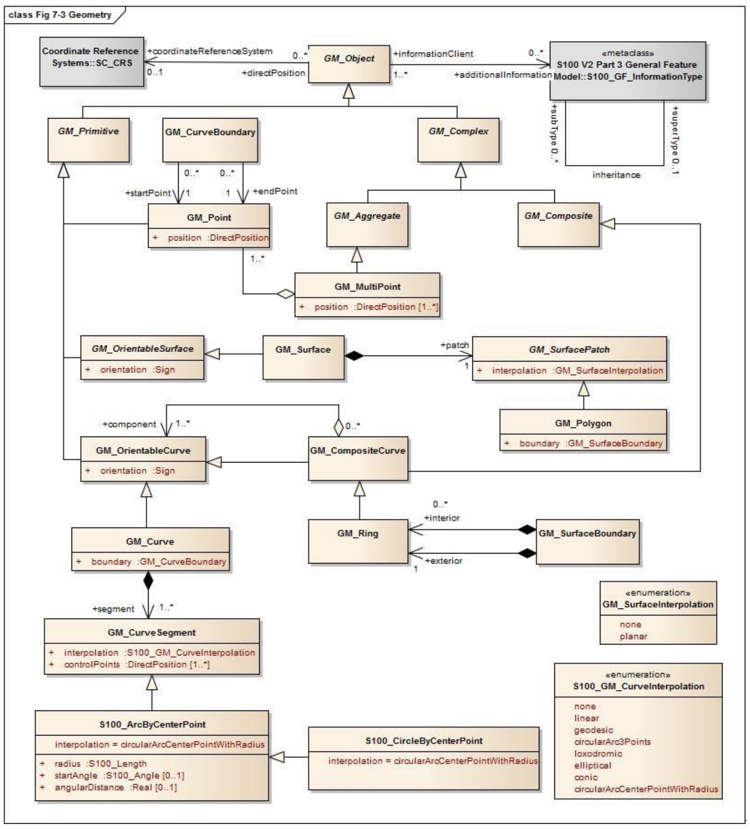
7-4.1.3 GM_Position

7-4.1.3.1 Semantics

The data type *GM_Position* (Figure 7-2) consists of either a *DirectPosition* or a reference to a *GM_Point* (*GM_PointRef*) from which a *DirectPosition* can be obtained.

This profile does not permit the use of the indirect position (*GM_PointRef*).

7-4.2 Simple geometry


class Fig 7-3 Geometry
Coordinate Reference Systems::SC_CRS
+coordinateReferenceSystem
0..1
0..*
+directPosition
GM_Object
+informationClient
1..*
0..*
+additionalInformation
«metaclass» S100 V2 Part 3 General Feature Model::S100_GF_InformationType
+subType 0..*
+superType 0..1
inheritance
GM_Primitive
GM_CurveBoundary
GM_Complex
0..*
0..*
+startPoint
1
1
+endPoint
GM_Point
+ position :DirectPosition
GM_Aggregate
GM_Composite
1..*
GM_MultiPoint
+ position :DirectPosition [1..*]
GM_OrientableSurface
+ orientation :Sign
GM_Surface
+patch
1
GM_SurfacePatch
+ interpolation :GM_SurfaceInterpolation
GM_Polygon
+ boundary :GM_SurfaceBoundary
+component
1..*
0..*
GM_OrientableCurve
+ orientation :Sign
GM_CompositeCurve
0..*
+interior
GM_Ring
GM_SurfaceBoundary
1
+exterior
GM_Curve
+ boundary :GM_CurveBoundary
+segment
1..*
GM_CurveSegment
+ interpolation :S100_GM_CurveInterpolation
+ controlPoints :DirectPosition [1..*]
«enumeration» GM_SurfaceInterpolation
none
planar
«enumeration» S100_GM_CurveInterpolation
none
linear
geodesic
circularArc3Points
loxodromic
elliptical
conic
circularArcCenterPointWithRadius
S100_ArcByCenterPoint
interpolation = circularArcCenterPointWithRadius
+ radius :S100_Length
+ startAngle :S100_Angle [0..1]
+ angularDistance :Real [0..1]
S100_CircleByCenterPoint
interpolation = circularArcCenterPointWithRadius

