Quality Information Framework (QIF) – An Integrated Model for Manufacturing Quality Information

Part 3: Model Based Definition (MBD) Information Model and XML Schema File Version 2.0

QIF Version 2.0

ANSI/QIF Part 3–2014

Dimensional Metrology Standards Consortium, Inc. (DMSC)

Prepared and Published by:
Dimensional Metrology Standards Consortium, Inc. (DMSC)
1350 Alsbury Blvd., #514
Burleson, Texas 76028
Tel: (817) 461-1092
email: support@qifstandards.org

www.QIFStandards.org

Contents

Figures

Foreword

The Dimensional Metrology Standards Consortium (DMSC, Inc.) is an American National Standards Institute (ANSI) accredited standards developing organization, as well as an A-Liaison to the International Organization for Standardization (ISO). The mission of the DMSC is to identify urgently needed standards in the field of dimensional metrology, and to promote, foster, and encourage the development and interoperability of these standards, along with related and supporting standards that will benefit the industry as a whole. More information about the DMSC can be found at www.dmsc-inc.org.

The Quality Information Framework (QIF) information model was developed by domain experts from the manufacturing quality (that is metrology) community representing a wide variety of industries and quality measurement needs. Specifically for the QIF Model-Based Definition (QIF MBD) work, contributors include:

- Capvidia
- Mitutoyo America
- National Institute of Standards and Technology
- Origin International Inc.

The bulk of the work on this document was performed by the Complete and Accurate Model-Based Definition (CAMBD) Working Group, approved and revised as needed by the Quality Information Framework (QIF) Working Group, and given final approval for ANSI review by the DMSC's Quality Management Systems (QMS) Committee. More information about DMSC's QIF effort can be found at www.qifstandards.org.

This document is Part 3 of 8 parts of the second publicly reviewed edition of a QIF standard. This version is solely a product of the DMSC and its committees and working groups.

Version 2.0 of QIF consists of the following parts, under the general title *Quality Information Framework (QIF) — An Integrated Model for Manufacturing Quality Information*:

Part 1: Overview and Fundamental Principles Version 2.0

Part 2: QIF Library Information Model and XML Schema Files Version 2.0

Part 3: QIF Model Based Definition (MBD) Information Model and XML Schema File Version 2.0

Part 4: QIF Plans Information Model and XML Schema File Version 2.0

Part 5: QIF Resources Information Model and XML Schema File Version 2.0

Part 6: QIF Rules Information Model and XML Schema File Version 2.0

Part 7: QIF Results Information Model and XML Schema File Version 2.0

Part 8: QIF Statistics Information Model and XML Schema File Version 2.0

HTML-based data model viewer

The DMSC will make available an html-file based data dictionary for the entire QIF information model as an aid to understanding QIF. This data dictionary is non-normative material, but describes the normative content of the QIF data model. The html files facilitate viewing the complete data model, including all six application areas and Library content, using pictures and text. A user has the ability, through an internet browser, to follow navigation links forward and backward through the data model description using mouse clicks.

Introduction

The QIF MBD information model defines an open, 3D model-based product definition (MBD). The model is independent of any CAD system. It includes a full suite of formally modeled characteristics (dimensions, tolerances, etc.) and features, associated to a complete model of parts and assemblies (topology, views, etc.) and metadata. It supports interchange of part and assembly models between CAD systems and transfer of those models to downstream processes.

A QIF MBD instance file can replace a traditional 2D drawing or be used together with a drawing.

QIF MBD defines a digital data format to convey part geometry (typically called the "CAD model") and information to be consumed by downstream manufacturing quality processes, about the essential characteristics of the product. Information defined in a QIF MBD model is more useful than is possible on a 2D drawing. For example, each product manufacturing information (PMI) object can have associated geometry, thereby conveying information to downstream processes. PMI objects include, but are not limited to: dimensions, geometric tolerances, datums, and datum reference frames.

The QIF MBD data format defines manufacturing information to form a "digital bridge" connecting CAD processes for product design and downstream manufacturing quality processes that generate physical products. The QIF MBD data model provides a vendor-neutral format and ensures that software solutions that support it, by either writing or reading QIF instance files, can exchange manufacturing quality data efficiently and accurately.

Quality Information Framework (QIF) — An Integrated Model for Manufacturing Quality Information

Part 3: Model-Based Definition (MBD) Information Model and XML Schema File

1 Scope

1.1 Contents of this document

This standard defines the QIF MBD data model for a portion of the QIF manufacturing quality information model focused on model-based definition and designated as QIF MBD. QIF MBD defines a digital data format to convey part geometry (typically called the "CAD model") and information to be consumed by downstream manufacturing quality processes. The QIF MBD model consists of definitions for data types, elements, the logical relationships between them, and the semantics of the quality information. The QIF MBD model is defined using the XML Schema Definition Language (XSDL). The QIF MBD model is a digital data exchange mechanism that can be easily incorporated in software developed by commercial solution vendors that implements manufacturing quality systems.

The top level of the QIF MBD model is defined in the QIFDocument.xsd XML schema file. That file includes, directly or indirectly, all of the 15 XML schema files in the QIF Library. An XML instance file describing a model-based definition with PMI will contain information items drawn from QIFDocument.xsd and most of the QIF Library files, particularly Product.xsd, Characteristics.xsd, and Features.xsd. The Part 3 document describes primarily the CAD model and assembly model embodied in Product.xsd, Topology.xsd, and Geometry.xsd.

XSDL also supports the definition of rules and validation checks of QIF instance files. Almost all of these check that a connection made by reference to an object that is supposed to be of a particular type does, in fact, reference an object of the correct type. Hundreds of these are included in the overall QIF model, and most of them are applicable to QIF MBD instance files. This document describes these only briefly.

1.2 QIF MBD Information Model Application Architecture

The QIF MBD model is designed to support transfer of model-based definition information between a variety of types of applications as shown in Figure 1. Systems that can generate QIF MBD files are shown on the left, and systems that can consume QIF MBD files on the right.

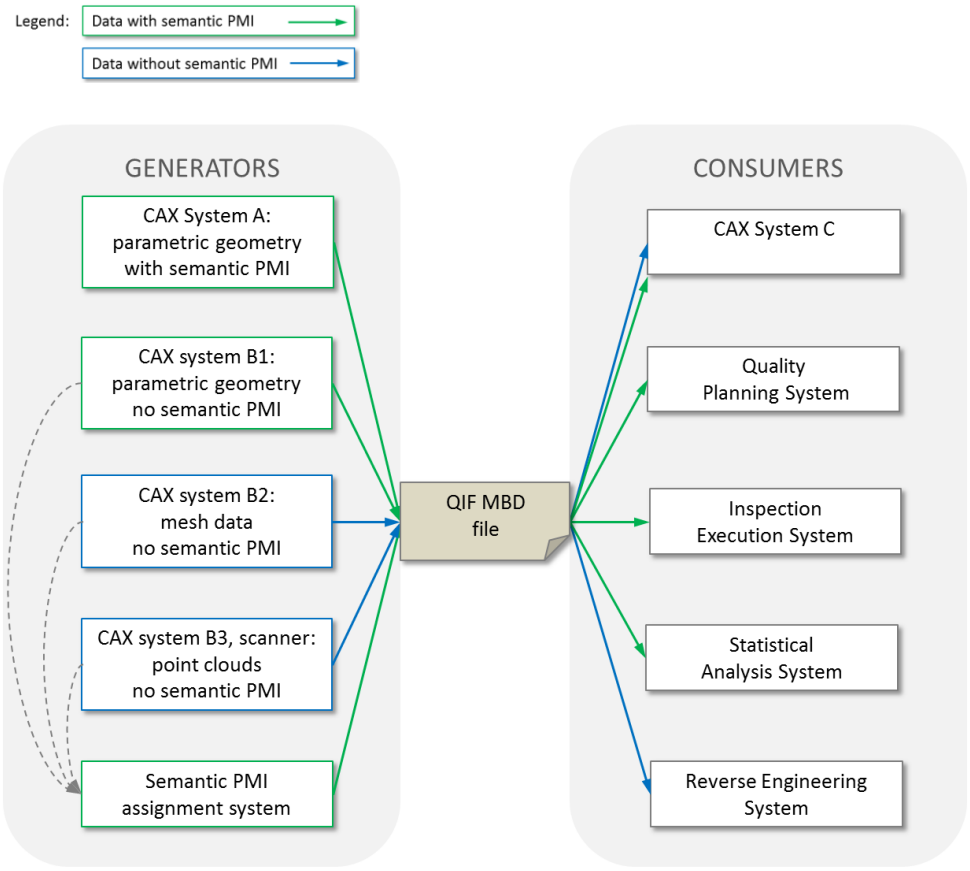


Figure 1 – Workflow of QIF MBD Information

A CAX system such as System A in Figure 1 that has semantic PMI and is QIF-enabled will be able to output a QIF MBD file with semantic PMI.

CAX systems such as Systems B1/B2/B3 in the Figure 1 that do not have semantic PMI and are QIF-enabled will be able to output a QIF MBD file containing a product definition without PMI. Although QIF MBD is designed to include PMI, it is not required.

Output from Systems B1/B2/B3 may be fed into a QIF-enabled semantic PMI assignment system, which would be able to write a QIF MBD file containing the product definition with semantic PMI.

Once a QIF MBD file is generated, by whatever means, it may be fed into any QIF-enabled system.

QIF enabled CAX System C can read QIF MBD files. If System C fully supports semantic PMI, then it can consume all the information from a QIF MBD file. If not, it can only consume product definition without PMI from QIF MBD files. Thus, QIF MBD enables CAD-to-CAD transfers.

A QIF-enabled Quality Planning System can read QIF MBD files. If it does high level planning (the plan specifies only what to inspect), it may only need the PMI stored in a file. If it does low level planning (the plan specifies what and how to inspect) it may need all the information in the file.

Other QIF-enabled quality systems, such as inspection execution systems and statistical analysis systems, will be able to consume a QIF MBD file, produce graphical displays for the user, and use the PMI for calculations.

QIF MBD was designed for the uses described above. QIF MBD files may have additional uses, which are not defined here.

2 Conformance

Software products that implement this specification to write QIF MBD XML instance files must:

- follow the rules of XML when writing QIF MBD instance files
- generate instance files that validate against the QIFDocument schema
- employ semantics of the information written that complies with the referenced standards and with the data dictionary in this specification.