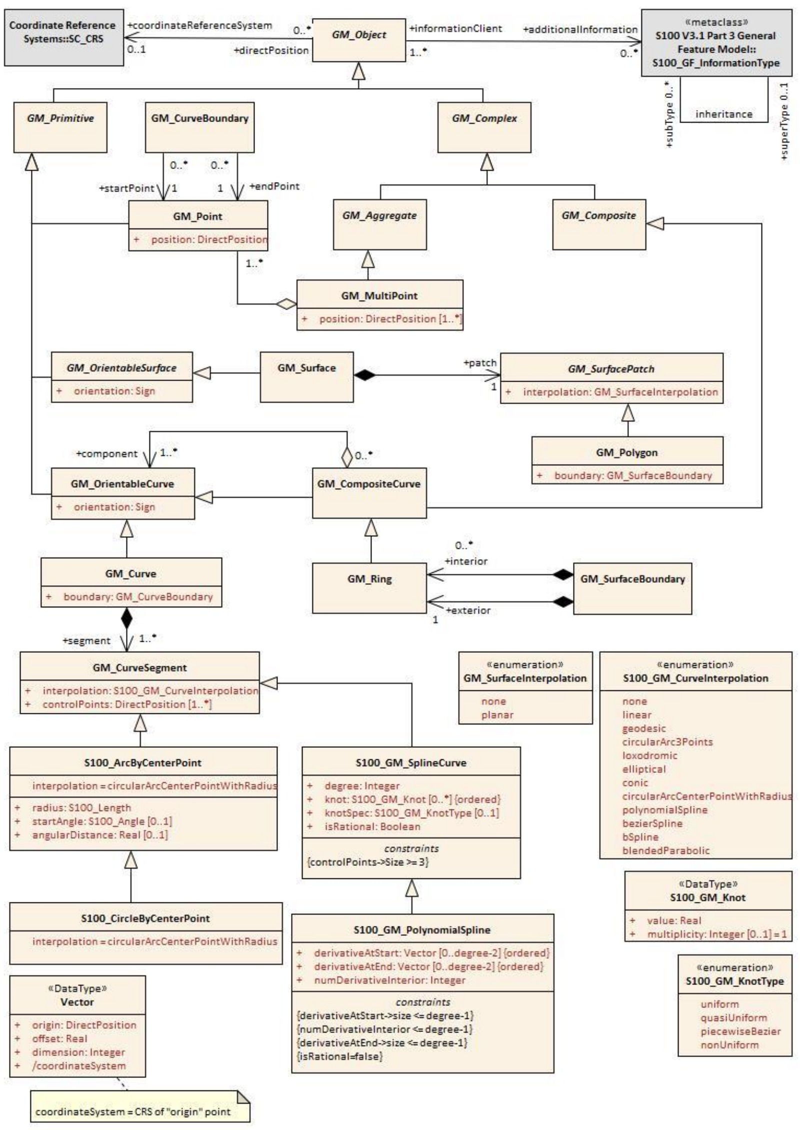

Coordinate Reference Systems::SC_CRS
+coordinateReferenceSystem
0..1
0..*
+directPosition
GM_Object
+informationClient
1..*
+additionalInformation
0..*
«metaclass» S100 V3.1 Part 3 General Feature Model:: S100_GF_InformationType
+subType 0..*
inheritance
+superType 0..1
GM_Primitive
GM_CurveBoundary
GM_Complex
+startPoint
+endPoint
GM_Point
+ position: DirectPosition
GM_Aggregate
GM_Composite
GM_MultiPoint
+ position: DirectPosition [1..*]
GM_OrientableSurface
+ orientation: Sign
GM_Surface
+patch
GM_SurfacePatch
+ interpolation: GM_SurfaceInterpolation
GM_Polygon
+ boundary: GM_SurfaceBoundary
+component
GM_OrientableCurve
+ orientation: Sign
GM_CompositeCurve
GM_Curve
+ boundary: GM_CurveBoundary
GM_Ring
+interior
+exterior
GM_SurfaceBoundary
+segment
GM_CurveSegment
+ interpolation: S100_GM_CurveInterpolation
+ controlPoints: DirectPosition [1..*]
«enumeration» GM_SurfaceInterpolation
none
planar
«enumeration» S100_GM_CurveInterpolation
none
linear
geodesic
circularArc3Points
loxodromic
elliptical
conic
circularArcCenterPointWithRadius
polynomialSpline
bezierSpline
bSpline
blendedParabolic
S100_ArcByCenterPoint
interpolation = circularArcCenterPointWithRadius
+ radius: S100_Length
+ startAngle: S100_Angle [0..1]
+ angularDistance: Real [0..1]
S100_GM_SplineCurve
+ degree: Integer
+ knot: S100_GM_Knot [0..*] {ordered}
+ knotSpec: S100_GM_KnotType [0..1]
+ isRational: Boolean
constraints
{controlPoints->Size >= 3}
«DataType» S100_GM_Knot
+ value: Real
+ multiplicity: Integer [0..1] = 1
S100_CircleByCenterPoint
interpolation = circularArcCenterPointWithRadius
S100_GM_PolynomialSpline
+ derivativeAtStart: Vector [0..degree-2] {ordered}
+ derivativeAtEnd: Vector [0..degree-2] {ordered}
+ numDerivativeInterior: Integer
constraints
{derivativeAtStart->size <= degree-1}
{numDerivativeInterior <= degree-1}
{derivativeAtEnd->size <= degree-1}
{isRational=false}
«enumeration» S100_GM_KnotType
uniform
quasiUniform
piecewiseBezier
nonUniform
«DataType» Vector
+ origin: DirectPosition
+ offset: Real
+ dimension: Integer
+ /coordinateSystem
coordinateSystem = CRS of "origin" point

Figure 7-3 — Geometry

7-4.2.1 S100_GM_CurveInterpolation

7-4.2.1.1 Semantics

S100_GM_CurveInterpolation (Figure 7-3) is a list of codes to be used to identify the interpolation mechanisms specified by an application schema.

In this profile, the types of interpolation available are limited to the following:

1) None (none) – the interpolation is not specified. The assumption is that the curve conforms to the spatial object type, if constrained by that (for example, for arcs and circles) or, if not so constrained, is loxodromic.

2) Linear (linear) – the interpolation is defined by a series of DirectPositions on a straight line between each consecutive pair of controlPoints.

3) Geodesic (geodesic) – the interpolation mechanism shall return DirectPositions on a geodesic curve between each consecutive pair of controlPoints. A geodesic curve is a curve of shortest length. The geodesic shall be determined in the coordinate reference system of the *GM_Curve* in which the *GM_CurveSegment* is used.

4) Circular arc by 3 points (circularArc3Points) – the interpolation defined by a series of three DirectPositions on a circular arc passing from the start point through the middle point to the end point for each set of three consecutive controlPoints. The middle point is located halfway between the start and end point.

5) Loxodromic (loxodromic) – the interpolation method shall return DirectPositions on a loxodromic curve between each consecutive pair of controlPoints. A loxodrome is a line crossing all meridians at the same angle, that is a path of constant bearing.

6) Elliptical arc (elliptical): for each set of four consecutive controlPoints, the interpolation mechanism shall return DirectPositions on an elliptical arc passing from the first controlPoint through the middle controlPoints in order to the fourth controlPoint. Note: if the four controlPoints are co-linear, the arc becomes a straight line. If the four controlPoints are on the same circle, the arc becomes a circular one.

7) Conic arc (conic): the same as elliptical arc but using five consecutive points to ~~determina~~ determine a conic section.

8) Circular arc with centre and radius (circularArcCenterPointWithRadius) – the interpolation is defined by an arc of a circle of the specified radius centred at the position given by the single control point. The arc starts,at the start angle parameter and extends for the angle given by the angular distance parameter. This interpolation type shall be used only with S100_ArcByCenterPoint and S100_CircleByCenterPoint geometry. The precise semantics of the parameters are defined in clause 7-5.2.20 (S100_ArcByCenterPoint).

9) Polynomial (polynomialSpline) – the control points are ordered as in a line-string, but they are spanned by a polynomial function. Normally, the degree of continuity is determined by the degree of the polynomials chosen.

10) Bézier Spline (bezierSpline) – the data are ordered as in a line string, but they are spanned by a polynomial or spline function defined using the Bézier basis. Normally, the degree of continuity is determined by the degree of the polynomials chosen.

11) B-spline (bSpline) – the control points are ordered as in a line string, but they are spanned by a polynomial or rational (quotient of polynomials) spline function defined using the B-spline basis functions (which are piecewise polynomials). The use of a rational function is determined by the Boolean flag "isRational." If isRational is TRUE then all the DirectPositions associated with the control points are in homogeneous form. Normally, the degree of continuity is determined by the degree of the polynomials chosen.

~~8)~~12) Blended parabolic (blendedParabolic) – the control points are ordered as in a line-string, but are spanned by a function that blends segments of parabolic curves defined by triplet sequences of successive data points. Each triplet includes the final two points of its predecessor. Further details of the semantics are provided in clause 7-4.2.2.2.

7-4.2.2 GM_CurveSegment

7-4.2.2.1 Semantics

A *GM_CurveSegment* (Figure 7-3) defines the position, shape and orientation of a single *GM_Curve*. A *GM_CurveSegment* consists either of positions which are joined by straight lines, or positions which fall on a line defined by a particular type of interpolation as described in 7-5.2.1.

7-4.2.2.2 Semantics of specific interpolations

The curve interpolation type *blendedParabolic* is intended for representing smooth curves (or segments) using a reasonably low number of control points. This interpolation type means that the curve segment encoded in the control point array is composed of sequentially blended parabolic curves. The parabolic curves to be blended are determined by successive triplets of control points. Each triplet shares the two last points of its predecessor. The figure below illustrates the concept.

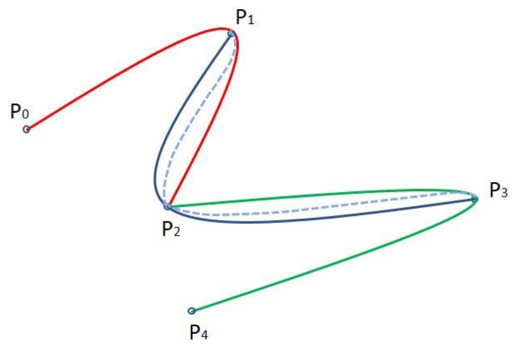


Figure 7-4 Ilustration of blended parabolic interpolation

The sequence of 5 points P_0 – P_4 determines 3 parabolic segments: S1(P_0-P_1-P_2), S2(P_1-P_2-P_3) and S3(P_2-P_3-P_4). The curve between P_1 and P_2 is determined by blending the P_1-P_2 segments of S1 and S2 while the curve between P_2 and P_3 is determined by blending the P_2-P_3 segments of S2 and S3. The resultant curve between P_1 – P_3 is shown by the dashed line.

The resultant curve between two control points is computed as a combination of the two parabolas which share the two control points, for example it may be computed using the convex combination $S(P_i, P_{i+1}) = (1-t) * S_i + t * S_{i+1}$ where t varies from 0 to 1 as the path progresses from P_i to P_{i+1}.

In practice it is not necessary to compute the equations of the parabolas to be blended, as the interpolated points can be computed using the coordinates of the control points. For example, the convex combination given earlier results in the formula below for the curve from P_k to P_{k+1}, which is applied to the X and Y dimensions separately:

$$P(t) = P_k + \tfrac{1}{2} t (P_{k+1} - P_{k-1})$$
$$- \tfrac{1}{2} t^2 (P_{k+2} - 4P_{k+1} + 5P_k - 2P_{k-1})$$
$$+ \tfrac{1}{2} t^3 (P_{k+2} - 3P_{k+1} + 3P_k - P_{k-1})$$

For open curves the first interpolated segment of the curve segment as a whole can be generated by adding a fictitious point preceding the first point in the control point array, with coordinate values such that the second derivative in t-space (the *acceleration* of the curve) at the first point in the control point array is zero. (This allows the use of the same blending formula as for the rest of the curve.) The final interpolated segment can be computed in a similar manner by adding a fictitious control point after the last point in the array.

For closed curves, continuity and smoothness at the first and last point in the control point array require that the first triplet of control points be the same as the last triplet (or equivalently, that the curve be specifically designated as a closed curve so that the construction procedure can 'wrap around' the beginning and end of the control points array).

Due to distortions caused by applying plane methods to curved surfaces, the *blendedParabolic* interpolation should not be used where the precise location of the resultant curve is important. (It is possible to achieve higher precision by increasing the number of control points, but that defeats the pupose of using this interpolation type.)

Curves with this interpolation type have the following characteristics:

- Smooth representations with a reasonably low number of control points. However, the smoothness properties are not as high quality as cubic splines.
- Less expensive computationally than cubic splines.
- Better local control – for example, moving a control point affects only the two segments it begins and terminates and their immediate neighbors.
- There must be at least 3 points in the control points array.

7-4.2.3 GM_SurfaceInterpolation

7-4.2.3.1 Semantics

GM_SurfaceInterpolation (Figure 7-3) is a list of codes which are used to identify the method of interpolation.