Software products that implement this specification to read QIF MBD instance files must:

- be able to read any valid QIF MBD XML instance file and extract all numerical and semantic data correctly.

3 Normative References

The following referenced documents are indispensable 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 any amendments) applies.

ANSI/DMIS 105.2, Part 1-2009, *Dimensional Measuring Interface Standard, DMIS 5.2 Standard, Part 1*. Also available as ISO 22093:2011 *Industrial automation systems and integration -- Physical device control -- Dimensional Measuring Interface Standard (DMIS)*

ASME B1.7-2006, *Screw Threads: Nomenclature, Definitions, and Letter Symbols*

ASME Y14.36-1996, *Surface Texture Symbols*

ASME Y14.6-2001, *Screw Thread Representation*

ASME Y14.5M-1994 (reaffirmed 2004), *Dimensioning and Tolerancing - Engineering Drawing and Related Documentation Practices*

ASME Y14.5-2009, *Dimensioning and Tolerancing - Engineering Drawing and Related Documentation Practices*

ASME Y14.41-2003, *Digital Product Definition Data Practices*

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

ISO/IEC 9834-8:2008. *Information technology -- Open Systems Interconnection -- Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components*

ISO/IEC 11578:1996: "Information technology - Open System Interconnection - Remote Procedure Call (RPC)"

ISO/IEC Guide 99:2007 (E/F) – *International vocabulary of metrology – Basic and general concepts and associated terms* (VIM)

XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004

XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply. All terms are defined in the Part 1 document. The first group contains general QIF terms. The second group contains terms defined to describe the information in the QIF MBD application area.

4.1 General QIF terms referenced in the QIF MBD application area

The following general terms are referenced by the QIF MBD application area and defined in the QIF Part 1 document. The terms are repeated here for the convenience of the reader.

assembly (IEEE 75)
a number of parts or combination thereof that are joined together to perform a specific function and subject to disassembly without degradation of any of the parts

characteristic
a control placed on an element of a feature such as its size, location or form, which may be a specification limit, a nominal with tolerance, a feature control frame, or some other numerical or non-numerical control

entity
the basic unit of information in a file

NOTE The term applies to single items which may be individual elements of geometry, collections of annotations to form dimensions, or collections of entities to form structured entities.

geometric
related to shape information: e.g., points, curves, surfaces, and volumes

normal
perpendicular to a surface and pointing away from the surface material

normal vector
a unit vector perpendicular to a surface in 3 dimensions or a unit vector perpendicular to a curve in 2 dimensions

4.2 Terms defined for the QIF MBD application area

The following terms are introduced by the QIF MBD application area and defined in the QIF Part 1 document. The definitions are repeated here for the convenience of the reader.

4.2.1 clipping plane

a bounding plane surface which abbreviates the intended display of data to that portion which lies on one or the other side of the plane

4.2.2 control points

a set of points that determine the shape of a NURBS object

4.2.3 control polygon

a polygon formed by the control points of a NURBS object

4.2.4 generatrix

a curve to be swept to generate an extruded surface, or revolved to generate a surface of revolution

4.2.5 knot vector

an increasing sequence of real numbers which divides the parametric space of a NURBS into intervals (called spans)

NOTE The number of knots is equal to the number of control points plus the order. The knot vector is a uniform one if all knots are equally spaced and of multiplicity one.

4.2.6 trimming contour

a 3D contour which is used for trimming a surface

4.2.7 wire-frame

a method of geometric modeling in which a two- or three-dimensional object is represented by object edges

5 Symbols and Abbreviated Terms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ASME	American Society of Mechanical Engineers
BREP	Boundary Representation
CAD	Computer-Aided Design
CAIPP	Computer-Aided Inspection Process Planning
CAM	Computer-Aided Machining or Computer-Aided Manufacturing
CAX	Computer-Aided Technologies
CMM	Coordinate Measuring Machine
CoP	Cloud of Points
COTS	Commercial Off-The-Shelf

DME	Dimensional Measuring Equipment
DMIS	Dimensional Measuring Interface Standard
DMSC	Dimensional Metrology Standards Consortium
DRF	Datum Reference Frame
ERP	Enterprise Resource Planning
GD&T	Geometric Dimensioning and Tolerancing
GPS	Geometrical Product Specifications
GUID	Globally Unique Identifier
ISO	International Organization for Standardization
MBD	Model Based Definition
MES	Manufacturing Execution Systems
MRI	Measurement Resources Information
MRP	Materials Resource Planning
MSA	Measurement Systems Analysis
NURBS	Non-uniform Rational Basis Spline
PDPMI	Product Definition with Product Manufacturing Information
PMI	Product Manufacturing Information
QIF	Quality Information Framework
QMS	Quality Measurement Standards (a DMSC committee)
QPId	QIF Persistent Identifier
R&R	Repeatability and Reproducibility
SI	The International Systems of Units
SPC	Statistical Process Control
SQC	Statistical Quality Control
STEP	STandard for the Exchange of Product model data (ISO 10303)
UUID	Universally Unique Identifier

XML	eXtensible Markup Language
XSDL	XML Schema Definition Language

6 QIF MBD information model requirements

All CAD systems generate models of the nominal shape of products. Designers and/or manufacturing quality experts add information to the nominal shape information to instruct downstream manufacturing quality functions how to perform actions. This combination of information is called Product Definition with Product Manufacturing Information (PDPMI).

The basic requirement of QIF MBD is that it must be able to represent PDPMI. The QIF MBD model must be able to capture the product definition information coming from a CAX authoring system. The QIF MBD model must also be able to capture the PMI that comes from the CAX authoring system.

A second requirement is that the QIF MBD model must represent PMI in a form that can be consumed by downstream application software. That is, the representation must be able to graphically present the PMI. The representation must also enable processing the PMI without graphical presentation of it. Being able to display the PMI is essential for human use. Being able to use the PMI without graphical presentation is essential for computer use.

Many products are assemblies of parts. A third requirement of the QIF MBD model is that it be able to represent assemblies.

The representation of PMI used in QIF MBD is described in Part 1 and Part 3 of the standard as well as in the XML schema files containing the formal model. This Part of the standard focuses on the representation of the definition of the nominal shape of a product and the representation of assemblies.

7 Overview of the product data model

7.1 Design principles

QIF defines a complete and unambiguous model structure implementing a clear and efficient way for representing simple parts and complex assemblies.

QIF product definition implements Boundary Representation (BREP) — a method for defining shapes using their boundaries. A distinctive characteristic of Boundary Representation is the clear separation of the shape definition data in the following two types: topology and geometry. Topology defines the relationship between the geometric entities that can be linked together to produce complex shapes. In contrast to the geometry, topology does not consider the metric properties of objects. Geometry defines the shape of the model entities, independent of their topological relationships.

The QIF data model supports two modeling concepts – exact modeling (with use of one global model tolerance) and tolerant modeling (the model tolerance can vary from one entity to another). In both cases, the tolerance concept applies to the numerical accuracy of numbers in

the file. This is distinct from the tolerance concept in PMI, which applies to manufacturing tolerances.

QIF geometric objects can be categorized in different ways. The most natural way is to separate objects by dimension: zero-dimensional (points), one-dimensional (curves) and two-dimensional (surfaces) objects.

In order to avoid any potential interoperability issues, the QIF data model defines all geometric entities explicitly. No entities such as blends, which can be interpreted and implemented differently by different systems, are allowed.

QIF supports a hybrid model representation allowing data of different natures within one model. This includes items such as parametric, mesh, and point cloud data.

For any model entity, a variety of traditional CAD attributes can be specified (such as: color, label, layer, visibility status, etc.) with possible arbitrary extensions implemented as user-defined attributes.

7.1.1 Binary data blocks

To store binary data blocks in an xml file the Base-64 encoding is used; it is implemented as one of the standard xml types, xs:base64Binary.

In QIF there are two application areas where the binary data can be used.

The first use is one type of user-defined attributes (AttributeUser) - BinaryDataType.

The second use of binary data is for providing an alternative, more efficient and compact way of storing large amounts of simple-type data such as 3D points. This noticeably reduces file size and significantly increases file parsing speed. These two factors can be crucial in working with such objects as point clouds or tessellated data.

The BinaryDataType represents a Base64-encoded arbitrary binary data block. For base64Binary data, the entire binary stream is encoded using the Base64 Alphabet in RFC 2045.

Fields of BinaryDataType:

Field Name	Data Type	Description
<body>	xs:base64Binary	The binary data encoded in the Base-64 format.
@N	xs:unsignedInt	The N attribute shows the size of the binary block in bytes.

Example:

```
<Edge id="459" Parent="453">
  <Attributes Number="3">
    <!-- A user defined attribute STD02 - 14 bytes of binary data -->
```



```
    <AttributeUser Name="STD02" NameUserAttribute="T1">
      <UserDataBinary N="14">
        c2FtcGxlIHN0cmluZwA=
      </UserDataBinary>
    </AttributeUser>
  </Attributes>
  ...
</Edge>
```

The fields where the binary alternatives can be used are shown in the table below. The corresponding non-binary field names are the same but without the "Binary" suffix.

Field Name (binary array)	Array Element Type
Nurbs12CoreType/CPsBinary	2D point (pair of doubles)
Nurbs13CoreType/CPsBinary	3D point (triplet of doubles)
Polyline12CoreType/PointsBinary	2D point (pair of doubles)
Polyline13CoreType/PointsBinary	3D point (triplet of doubles)
PathTriangulationCoreType/EdgesBinary	Pair of integers
MeshTriangleCoreType/TrianglesBinary	Triplet of integers
MeshTriangleCoreType/NeighboursBinary	Triplet of integers
MeshTriangleCoreType/VerticesBinary	3D point (triplet of doubles)
MeshTriangleCoreType/NormalsBinary	3D unit vector (triplet of doubles)
FaceMeshType/TrianglesBinary	Unsigned integer
FaceMeshType/TrianglesVisibleBinary	Unsigned integer
FaceMeshType/TrianglesHiddenBinary	Unsigned integer
FaceMeshType/TrianglesColorBinary	Triplet of unsigned bytes
PointCloudType/PointsBinary	3D point (triplet of doubles)
PointCloudType/NormalsBinary	3D unit vector (triplet of doubles)

The binary representation:

Array Element Type	Binary Representation	Bytes	Endianness
double	IEEE 754-2008 binary64 double precision floating point format.	8	Little-endian
integer	32 bit integer	4	Little-endian
unsigned integer	32 bit unsigned integer	4	Little-endian
unsigned byte	8 bit unsigned integer	1	Little-endian

The ArrayBinaryType represents an array of Base64-encoded binary elements. For data the entire binary stream is encoded using the Base64 Alphabet in RFC 2045.