In this profile, the types of *interpolation* are constrained to the following:

1) None (none) – the interior of the surface is not specified. The assumption is that the surface follows the reference surface defined by the coordinate reference system.

2) Planar (planar) – the interpolation is a section of a planar, or flat, surface. The boundary in this case shall be contained within that plane.

7-4.2.4 GM_SurfacePatch

7-4.2.4.1 Semantics

The *GM_SurfacePatch* (Figure 7-3) is the abstract root class for all 2-dimensional geometric constructs. It uses a single interpolation to define the shape and position of the associated *GM_Surface* primitives.

7-4.2.5 GM_Polygon

7-4.2.5.1 Semantics

A *GM_Polygon* (Figure 7-3) is defined by a boundary (see 7-5.2.7 below) and an underlying surface to which this boundary is connected. The polygon uses planar interpolation. A *GM_Polygon* is a subtype of *GM_SurfacePatch*.

7-4.2.6 GM_Curve

7-4.2.6.1 Semantics

GM_Curve (Figure 7-3) is a descendent subtype of *GM_Primitive* through *GM_OrientablePrimitive*. It is the basis for 1-dimensional geometry. A curve is a continuous image of an open interval and so could be written as a parameterized function such as c(t):(a, b)$\rightarrow E_n$ where "t" is a real parameter and En is Euclidean space of dimension n (usually 2 or 3, as determined by the coordinate reference system). Any other parameterization that results in the same image curve, traced in the same direction, such as any linear shifts and positive scales such as e(t) = c(a + t(b-a)):(0,1) $\rightarrow E_n$, is an equivalent representation of the same curve. For the sake of simplicity, *GM_Curve* should be parameterized by arc length, so that the parameterization operation inherited from *GM_GenericCurve* (see ISO 19107 clause 6.4.7) will be valid for parameters between 0 and the length of the curve.

Curves are continuous, connected, and have a measurable length in terms of the coordinate system. The orientation of the curve is determined by this parameterization, and is consistent with the tangent function, which approximates the derivative function of the parameterization and shall always point in the "forward" direction. The parameterization of the reversal of the curve defined by c(t):(a, b)→E_n would be defined by a function of the form s(t) = c(a + b - t):(a, b)→E_n.

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.

Individual product specifications may constrain the interpolation types allowed for spatial attributes.

EXAMPLE: An isobar feature is constrained to curves consisting only of segments with interpolation type *polynomialSpline* and degree 3 (i.e., cubic splines).

7-4.2.7 GM_CurveBoundary

7-4.2.7.1 Semantics

The boundary of *GM_Curve* shall be represented as *GM_CurveBoundary*.

7-4.2.8 GM_OrientableCurve

7-4.2.8.1 Semantics

A *GM_OrientableCurve* (Figure 7-3) is a *GM_Curve* with an associated orientation inherited from *GM_OrientablePrimative*.

7-4.2.9 GM_OrientableSurface

7-4.2.9.1 Semantics

A *GM_ OrientableSurface* (Figure 7-3) is a *GM_Surface* with an associated orientation inherited from its *GM_OrientablePrimative* parent.

7-4.2.10 GM_Point

7-4.2.10.1 Semantics

GM_Point (Figure 7-3) is a 0-dimensional geometric primitive (*GM_Primitive*).

GM_Point is the data type for a geometric object consisting of one and only one point.

7-4.2.11 GM_Primitive

7-4.2.11.1 Semantics

GM_Primitive (Figure 7-3) is the abstract root class for all geometric primitives defined in this profile. A *GM_Primitive* is a *GM_Object*. *GM_Primitive* consists of three sub-types. *GM_Point* which is 0 -dimensional; *GM_Curve* which is 1-dimensional and *GM_Surface* which is 2-dimensional. All geometric primitives (*GM_Primitive*) must be part of at least one *GM_Aggregate* (see ISO 19107 clause 8.10.1). There is no direct link between each *GM_Primitive* and the coordinate reference system *SC_CRS* used for defining the position of the *GM_Primitive.* All *GM_Primitive* contained within a *GM_Aggregate* use the same *SC_CRS* for defining their position.

7-4.2.12 GM_Ring

7-4.2.12.1 Semantics

A *GM_Ring* (Figure 7-3) is composed of a number of references to *GM_OrientableCurves*. The endpoint of *GM_OrientableCurve* “n” is the startPoint of *GM_OrientableCurve* “n+1” and the first startpoint is coincident with the last endpoint, meaning the *GM_Ring* is closed. A *GM_Ring* must be simple, that is it does not intersect itself.

7-4.2.13 GM_Surface

7-4.2.13.1 Semantics

GM_Surface (Figure 7-3) is a subclass of *GM_Primitive* and is the basis for 2-dimensional geometry. It is a *GM_OrientableSurface* with a positive orientation.

This profile does not use instances of *GM_Surface*. A *GM_Surface* within this profile must be subtyped as a *GM_Polygon.*

7-4.2.14 GM_SurfaceBoundary

7-4.2.14.1 Semantics

The boundary of *GM_Surfaces* shall be represented as *GM_SurfaceBoundary* (Figure 7-3).

A *GM_SurfaceBoundary* consists of references to a combination of at least one exterior *GM_Ring* and zero or more interior *GM_Ring*. The rings must be closed as described in ISO 19107 Clause 6.6.11.1.

7-4.2.15 GM_Complex

7-4.2.15.1 Semantics

A *GM_Complex* (Figure 7-3) is a collection of geometrically separate, simple *GM_Primitive*. If a *GM_Primitive* (other than a *GM_Point*) is in a particular *GM_Complex*, then there exists a set of primitives of lower dimension in the same complex that form the boundary of this primitive. For example a *GM_Surface* is a 2 dimensional object, its boundary consists of *GM_Curve* which are 1 dimensional.

7-4.2.16 GM_Composite

7-4.2.16.1 Semantics

A geometric composite, *GM_Composite* (Figure 7-3), is a collection of primitives which must have geometry of the same type and which could exist as a single example of that primitive. For example, a composite curve is a collection of curves which could equally be represented by a single curve. This does not apply to *GM_Point* which can only contain one point.

7-4.2.17 GM_CompositeCurve

7-4.2.17.1 Semantics

A *GM_CompositeCurve* (Figure 7-3) has all the geometric properties of a curve. A composite curve is a sequence of *GM_OrientableCurve*, each curve (except the first) begins where the previous curve ends.

7-4.2.18 GM_Aggregate

7-4.2.18.1 Semantics

The aggregates, *GM_Aggregate* (Figure 7-3) gather geometric objects. Since they will often use orientation modification, the curve reference and surface references do not go directly to the *GM_Curve* and *GM_Surface*, but are directed to G*M_OrientableCurve* and *GM_OrientableSurface*.

Most geometric objects are contained in features, and cannot be held in collections that are strong aggregations. For this reason, the collections described in this clause are all weak aggregations, and shall use references to include geometric objects.

NOTE The subclasses of *GM_OrientablePrimitive* are handled in such a manner that the reference object can link to a specific orientation of that object.

7-4.2.19 GM_MultiPoint

7-4.2.19.1 Semantics

GM_MultiPoint is an aggregate class containing only points. The association role "element" shall be the set of *GM_Point* contained in this *GM_MultiPoint*.

7-4.2.20 S100_ArcByCenterPoint

7-4.2.20.1 Semantics

An S100_ArcByCenterPoint is an arc of the circle with centre given by the single control point and radius given by the *radius* parameter. Radius is geodesic distance from the centre. The arc starts at the bearing given by the *start angle* attribute and ends at the bearing calculated by adding the value of the *angular distance* parameter to the start angle. The direction of the arc is given by the sign of the angular distance, with positive values indicating a clockwise direction with respect to an observer located vertically above the centre point. Bearings are relative to true north except that arcs centred at either pole (where true north is undefined or ambiguous) shall use the prime meridian as the reference direction.

Start angle must be in degrees and is limited to the range [0.0, 360.0]. Angular distance must be in degrees and is limited to the range [-360.0, +360.0]. The upper bound on radius varies with location and reference geoid but shall be less than the minimum geodesic distance from the position of the centre to its antipodal point. Tools or product specifications may impose a lower limit on radius.

7-4.2.21 S100_CircleByCenterPoint

7-4.2.21.1 Semantics

An S100_CircleByCenterPoint is a circle with centre given by the single control point and radius given by the *radius* parameter. Start angle and angular distance may be omitted. The semantics and limits of the attributes are the same as S100_ArcByCenterPoint with start angle assumed to be 0.0° and angular distance assumed to be +360.0° if not provided. If provided, angular distance must be +360.0 or -360.0.

7-4.2.22 S100_GM_SplineCurve

7-4.2.22.1 Semantics

All splines share the property that they can be represented by parametric functions that map into the coordinate system of the geometric object that they will represent. Spline Curves come in essentially two forms: interpolant and approximant.

Interpolating splines ("interpolant") pass through each of the given control points. In general, the curves are defined by their data points with extra conditions at boundary points (the data points at either end of the segment), and the level of continuity (e.g., C^0 continuity at a point means the curve is connected at the point; C^1 that the segments on either side have the same first derivative at the point). A cubic spline passes through each data point, is continuous and has a smooth tangent at each point.

The second type ("approximants") only approximate the control points. These splines use sets of real valued functions are all defined on a single common domain (e.g., the interval [0.0, 1.0]), are always non-negative in their values and always sum as a complete set to 1.0 for their entire domain. These functions are used in vector equations so that the tracing of the curve is a weighted average. The spline curve always lies in the convex hull of the control points. Since such functions are defined in vector form, they can generally be used in any target dimension coordinate system.

Approximants have nice properties involving ease of representation, ease of calculation, smoothness, and some form of convexity. They do not usually pass through the control point, but if the control point array is dense enough, the local properties will force a good approximation of them, and will give a well-behaved curve in terms of shape and smoothness.

S100_GM_SplineCurve and its subclass(es) must have values of curveInterpolation that are appropriate to the type of curve, i.e., one of polynomialSpline, bezierSpline, or bSpline as appropriate.

Due to distortions caused by applying plane methods to curved surfaces and the nature of splines and blended curve as approximations, the various spline and *blendedParabolic* interpolations should not be used where accuracy in the location of the resultant curve is important, such as defining the boundaries of restricted areas. (In principle it is possible to produce high-precision curves by increasing the number of control points, but that defeats the purpose of using these interpolation types.)

For the reasons mentioned in 7-4.1.1.1 and the omission of curveForm, this class is given an 'S100_' prefix.

7-4.2.22.2 Attributes

knot: The attribute "knot" is an array of knots, each of which define a value in the parameter space of the spline, and will be used to define the spline basis functions. The *knot* data type holds information on knot multiplicity. The parameter values in this array must be monotonic and strictly increasing, i.e., each value must be greater than its predecessor.

degree: The attribute "degree" shall be the degree of the polynomials used for defining the interpolation. Rational splines will have this degree is the limiting degree for both the numerator and denominator of the rational functions being used for the interpolation.

knotSpec: The attribute "knotSpec" gives the type of knot distribution used in defining this spline. This is for information and possible implementation optimizations, and must be set according to the different construction-functions.

isRational: The attribute "isRational" indicates that the spline uses rational functions to define the curve. This is done by creating a polynomial spline on homogeneous coordinates, and projecting back to regular coordinates when all calculations are done. The attribute "isRational" must be "TRUE" if and only if the control points of the spline are in homogeneous coordinates, each point having a weight.

The ISO 19107 attribute "curveForm" is not used since it is for information only, used to capture the original intention.