Fields of ArrayBinaryType:

Field Name	Data Type	Description
<body>	xs:base64Binary	Binary data encoded in Base-64 format.
@N	xs:unsignedInt	The N attribute shows how many elements are present in this array.
@sizeElement	xs:unsignedInt	The sizeElement attribute shows the size (in bytes) of one element stored in the array. The total size of the binary array can be calculated as: N*sizeElement.

Example:

```
<!-- Binary -->
<PointCloud id="3">
  <PointsBinary N="31" sizeElement="24">
    ANwUZdaTPcD9///9B/QzwCx7CR2wKypAZHlAwox0PcAFAADwH+gzwAzaS+jAVipAwMzMQlZV
    PcAFAADKR9wzwBsAABAzgSpAsZfQ3jI2PcAJAACAf9AzwJvQXkIHqypAfJtijiIXPcD+//8F
    x8QzwHkvoS0+1CpAo5mZSSX4PMAMAABQHrkzwPf//3/Y/CpAalOMCDvZPMAEAABSha0zwAEm
    tOfWJCtAbYpRw2O6PMAOAAAA/KEzwKOE9hI6TCtA9P//cZ+bPMD7//9NgpYzwP7//68CcytA
    mXWuDO58PMDy//8vGIszwFh7CW0xmStAiKxzi09ePMD+//+ZvX8zwCvaS/jGvitAWmZm5sM/
    PMAMAACAcnQzwPL////D4ytAdGSdFUshPMAMAADWNmkzwJnQXjIpCCxAVGgvEeUCPMAIAACQ
    Cl4zwGMvoT33KyxAJTMz0ZHkO8AKAACi7VIzwAMAANAuTyxAn4a/TVHGO8AAAAAA4EczwPUl
    tJfQcSxACCTrfiOoO8D2//+d4TwzwLSE9kLdkyxAw8zMXAiKO8D1//9v8jEzwPT//39VtSxA
    a0J73/9rO8AIAABqEiczwC97Cf051ixALEYN/wlOO8ACAACAQRwzwBnaS2iL9ixAoJmZsyYw
    O8AIAACmfxEzwPX//29KFi1ADv429VUSO8AKAADQzAYzwK/QXsJ3NS1AEjX8u5f0OsD2///x
    KPwywFEvoQ0UVC1ADgAAAOzWOsAOAAAAlPEywAAAAAAgci1ASSBZuVK5OsAJAADuDecywP4l
    tEecjy1AQ1ce4MubOsANAACwltwywMeE9pKJrC1AXmZmbFd+OsAIAAA6LtIywA8AAJDoyC1A
    NQ9IVvVgOsAFAACA1McywE97Ce255C1A8xLalaVDOsDz//91ib0ywAXaS1j+/y1ANDMzI2gm
    OsD4//8PTbMywBoAAIC2Gi5AWzFq9jwJOsACAABCH6kywJvQXhLjNC5A
  </PointsBinary>
</PointCloud>

<!-- Text -->
<PointCloud id="3">
  <Points N="31">
    -29.5774901557852 -19.9532469511032 13.0853280138086
    -29.4552728087814 -19.9067373275757 13.1694405167191
    -29.3333474874496 -19.8604704141617 13.2523427009583
    -29.211713720251 -19.8144454956055 13.3340397587529
    -29.0903710356465 -19.7686618566513 13.4145368823299
    -28.9693189620972 -19.7231187820435 13.493839263916
    -28.848557028064 -19.6778155565262 13.5719520957382
    -28.7280847620081 -19.6327514648438 13.6488805700231
    -28.6079016923904 -19.5879257917404 13.7246298789978
    -28.4880073476721 -19.5433378219604 13.799205214889
    -28.368401256314 -19.4989868402481 13.8726117699235
    -28.2490829467773 -19.4548721313477 13.9448547363281
    -28.130051947523 -19.4109929800034 14.0159393063298
    -28.0113077870122 -19.3673486709595 14.0858706721553
    -27.8928499937057 -19.3239384889603 14.1546540260315
    -27.7746780960648 -19.28076171875 14.2222945601852
```

```
    -27.6567916225504 -19.2378176450729 14.2887974668432
    -27.5391901016235 -19.1951055526733 14.3541679382324
    -27.4218730617453 -19.1526247262955 14.4184111665796
    -27.3048400313766 -19.1103744506836 14.4815323441117
    -27.1880905389786 -19.068354010582 14.5435366630554
    -27.0716241130122 -19.0265626907349 14.6044293156377
    -26.9554402819386 -18.9849997758865 14.6642154940852
    -26.8395385742188 -18.9436645507813 14.722900390625
    -26.7239185183137 -18.9025563001633 14.7804891974838
    -26.6085796426844 -18.8616743087769 14.8369871068884
    -26.4935214757919 -18.8210178613663 14.8923993110657
    -26.3787435460974 -18.7805862426758 14.9467310022425
    -26.2642453820617 -18.7403787374496 14.9999873726456
    -26.150026512146 -18.7003946304321 15.052173614502
    -26.0360864648113 -18.6606332063675 15.1032949200383
  </Points>
</PointCloud>
```

7.1.2 Entity attributes

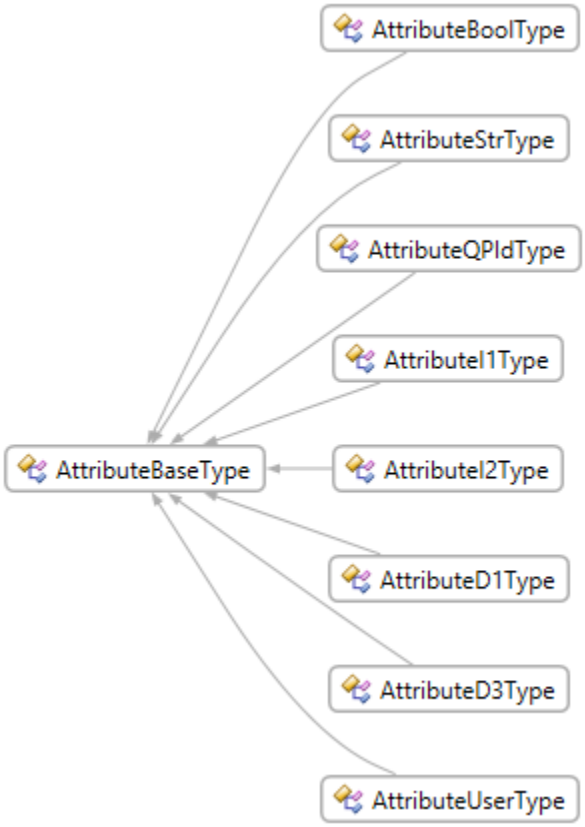


Figure 2 – Entity attributes

All entity attributes contain the following fields:

Field Name	Data Type	Description
@name	xs:string	The name of the entity attribute. This name is a unique identifier of an attribute within the entity.

7.1.2.1 Boolean entity attribute

The AttributeBool describes an entity attribute of Boolean type.

Fields:

Field Name	Data Type	Description
Value	xs:boolean	The Boolean value.

Example:

```
<AttributeBool name="bool_attr" value="1"/>
```

7.1.2.2 String entity attribute

The AttributeStr describes an entity attribute of string type.

Fields:

Field Name	Data Type	Description
@value	xs:string	The string value.

Example:

```
<AttributeStr name="str_attr" value="Value of string attribute"/>
```

7.1.2.3 Integer entity attribute

The AttributeI1 describes an entity attribute of integer type.

Fields:

Field Name	Data Type	Description
@value	xs:integer	The integer value.

Example:

```
<AttributeI1 name="int_attr" value="1987"/>
```

7.1.2.4 Pair of integers entity attribute

The AttributeI2 describes an entity attribute of ‘pair of integers’ type.

Fields:

Field Name	Data Type	Description
@value	I2Type	The pair of integer values.

Example:

```
<AttributeI2 name="I2_attr" value="223 -34"/>
```

7.1.2.5 Double entity attribute

The AttributeD1 describes an entity attribute of double type.

Fields:

Field Name	Data Type	Description
@value	xs:double	The double value.

Example:

```
<AttributeD1 name="D1_attr" value="-12.230243"/>
```

7.1.2.6 Triplet of doubles entity attribute

The AttributeD3 describes an entity attribute of ‘triplet of doubles’ type.

Fields:

Field Name	Data Type	Description
@value	D3Type	The triplet of double values.

Example:

```
<AttributeD3 name="D3_attr" value="23.434 -0.927 37.321"/>
```

7.1.2.7 Custom entity attribute

The AttributeUser defines a user-defined entity attribute which contains a binary array or any user-defined structured XML data.

Fields:

Field Name	Data Type	Description
@nameUserAttribute	xs:string	The name of user-defined attribute type. The structure of the user-defined attribute can be identified by this name.
UserDataXML or UserDataBinary	xs:any or BinaryDataType	UserDataXML element: each element is a user-defined XML element from any namespace that is not the target namespace. The XML processor will validate elements if the corresponding schema is present, if the schema is not present, no errors will occur. or UserDataBinary element: a binary block of user data.