7-4.2.22.3 Semantics of specific varieties

A B-spline is a piecewise parametric polynomial or rational curve described in terms of control points and basis functions. If the knotSpec is not present, then the knotType is uniform and the knots are evenly spaced, and except for the first and last have multiplicity = 1. At the ends the knots are of multiplicity = degree+1. If the knotType is uniform they need not be specified. B-splines must have curveInterpolation set to *bSpline*. The basis functions for B-splines depend on the degree and are defined in textbooks in mathematics, computer graphics, and computer-aided geometric design.

A B-spline curve is a piecewise Bézier curve if it is quasi-uniform except that the interior knots have attribute multiplicity[1] = “degree” rather than having multiplicity one. In this subtype the knot spacing shall be 1.0, starting at 0.0. A piecewise Bézier curve that has only two knots, 0.0, and 1.0, each of multiplicity (degree+1), is equivalent to a simple Bézier curve.

Bézier splines are polynomial splines that use Bézier or Bernstein polynomials for interpolation purposes. These polynomials are defined in textbooks in mathematics, computer graphics, and computer-aided geometric design. Bézier splines must have curveInterpolation set to *bezierSpline*.

7-4.2.23 S100_GM_PolynomialSpline

7-4.2.23.1 Semantics

A polynomial spline is a polynomial curve passing through the points in the control points array. Construction of such a spline depends on the constraints, which may include:

- restrictions on values or derivatives of the spline at the data points
- restrictions on the continuity of various derivatives at chosen points
- degree of the polynomial in use.

A~~n~~ polynomial spline of degree *n* shall be defined piecewise between knot parameter values, as an n-degree polynomial, with up to C^{n-1} continuity at the control points where the defining polynomial may change.

This level of continuity shall be controlled by the attribute numDerivativesInterior, which shall default to (degree-1).

Constructive parameters may include constraints for as many as "degree – 1" derivatives of the polynomials at each knot.

The major difference between the polynomial splines, the b-splines (basis splines) and Bézier splines is that polynomial splines pass through their control points, making the control point and sample point array identical.

[1] This is the attribute named “multiplicity” of class GM_Knot.

Formatted: English (United States)

7-4.2.23.2 Attributes

derivativeAtStart, derivativeAtEnd (vector): The attribute "derivativeAtStart" shall be the values used for the initial derivatives (up to degree – 2) used for interpolation in this curve at the start point of the spline. The attribute "derivativeAtEnd" shall be the values used for the final derivative (up to degree – 2) used for interpolation in this curve at the end point of the spline. These attributes are used to ensure continuity and smoothness with predecessor and successor curves if any, e.g., if this curve segment is one of a sequence of curve segments, or if the curve is part a composite curve.

numDerivativesInterior (Integer): The attribute numDerivativesInterior is the number of continuous derivatives required at interior knots (i.e., between the first and the last knot). The attribute "numDerivativesInterior" specifies the type of continuity that is guaranteed interior to the curve. The value of "0" means C^0 continuity (which is a mandatory minimum level of continuity) the value "1" means C^1 continuity, etc.

7-4.2.23.3 Semantics of specific varieties

Cubic splines are polynomial splines with degree = 3. The number of points in the control points array must be 3*N+1 where N is the number of cubic pieces.

7-4.2.24 S100_GM_Knot (DataType)

7-4.2.24.1 Semantics

The knots are values from the domain of a constructive parameter space for curves, surfaces and solids[2]. Each knot sequence is used for a dimension of the parameter space $k_i = \{u_0, u_1, u_2 ...\}$. Thus, in a surface using a functional interpolation such as a b-spline, there will be two knot-sequences, one for each parameter, $k_{i,j} = (u_i, v_j)$.

In the knot sequence for a b-spline, a knot can be repeated (affecting the underlying spline formulae). In other curves, knots will all be multiplicity 1. In S-100 knot sequences are represented as in the ISO 19107 (~~:~~2017 draft) model, i.e., distinct values accompanied by a multiplicity, i.e., $k_i = (t \in \mathrm{R}, m \in \mathrm{Z})$. The alternative storage form (simple sequence, with repetitions or each knot according to its multiplicity) is acceptable in a data format if required by the encoding standard on which the data format is based.

7-4.2.24.2 Attributes

value (Real). The attribute "value" is the value of the parameter at the knot of the spline. The values of successive knots must be monotonically increasing.

Multiplicity (Integer): The attribute "multiplicity" is the multiplicity of the knot.

7-4.2.25 S100_GM_KnotType

7-4.2.25.1 Semantics

A B-spline is uniform if and only if all knots are of multiplicity one and they differ by a positive constant from the preceding knot. A B-spline is quasi-uniform if and only if the knots are of multiplicity (degree+1) at the ends, of multiplicity one elsewhere and they differ by a positive constant from the preceding knot. This enumeration is used to describe the distribution of knots in the parameter space of various splines. Possible values are:

1) Uniform (uniform): knots are equally spaced, all multiplicity 1.
2) Non-uniform (nonUniform): knots have varying spacing and multiplicity.
3) Quasi-Uniform (quasiUniform): the interior knots are uniform, but the first and last have multiplicity one larger than the degree of the spline (p+1).

[2] Solids are not implemented in S-100.

Formatted: English (United States)

~~1)~~4) Piecewise Bézier (piecewiseBezier): the underlying spline is formally a Bézier spline, but knot multiplicity is always the degree of the spline except at the ends where the knot degree is (p+1). Such a spline is a pure Bézier spline between its distinct knots.

7-4.2.26 Vector

The datatype Vector must be associated with a point on the GeometricReferenceSurface (e.g., the surface of the geoid) to be well defined. The attributes of the vector also specify the "start position" of the vector.

7-4.2.26.1 Attributes

origin: DirectPosition – The attribute origin is the location of the point on the GeometricReferenceSurface for which the vector is a tangent. The direct position is associated with a coordinate system; this determines the coordinate system for the vector. The direct position's spatial dimension determines the dimension of the vector.

offset: Real [1..*] – The attribute offset uses the coordinate system of the direct position and represents the local tangent vector in terms of the differentials of the local coordinates. The offset values are the magnitude of the vector along each coordinate axis.

dimension: Integer – The attribute dimension is the dimension of the origin and therefore the dimension of the local tangent space of the vector.

coordinateSystem: the attribute coordinateSystem is the same as the coordinate system of the origin.

For curve spatial types the origin will be the point at which the vector is defined; the offset will be the latitude and longitude differentials, which together indicate the magnitude and direction of the vector; the dimension will be 2 for curves with control points encoded as latitude/longitude; and the coordinateSystem being the same as that of the origin is not encoded.

7-4.3 Geometry configurations

Figure 7-3 depicts a one size fits all geometry model which can be further constrained in both dimensionality and complexity. This is broken down into 5 basic levels.

7-4.3.1 Level 1 – 0-, 1-Dimension (no constraints)

A set of isolated point and curve primitives. Curves do not reference points (no boundary), points and curves may be coincident. Areas are represented by a closed loop of curves.

7-4.3.2 Level 2a – 0-, 1-Dimension

A set of point and curve primitives with the following constraints:

1) Each curve must reference a start and end point (they may be the same).
2) Curves must not self-intersect as shown in Figure 7-~~4~~5.
3) Areas are represented by a closed loop of curves beginning and ending at a common point.
4) In the case of areas with holes, all internal boundaries must be completely contained within the external boundary and the internal boundaries must not intersect each other or the external boundary. Internal boundaries may touch other internal boundaries or the external boundary tangentially (that is at one point) as shown in Figure 7-~~5~~6.
5) The outer boundary of a surface must be in a clockwise direction (surface to the right of the curve) and the curve orientation positive. The inner boundary of a surface must be in a counter-clockwise direction (surface to the right of the curve) and the curve orientation negative as shown in Figure 7-~~6~~7.

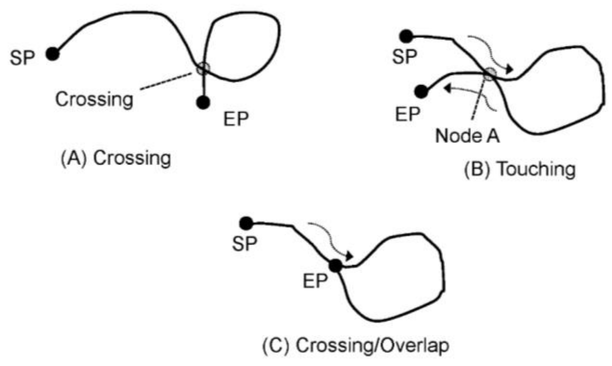


Figure 7-54 Self Intersect Example (Invalid Geometries)

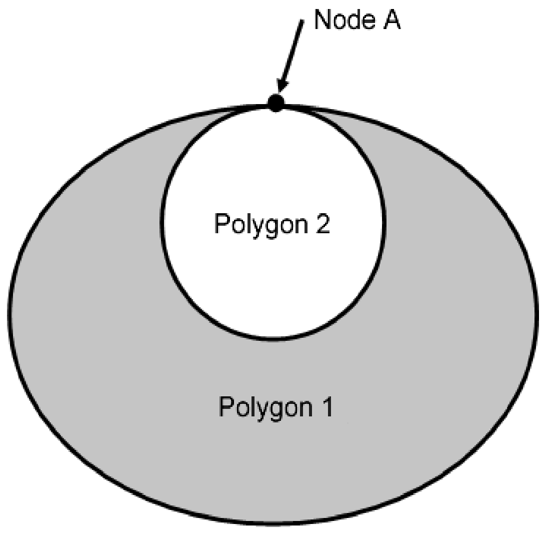


Figure 7-6~~5~~ Area Holes (Valid Geometries)

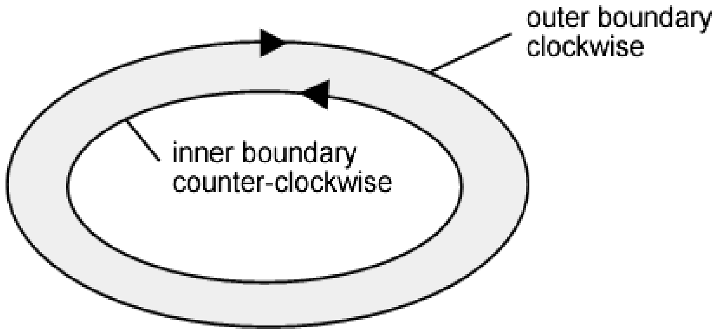


Figure 7-7~~6~~ — Boundary Direction

7-4.3.3 Level 2b – 0-, 1-Dimension

A set of point and curve primitives. The constraints for Level 2a apply plus the following:

1) Each set of primitives must form a geometric complex;
2) Curves must not intersect without referencing a point at the intersection;
3) Duplication of coincident geometry is prohibited.

7-4.3.4 Level 3a – 0-, 1- and 2-Dimension

A set of point, curve and surface primitives. The constraints for Level 2a applies.

7-4.3.5 Level 3b – 0-, 1- and 2-Dimension

A set of point, curve and surface primitives. The constraints for Levels 2a and 2b apply plus the following:

1) Surfaces must be mutually exclusive and provide exhaustive cover.

Appendix 7-A **Examples** (informative)

7-A-1. Curve Example

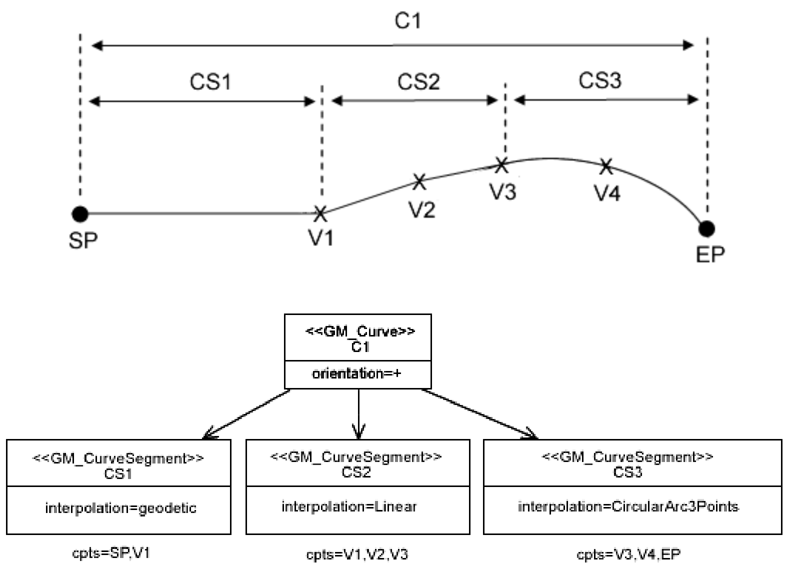
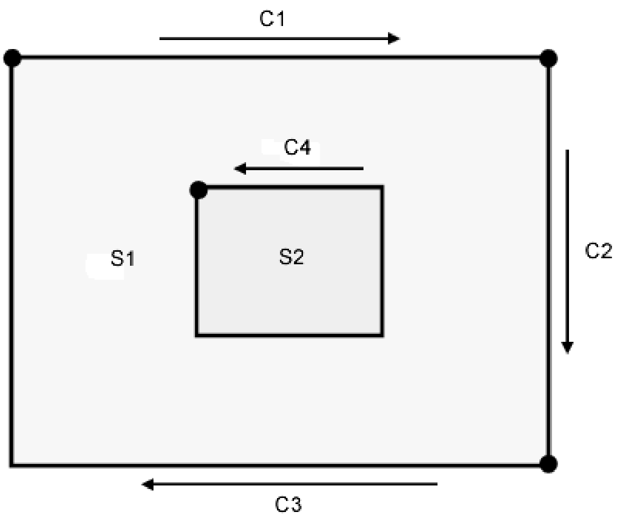


Figure 7-A.1 — Curve Example

The following describes the geometrical elements of the curve example (Figure 7-A.1).

C1 (GM_Curve) consists of CS1, CS2 and CS3 (GM_CurveSegment). CS1 uses a geodetic interpolation, CS2 linear and CS3 circularArc3Points. SP (start point) and EP (end point) (GM_Point) are the start and end points of C1 and can also be used indirectly as a 0 dimension position for a point feature. An array of control points for each segment consists of a combination of SP, EP and vertices as indicated in the above diagram. The orientation of C1 is + (forward) from SP to EP.

7-A-2. Surface Example



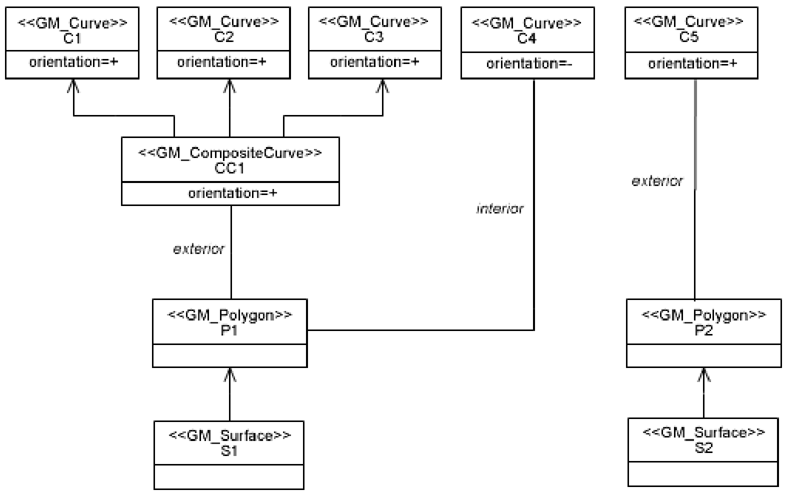


Figure 7-A.2 – Surface Example

The following describes the geometrical elements of the surface example (Figure 7-A.2).

S1 (GM_Surface) is represented by the surface patch P1 (GM_Polygon) the boundary of which consists of exterior and interior rings. The exterior ring CC1 (GM_CompositeCurve) is an aggregation of C1, C2, C3 (GM_Curve), the interior ring C4 is a simple GM_Curve.

7-A-3. 2.5 Dimensional Geometry Example

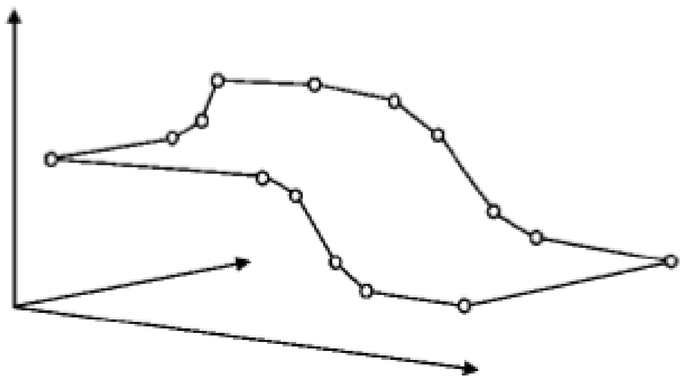


Figure 7-A.3 – 2.5D Example

In the depicted example, the curve which constitutes the exterior boundary of a GM_Polygon consists of an array of 3D control points. Note that the surface interpolation must be “none”, which means that the position of interior points is not determined. The “planar” interpolation would only be acceptable if all points were lying on a plane.

Page intentionally left blank