Example 1: attribute with user defined XML content:

```
<AttributeUser name="user_attr"
nameUserAttribute="user_type_structure">
  <UserDataXML>
    <Structure xmlns="http://userschema.org">
      <TextField>Some text</TextField>
      <IntegerField>2332</IntegerField>
    </Structure>
  </UserDataXML>
</AttributeUser>
```

Example 1: user defined XML schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://userschema.org"
  targetNamespace="http://userschema.org"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="1">

  <xs:complexType name="StructureType">
    <xs:sequence>
      <xs:element name="TextField" type="xs:string">
      </xs:element>
      <xs:element name="IntegerField" type="xs:integer">
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="Structure" type="StructureType"/>

</xs:schema>
```

Example2: attribute with binary content:

```
<AttributeUser name="user_attr" nameUserAttribute="user_type_image">
  <UserDataBinary N="240">
    iVBORw0KGgoAAAANSUhEUgAAACAAAAAUCAIAAABj86gYAAAABmJLR0QAAAAAAAD5Q7t/AAAA
    CXBIWXMAAA7EAAAOxAGVKw4bAAAAkElEQVR4nO2RyxGAIAwFqYd6qId6qId6PChMIDFBPqPx
    xJ4QHlkTzKGMGQXesgVDfhIEZyrWx+d49LYfEFQBXFuvQME6FS5ohqZZFaR2ABfYaV54R8ZK
    5lzSDQEZUboASdylgnKMwV4Hwp6K4p9cifxx74A3PTWigrqAj4gN4QMBfeR+LVzDrMUjK7IF
    Q9QFJ99Bg+O37n7tAAAAAElFTkSuQmCC
  </UserDataBinary>
</AttributeUser>
```


7.2 Geometry

QIF geometry defines the shape of the model entities and includes the subtypes shown in Figure 3.

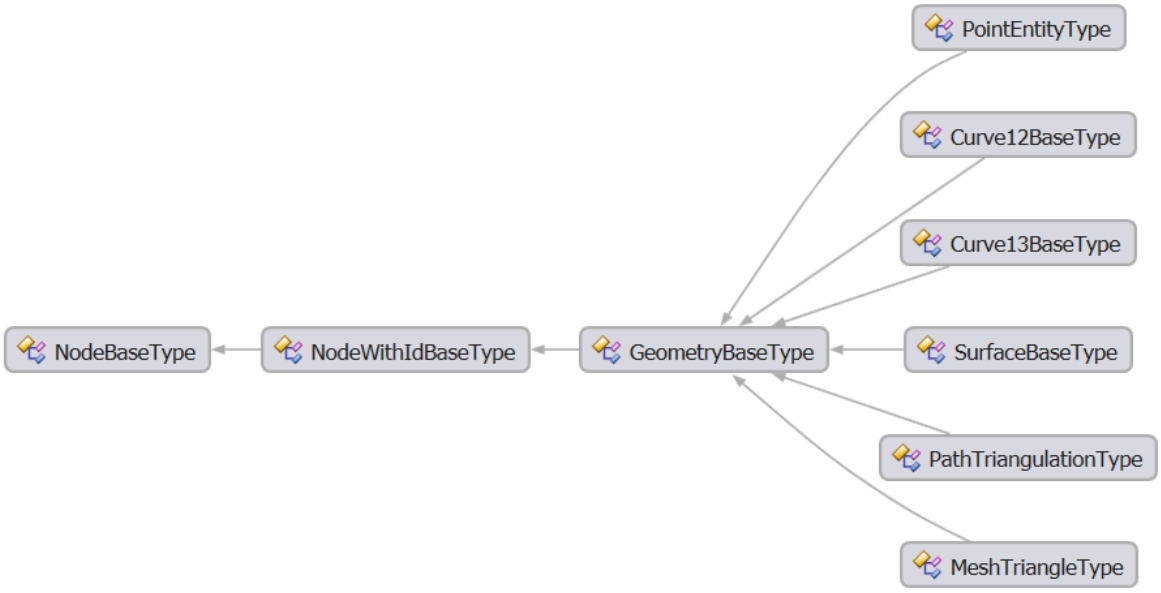


Figure 3 – Geometry Types

All geometric entities contain the following fields:

Field Name	Data Type	Description
@id	QIFIdType	The unique model entity identifier.
@label	xs:string	The model entity "nameplate". Normally it can be seen at the entity item in the model tree.
Attributes	AttributesType	User defined attributes (typified, binary array, or XML structured).

7.2.1 Point

The Point describes a point in 3D space. It is normally used as underlying geometry for vertices.

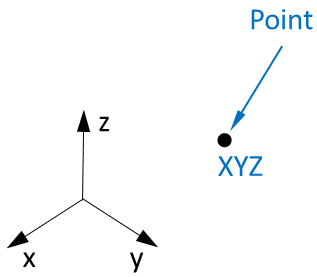


Figure 4 – Point

Fields:

Field Name	Data Type	Description
XYZ	PointSimpleType	The Cartesian three-dimensional coordinates of the 3D point.

Example:

```
<Point id="200">
  <XYZ>-0.2032 -0.0666 -0.0684</XYZ>
</Point>
```


7.2.2 2D Curves

This chapter describes curves defined in 2D space, curve(t):R1 → R2, where

- R1 is curve parameter space - the 1D real number space, where each point is defined as a single coordinate (t)
- R2 is the two-dimensional Euclidean space, where each point is defined as a coordinate pair (u, v).

The correspondence between 1D parameter space and 2D Euclidean space is illustrated in Figure 5.

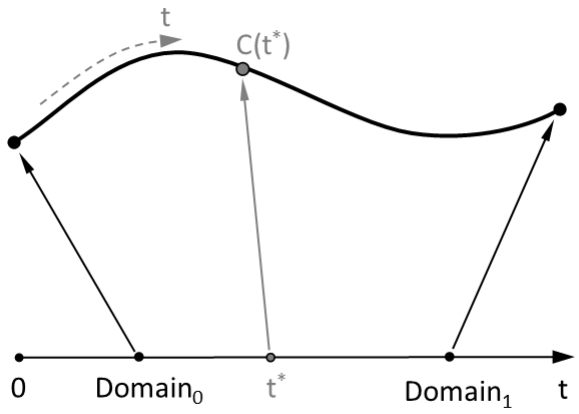


Figure 5 – 2D Parametric Curve

All types of 2D curves are derived from Curve12BaseType as shown in Figure 6.

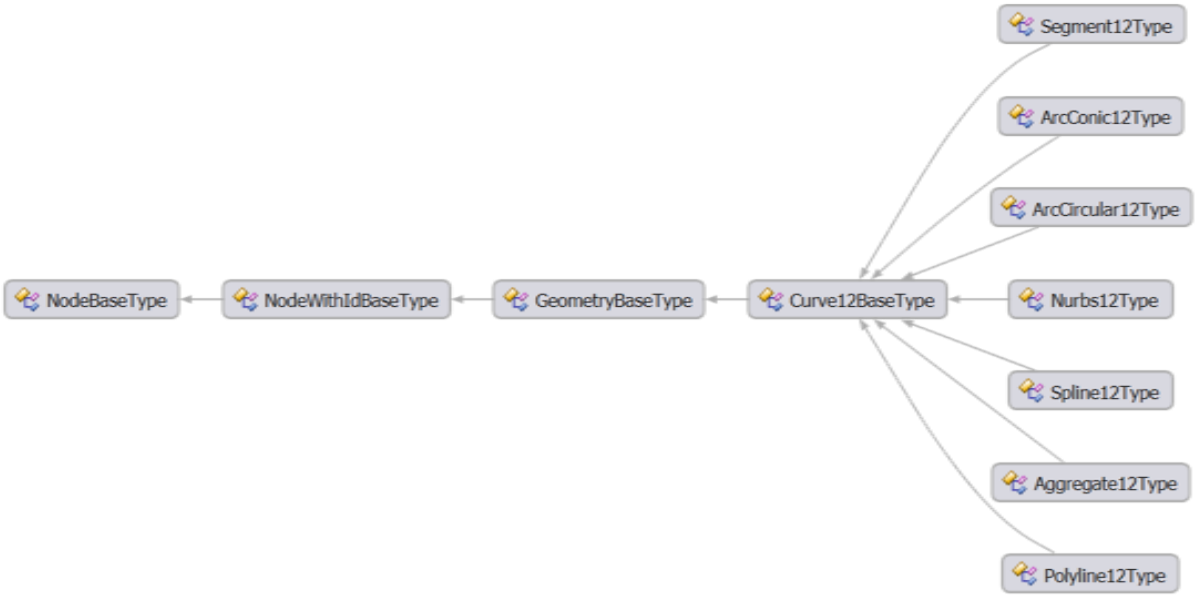


Figure 6 – 2D Curves Types

All 2D geometric curves are represented using a geometric core and a "wrapper" that includes an id for referencing.

The domain of every 2D curve is given in the domain attribute of the core, which is required.

The turned attribute of 2D curves is optional. If it does not appear, its default value is false.

2D curves are normally used to define a trimming curve in the parametric space of a surface.

All 2D curves contain the following fields:

Field Name	Data Type	Description
{Curve}/{Curve}Core	{Curve}CoreType	The curve core.
{Curve}/{Curve}Core/@domain	ParameterRangeType	The domain of the parameter space of the curve.

7.2.2.1 Segment

Segment12 describes a linear segment in 2D space as shown in Figure 7.

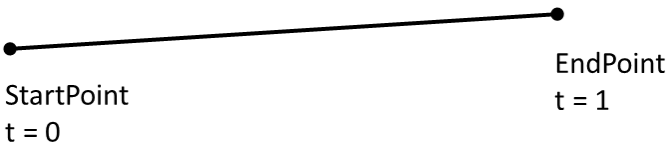


Figure 7 – 2D Segment

A linear segment is defined and positioned in 2D space with start and end points.

Function:

$$Segment12(t)\colon R_1 \rightarrow R_2$$

$$Segment12(t) = StartPoint + t(EndPoint - StartPoint), \qquad t \in [0,1]$$

Fields:

Field Name	Data Type	Description
Segment12Core/StartPoint	Point2dSimpleType	The beginning point of the linear segment.
Segment12Core/EndPoint	Point2dSimpleType	The end point of the linear segment.

Example:

```
<Segment12 id="350">
   <Segment12Core domain="0 1">
      <StartPoint>10.1 -2431.7</StartPoint>
      <EndPoint>34.1 4542.4</EndPoint>
   </Segment12Core>
```



```
</Segment12>
```

7.2.2.2 Polyline

Polyline12 describes a polyline in 2D space as shown in Figure 8.

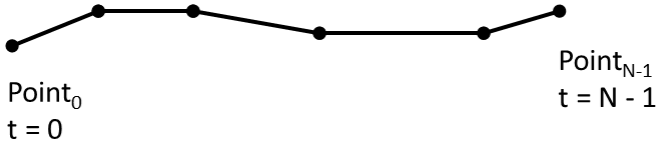


Figure 8 – 2D Polyline

A polyline is defined as a series of connected linear segments.

Function:

$$Polyline12(t)\colon R_1 \rightarrow R_2$$

$$Polyline12(t) = Point_i + (t - i)(Point_{i+1} - Point_i)$$

$$t \in [i, i+1], \qquad i \in [0, N-2], \qquad N - number\ of\ points$$

Fields:

Field Name	Data Type	Description
Polyline12Core/Points or Polyline12Core/PointsBinary	ArrayPoint2dType or ArrayBinaryType	The array of 2D polyline points.

Example:

```
<Polyline12 id="32">
  <Polyline12Core domain="0 4">
    <Points N="5">
      1.2 11.5
      4.1 13.5
      7.3 16.2
      8.9 23.5
      9.2 29.8
    </Points>
  </Polyline12Core>
</Polyline12>
```

7.2.2.3 Circular Arc Curve

ArcCircular12 describes a circular arc in 2D space as shown in Figure 9 and Figure 10.

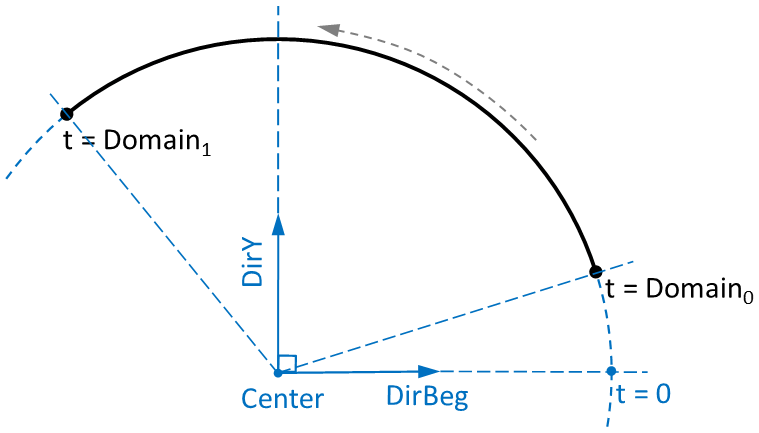


Figure 9 – 2D Circular Arc

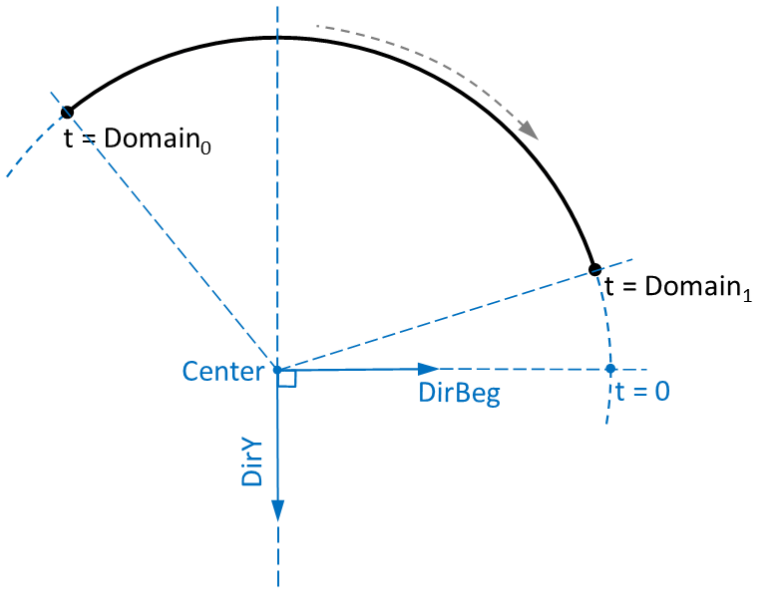


Figure 10 – 2D Circular Arc (turned)

A circular arc is defined as a portion of a circle.

Function:

$$ArcCircular12(t)\colon R_1 \rightarrow R_2$$

$$ArcCircular12(t) = Center + Radius(\cos(t)\, DirBeg + \sin(t)\, DirY)$$

$$DirY = \begin{cases} \begin{bmatrix} -DirBeg_y \\ DirBeg_x \end{bmatrix}, & turned = false \\ \begin{bmatrix} DirBeg_y \\ -DirBeg_x \end{bmatrix}, & turned = true \end{cases}$$

where t is in radians, $t \in (-\infty, +\infty)$

Fields:

Field Name	Data Type	Description
ArcCircular12Core/@turned	xs:boolean	This flag shows if the curve is turned.
ArcCircular12Core/Radius	xs:double	The arc radius.
ArcCircular12Core/Center	Point2dSimpleType	The center of arc.
ArcCircular12Core/DirBeg	UnitVector2dSimpleType	The unit vector representing the beginning of the circular arc.

Example:

```
<ArcCircular12 id="10">
  <ArcCircular12Core domain="0 3.1415926">
    <Radius>10.2</Radius>
    <Center>1.4 6.7</Center>
    <DirBeg>0.0 1.0</DirBeg>
  </ArcCircular12Core>
</ArcCircular12>
```

7.2.2.4 Conic Arc Curve

ArcConic12 describes a conic arc in 2D space as shown in Figure 11 through Figure 16.

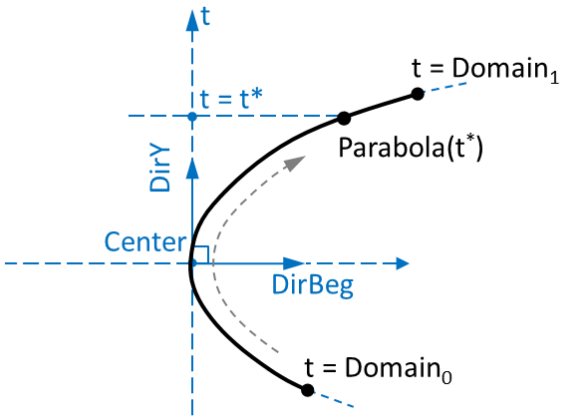


Figure 11 – 2D Conic Arc (form = PARABOLA)

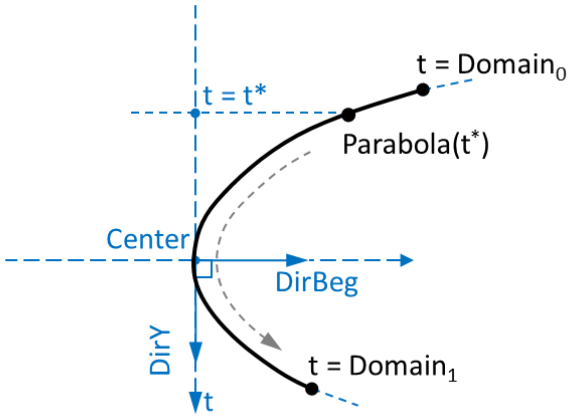


Figure 12 – 2D Conic Arc (form = PARABOLA, turned = true)

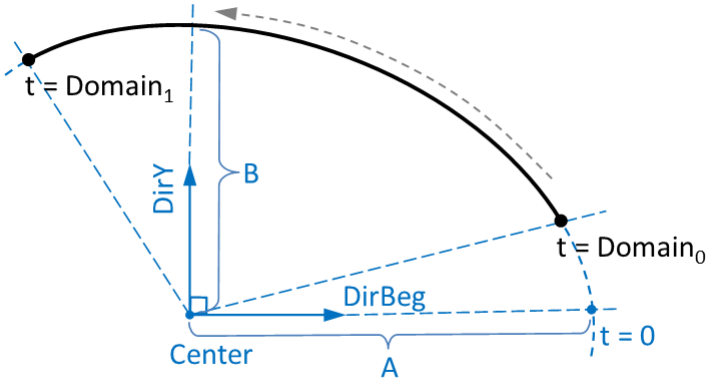


Figure 13 – 2D Conic Arc (form = ELLIPSE)

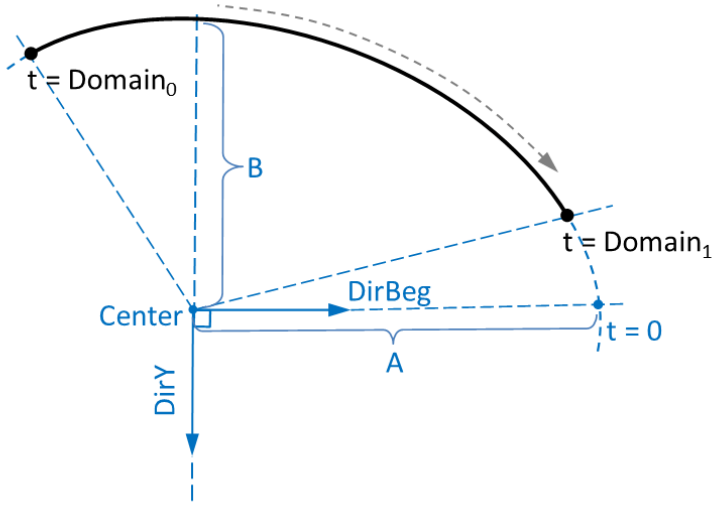


Figure 14 – 2D Conic Arc (form = ELLIPSE, turned = true)

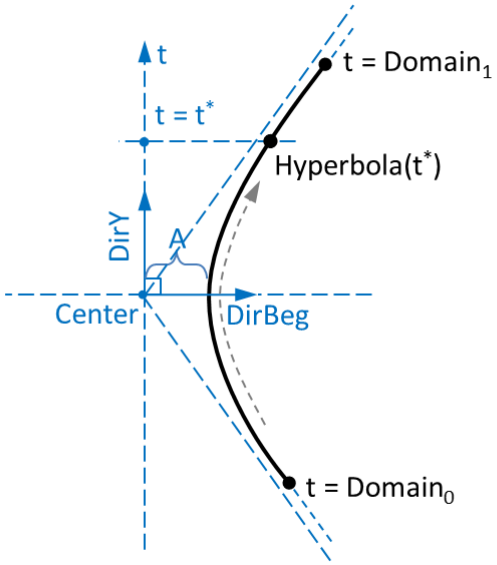


Figure 15 – 2D Conic Arc (form = HYPERBOLA)

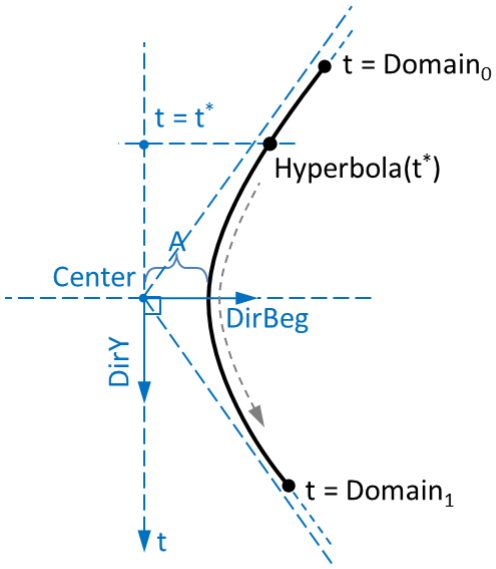


Figure 16 – 2D Conic Arc (form = HYPERBOLA, turned = true)

A conic arc is defined as a portion of a conic curve. The conic curve can have one the following forms: an ellipse, a parabola, or a hyperbola.

Function:

$$ArcConic12(t)\colon R_1 \rightarrow R_2$$

$$ArcConic12(t) = Center + x(t)DirBeg + y(t)DirY$$

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{cases} \begin{bmatrix} At \\ Bt^2 \end{bmatrix}, & form = PARABOLA \\ \begin{bmatrix} A\cos(t) \\ B\sin(t) \end{bmatrix}, & form = ELLIPSE \\ \begin{bmatrix} A\sqrt{1 + t^2/B^2} \\ t \end{bmatrix}, & form = HYPERBOLA \end{cases}$$

$$DirY = \begin{cases} \begin{bmatrix} -DirBeg_y \\ DirBeg_x \end{bmatrix}, & turned = false \\ \begin{bmatrix} DirBeg_y \\ -DirBeg_x \end{bmatrix}, & turned = true \end{cases}$$

$$t \in (-\infty, +\infty)$$

Fields:

Field Name	Data Type	Description
ArcConic12Core/@form	ArcConicFormEnumType	The form of the conic arc.

ArcConic12Core/@turned	xs:boolean	This flag shows if the curve is turned.
ArcConic12Core/A	xs:double	Ellipse: the major radius of the arc. Parabola: the coefficient “A”. Hyperbola: the semi-axis (the distance from the center to the extreme point).
ArcConic12Core/B	xs:double	Ellipse: the minor radius of the arc. Parabola: the coefficient “B”. Hyperbola: the implicit parameter.
ArcConic12Core/Center	Point2dSimpleType	The center point.
ArcConic12Core/DirBeg	UnitVector2dSimpleType	The unit vector representing the beginning of the conic arc.

Example:

```
<ArcConic12 id=”123”>
  <ArcConic12Core form=”PARABOLA” domain=”-0.2 3.5”>
    <A>4.5</A>
    <B>3.1</B>
    <Center>13.3 -43.2</Center>
    <DirBeg>1.0 0.0</DirBeg>
  </ArcConic12Core>
</ArcConic12>

<ArcConic12 id=”123”>
  <ArcConic12Core form=”ELLIPSE” domain=”10.3 18.9” turned=”1”>
    <A>14.2</A>
    <B>33.1</B>
    <Center>63.7 -34.0</Center>
    <DirBeg>1.0 0.0</DirBeg>
  </ArcConic12Core>
</ArcConic12>

<ArcConic12 id=”123”>
  <ArcConic12Core form=”HYPERBOLA” domain=”1.2 7.0”>
    <A>14.5</A>
    <B>30.1</B>
    <Center>0.3 3.2</Center>
    <DirBeg>1.0 0.0</DirBeg>
  </ArcConic12Core>
</ArcConic12>
```

7.2.2.5 Spline Curve

Spline12 describes a spline curve in 2D space as shown in Figure 17.

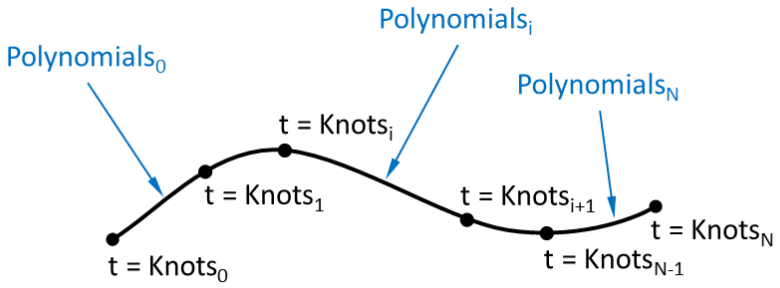


Figure 17 – 2D Spline Curve

A spline curve is defined as a sequence of parametric polynomial segments.

Function:

$$Spline12(t)\colon R_1 \to R_2$$

$$Spline12(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{Degree_p} (C_i^p)_x (t_p)^i \\ \sum_{i=0}^{Degree_p} (C_i^p)_y (t_p)^i \end{bmatrix}$$

$$N - number\ of\ polynomials$$

$$N = number\ of\ knots - 1$$

$$Coefficients = \{C^0, \dots, C^{N-1}\}$$

$$p - index\ of\ polynomials$$

$$C^p = \{c_0 \dots c_{Degree_p}\}, \quad Degree_p = Orders_p - 1$$

$$t_p - parameter\ of\ polynomials\ p$$

$$t_p \in \begin{cases} [Knots_p, Knots_{p+1}], & normalized = false \\ [0,1], & normalized = true \end{cases}$$

$$t_p = \begin{cases} t - Knots_p, & Normalized = false \\ (t - Knots_p)/(Knots_{p+1} - Knots_p), & Normalized = true \end{cases}$$

$$t \in [Knots_p, Knots_{p+1}], \qquad p \in [0, N-1]$$

Fields:

Field Name	Data Type	Description
Spline12Core/@normalized	xs:boolean	This flag shows if the curve is normalized.
Spline12Core/Knots	ArrayDoubleType	The spline breakpoints.
Spline12Core/Orders	ArrayNaturalType	The orders of the polynomial segments. The order is 'the degree of the polynomial' + 1. The size of this array is 'the number of spline breakpoints' – 1.
Spline12Core/Coefficients	ArrayPoint2dType	The coefficients of the polynomial segments are pairs. For each segment the number of coefficients is equal to the order on this segment. The total size of this array is the sum of all orders.

Example:

```
<Spline12 id="302">
  <Spline12Core domain="0 2">
    <Knots N="3">
      0 1 2
    </Knots>
    <Orders N="2">
      4 4
    </Orders>
    <Coefficients N="8">
      87.5 237.5
      14.0625 -7.8125
      0 0
      -1.5625 1.5625
      100 231.25
      9.375 -3.125
      -4.6875 4.6875
      1.5625 -1.5625
    </Coefficients>
  </Spline12Core>
</Spline12>
```

7.2.2.6 NURBS Curve

Nurbs12 describes a NURBS curve in 2D space as shown in Figure 18.

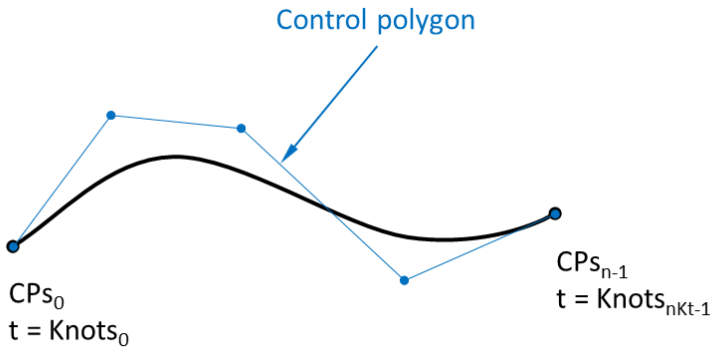


Figure 18 – 2D NURBS Curve

A NURBS curve is a freeform curve built on the B-spline basis functions and defined by its order (= degree + 1), a knot vector (an increasing sequence of real numbers which divides the parametric space in the intervals called knot spans), and an array of control points with an optional set of associated weights (positive real numbers). If the weights are not defined or equal, the curve is a polynomial one (otherwise rational).

Function:

$$Nurbs12(t)\colon R_1 \to R_2$$

$$Nurbs12(t) = \frac{\sum_{i=0}^{n-1} N_{i,Degree}\,(t) Weights_i CPs_i}{\sum_{i=0}^{n-1} N_{i,Degree}\,(t) Weights_i}$$

$$Degree = Order - 1$$

$$N_{i,j}(t) - the\ bspline\ basis\ functions$$

$$N_{i,0}(t) = \begin{cases} 1, & t \in [Knots_i, Knots_{i+1}) \\ 0, & otherwise \end{cases}$$

$$N_{i,p}(t) = \frac{t - Knots_i}{Knots_{i+p} - Knots_i} N_{i,p-1}(t) + \frac{Knots_{i+p+1} - t}{Knots_{i+p+1} - Knots_{i+1}} N_{i+1,p-1}(t)$$

$$t \in [Knots_0, Knots_{nKt-1}], \qquad nKt - number\ of\ knots$$

$$nKt = n + Order, \qquad n - number\ of\ control\ points$$

Fields:

Field Name	Data Type	Description
Nurbs12Core/Order	NaturalType	The order (= degree + 1).
Nurbs12Core/Knots	ArrayDoubleType	The knot vector is an increasing sequence of real numbers. The size of the knot vector is “number of control points” + “order”.
Nurbs12Core/CPs or Nurbs12Core/CpsBinary	ArrayPoint2dType or ArrayBinaryType	The array of 2D control points.
Nurbs13Core/Weights	ArrayDoubleType	The array of weights associated with control points (positive real numbers).

Example:

```
<Nurbs12 id=”607”>
  <Nurbs12Core domain=”0 1”>
    <Order>3</Order>
    <Knots N=”10”>
       0 0 0 0.33333333 0.33333333 0.66666667 0.66666667 1 1 1
    </Knots>
    <CPs N=”7”>
      -386.61 -44.00
      -386.61 -54.39
      -377.61 -49.19
      -368.61 -44.00
      -377.61 -38.80
      -386.61 -33.60
      -386.61 -44.00
    </CPs>
    <Weights N=”7”>
       1 0.5 1 0.5 1 0.5 1
    </Weights>
  </Nurbs12Core>
</Nurbs12>
```

7.2.2.7 Aggregate Curve

Aggregate12 describes an aggregate curve in 2D space as shown in Figure 19.

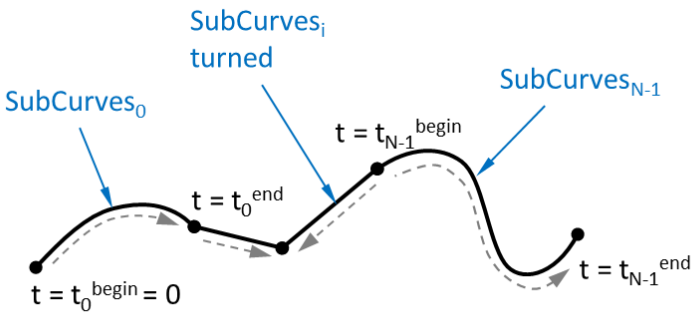


Figure 19 – 2D Aggregate Curve

An aggregate curve is a continuous curve which is defined as a sequence of oriented parametric sub-curves.

Function:

$$Aggregate12(t)\colon R_1 \to R_2$$

$$Aggregate12(t) = SubCurves_i(t_i)$$

$$t \in \left[t_i^{begin}, t_i^{end}\right], \qquad i \in [0, N-1], \qquad N - number\ of\ subcurves$$

$$Domain_i^j := Domain_j\ of\ SubCurves_i, \qquad j \in \{0,1\}$$

$$t_i^{begin} = \sum_{j<i}\left(domain_j^1 - domain_j^0\right)$$

$$t_i^{end} = t_i^{begin} + \left(domain_i^1 - domain_i^0\right)$$

$$t_i = \begin{cases} domain_i^0 + \left(t - t_i^{begin}\right), & SubCurves_i\ is\ not\ turned \\ domain_i^1 - \left(t - t_i^{begin}\right), & SubCurves_i\ is\ turned \end{cases}$$

Fields:

Field Name	Data Type	Description
Aggregate12Core/SubCurves	ArraySubCurve12Type	The array of oriented sub-curves.
Aggregate12Core/SubCurves/SubCurve/@turned	xs:boolean	The orientation of sub-curve.

Example:


```
<Aggregate12 id="419">
  <Aggregate12Core domain="0 4">
    <SubCurves N="4">
      <SubCurve turned="1">
        <Segment12Core domain="0 1">
          <StartPoint>-367.36 -214.75</StartPoint>
          <EndPoint>-367.36 -207.50</EndPoint>
        </Segment13Core>
      </SubCurve>
      <SubCurve>
        <Nurbs12Core domain="0 1">
          <Order>3</Order>
          <Knots N="6">
            0 0 0 1 1 1
          </Knots>
          <CPs N="3">
            -367.36 -214.75
            -366.61 -214.75
            -366.61 -214.75
          </CPs>
          <Weights N="3">
            1 0.70710678 1
          </Weights>
        </Nurbs12Core>
      </SubCurve>
      <SubCurve>
        <Segment12Core domain="0 1">
          <StartPoint>-366.61 -214.75</StartPoint>
          <EndPoint>-366.61 -207.50</EndPoint>
        </Segment12Core>
      </SubCurve>
      <SubCurve>
        <Nurbs12Core domain="0 1">
          <Order>4</Order>
          <Knots N="10">
            0 0 0 0 0.91406368 0.91406368 1 1 1 1
          </Knots>
          <CPs N="6">
            -366.61 -207.50
            -366.61 -207.50
            -366.88 -207.50
            -367.29 -207.50
            -367.33 -207.50
            -367.36 -207.50
           </CPs>
        </Nurbs12Core>
      </SubCurve>
    </SubCurves>
  </Aggregate12Core>
</Aggregate12>
```

7.2.3 3D Curves

This chapter describes curves defined in 3D Model Space, curve(t):R1→ R3, where

- R1 is curve parameter space – the 1D real number space, where each point is defined as a single coordinate (t)
- R3 is model space – the three-dimensional Euclidean space, where each point is defined as a coordinate triple (x, y, z) specified in a right-handed Cartesian coordinate system

The correspondence between 1D parameter space and 3D Euclidean space is illustrated in Figure 20.

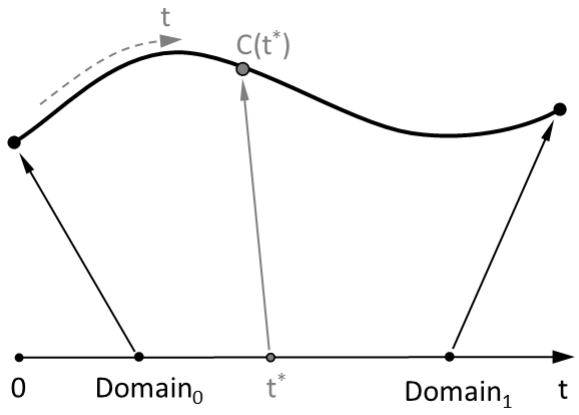


Figure 20 – 3D Parametric Curve

All types of model space curves are derived from Curve13BaseType as shown in Figure 21.

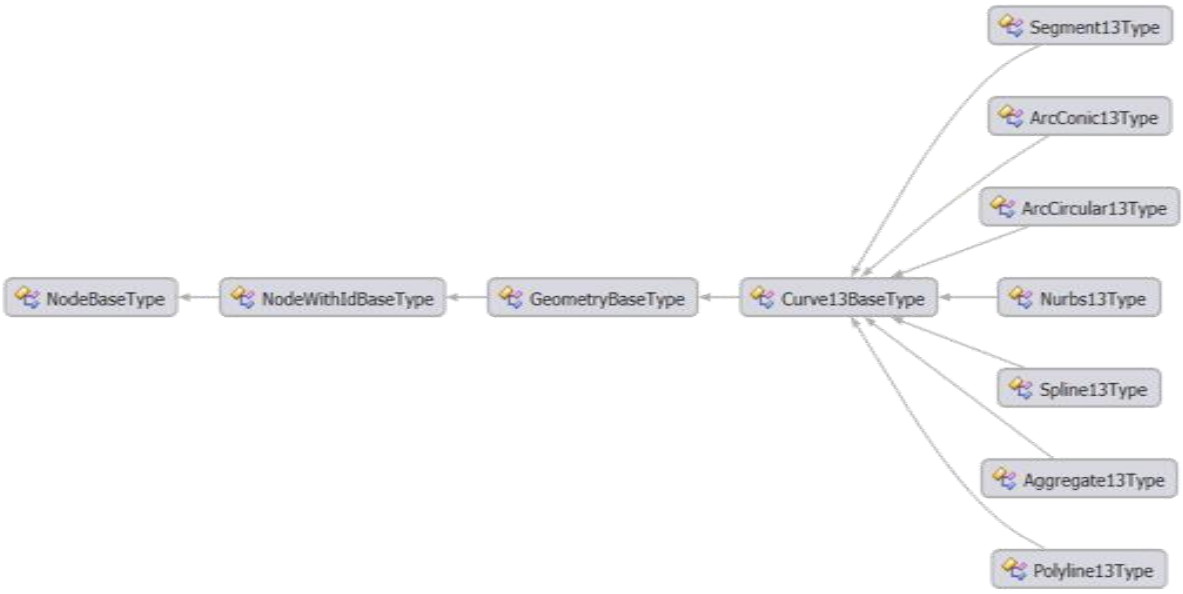


Figure 21 – 3D Curve Types

All 3D geometric curves are represented using a geometric core and a “wrapper" that includes an id for referencing.

The domain of every 3D curve is given in the domain attribute of the core, which is required.

The turned attribute of 3D curves is optional. If it does not appear, its default value is false.

The 3D curves are normally used as underlying geometry for topological edges.

All 3D curves contain the following fields:

Field Name	Data Type	Description
{Curve}/{Curve}Core	{Curve}CoreType	The curve core.
{Curve}/{Curve}Core/@domain	ParameterRangeType	The domain of the parameter space of the curve.
{Curve}/Transform	ElementReferenceType	The identifier of a three dimensional transformation matrix.

7.2.3.1 Segment

Segment13 describes a linear segment in 3D space as shown in Figure 22.

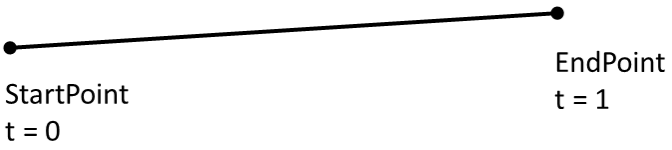


Figure 22 – 3D Segment

A linear segment is defined and positioned in 3D space with start and end points.

Function:

$$Segment13(t)\colon R_1 \rightarrow R_3$$

$$Segment13(t) = StartPoint + t(EndPoint - StartPoint), \qquad t \in [0,1]$$

Fields:

Field Name	Data Type	Description
Segment13Core/StartPoint	PointSimpleType	The beginning point of the linear segment.
Segment13Core/EndPoint	PointSimpleType	The end point of the linear segment.

Example:

```
<Segment13 id="350">
   <Segment13Core domain="0 1">
      <StartPoint>10.1 -2431.7 13.7</StartPoint>
      <EndPoint>34.1 4542.4 34.4</EndPoint>
   </Segment13Core>
 </Segment13>
```

7.2.3.2 Polyline

Polyline13 describes a polyline in 3D space as shown in Figure 23.

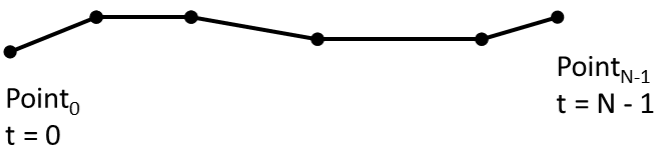


Figure 23 – 3D Polyline

A polyline is defined as a series of connected linear segments.

Function:

$$Polyline13(t)\colon R_1 \rightarrow R_3$$

$$Polyline13(t) = Point_i + (t - i)(Point_{i+1} - Point_i)$$

$$t \in [i, i+1], \qquad i \in [0, N-2], \qquad N - number\ of\ points$$

Fields:

Field Name	Data Type	Description
Polyline13Core/Points or Polyline13Core/PointsBinary	ArrayPointType or ArrayBinaryType	The array of 3D polyline points.

Example:

```
<Polyline13 id="32">
  <Polyline13Core domain="0 4">
    <Points N="5">
      1.2 11.5 10.4
      4.1 13.5 14.4
      7.3 16.2 17.1
      8.9 23.5 65.3
      9.2 29.8 84.4
    </Points>
  </Polyline13Core>
</Polyline13>
```

7.2.3.3 Circular Arc Curve

ArcCircular13 describes a circular arc in 3D space.

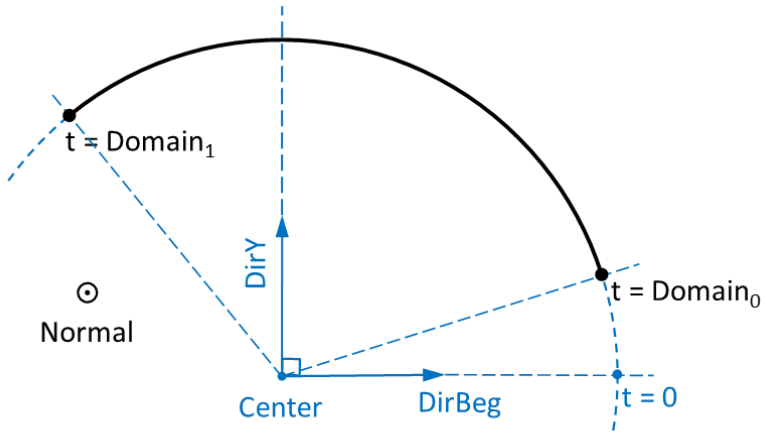


Figure 24 – 3D Circular Arc

A circular arc is defined as a portion of a circle as shown in Figure 24.

Function:

$$ArcCircular13(t)\colon R_1 \rightarrow R_3$$

$$ArcCircular13(t) = Center + Radius(\cos(t)\, DirBeg + \sin(t)\, DirY)$$

$$DirY = Normal \times DirBeg$$

where t is in radians, $t \in (-\infty, +\infty)$

Fields:

Field Name	Data Type	Description
ArcCircular13Core/Radius	xs:double	The arc radius.
ArcCircular13Core/Center	PointSimpleType	The center of arc.
ArcCircular13Core/DirBeg	UnitVectorSimpleType	The unit vector representing the beginning of the circular arc. The DirBeg must be perpendicular to the Normal.
ArcCircular13Core/Normal	UnitVectorSimpleType	The normal of the plane wherein the circular arc is defined.

Example:

```
<ArcCircular13 id="10">
  <ArcCircular13Core domain="0 3.1415926">
    <Radius>10.2</Radius>
    <Center>1.4 6.7 9.0</Center>
    <DirBeg>1.0 0.0 0.0</DirBeg>
    <Normal>0.0 0.0 1.0</Normal>
  </ArcCircular13Core>
</ArcCircular13>
```


7.2.3.4 Conic Arc Curve

ArcConic13 describes a conic arc in 3D space as shown in Figure 25 through Figure 27.

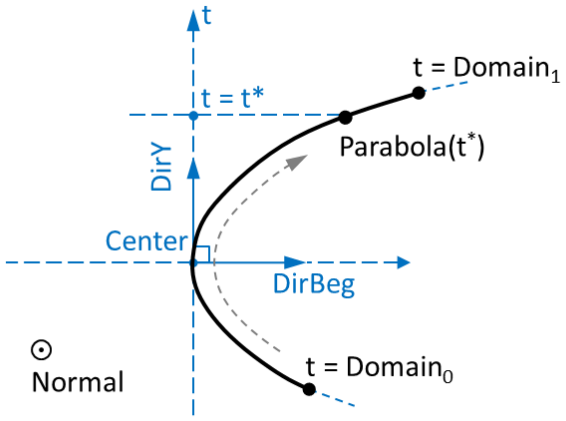


Figure 25 – 3D Conic Arc (form = PARABOLA)

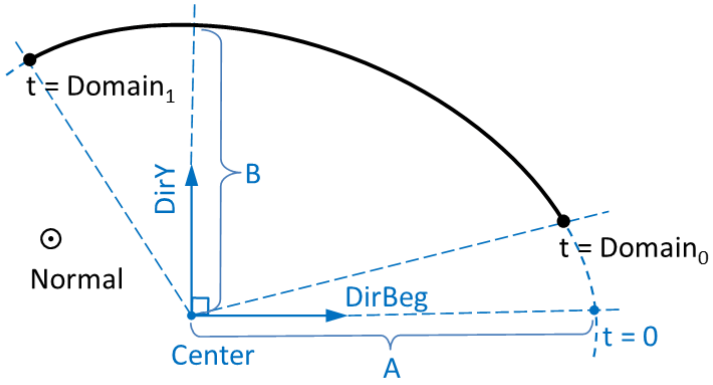


Figure 26 – 3D Conic Arc (form = ELLIPSE)

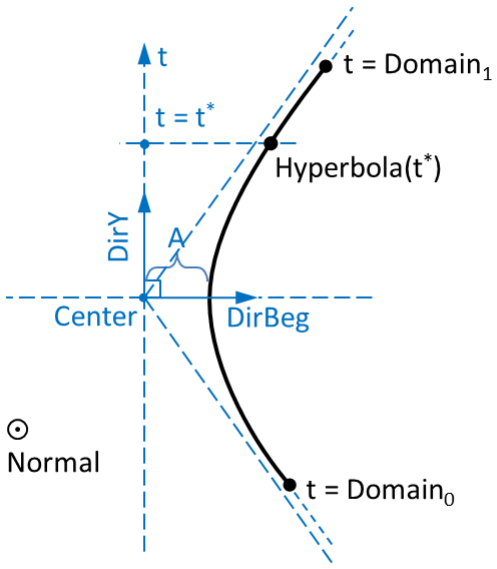


Figure 27 – 3D Conic Arc (form = HYPERBOLA)

A conic arc is defined as a portion of a conic curve. The conic curve can have one the following forms: an ellipse, a parabola, or a hyperbola.

Function:

$$ArcConic13(t)\colon R_1 \rightarrow R_3$$

$$ArcConic13(t) = Center + x(t)DirBeg + y(t)DirY$$

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{cases} \begin{bmatrix} At \\ Bt^2 \end{bmatrix}, & form = PARABOLA \\ \begin{bmatrix} A\cos(t) \\ B\sin(t) \end{bmatrix}, & form = ELLIPSE \\ \begin{bmatrix} A\sqrt{1 + t^2/B^2} \\ t \end{bmatrix}, & form = HYPERBOLA \end{cases}$$

$$DirY = Normal \times DirBeg$$

$$t \in (-\infty, +\infty)$$

Fields:

Field Name	Data Type	Description
ArcConic13Core/@form	ArcConicFormEnumType	The form of the conic arc.
ArcConic13Core/A	xs:double	Ellipse: the major radius of the arc. Parabola: the coefficient “A”. Hyperbola: the semi-axis (the distance from the center to the extreme point).

ArcConic13Core/B	xs:double	Ellipse: the minor radius of the arc. Parabola: the coefficient “B”. Hyperbola: the implicit parameter.
ArcConic13Core/Center	PointSimpleType	The center point.
ArcConic13Core/DirBeg	UnitVectorSimpleType	The unit vector representing the beginning of the conic arc. The DirBeg must be perpendicular to the Normal.
ArcConic13Core/Normal	UnitVectorSimpleType	The normal of the plane wherein the conic arc is defined.

Example:

```
<ArcConic13 id="123">
  <ArcConic13Core form="PARABOLA" domain="-0.2 3.5">
    <A>4.5</A>
    <B>3.1</B>
    <Center>13.3 -43.2 34.1</Center>
    <DirBeg>1.0 0.0 0.0</DirBeg>
    <Normal>0.0 1.0 0.0</Normal>
  </ArcConic13Core>
</ArcConic13>

<ArcConic13 id="123">
  <ArcConic13Core form="ELLIPSE" domain="10.3 18.9">
    <A>14.2</A>
    <B>33.1</B>
    <Center>63.7 -34.0 45.9</Center>
    <DirBeg>1.0 0.0 0.0</DirBeg>
    <Normal>0.0 1.0 0.0</Normal>
  </ArcConic13Core>
</ArcConic13>

<ArcConic13 id="123">
  <ArcConic13Core form="HYPERBOLA" domain="1.2 7.0">
    <A>14.5</A>
    <B>30.1</B>
    <Center>0.3 3.2 5.8</Center>
    <DirBeg>1.0 0.0 0.0</DirBeg>
    <Normal>0.0 1.0 0.0</Normal>
  </ArcConic13Core>
</ArcConic13>
```

7.2.3.5 Spline Curve

Spline13 describes a spline curve in 3D space as shown in Figure 28.

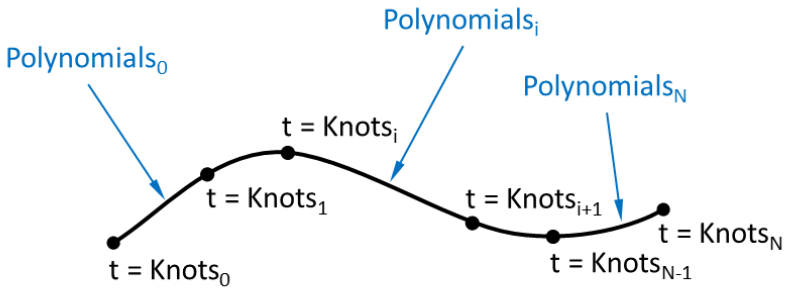


Figure 28 – 3D Spline Curve

A spline curve is defined as a sequence of parametric polynomial segments.

Function:

$$Spline13(t)\colon R_1 \to R_3$$

$$Spline13(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{Degree_p} (C_i^p)_x (t_p)^i \\ \sum_{i=0}^{Degree_p} (C_i^p)_y (t_p)^i \\ \sum_{i=0}^{Degree_p} (C_i^p)_z (t_p)^i \end{bmatrix}$$

$$N - number\ of\ polynomials$$

$$N = number\ of\ knots - 1$$

$$Coefficients = \{C^0, \dots, C^{N-1}\}$$

$$p - index\ of\ polynomials$$

$$C^p = \{c_0 \dots c_{Degree_p}\}, \quad Degree_p = Orders_p - 1$$

$$t_p - parameter\ of\ polynomials\ p$$

$$t_p \in \begin{cases} [Knots_p, Knots_{p+1}], & normalized = false \\ [0,1], & normalized = true \end{cases}$$

$$t_p = \begin{cases} t - Knots_p, & normalized = false \\ (t - Knots_p)/(Knots_{p+1} - Knots_p), & normalized = true \end{cases}$$

$$t \in [Knots_p, Knots_{p+1}], \qquad p \in [0, N-1]$$

Fields:

Field Name	Data Type	Description
Spline13Core/@normalized	xs:boolean	This flag shows if the curve is normalized.
Spline13Core/Knots	ArrayDoubleType	The spline breakpoints.
Spline13Core/Orders	ArrayNaturalType	The orders of the polynomial segments. The order is 'the degree of the polynomial' + 1. The size of this array is 'the number of spline breakpoints' - 1.
Spline13Core/Coefficients	ArrayPointType	The coefficients of the polynomial segments are triplets. For each segment the number of coefficients equal to the order on this segment. The total size of this array is the sum of all orders.

Example:

```
<Spline13 id="302">
  <Spline13Core domain="0 2">
    <Knots N="3">
      0 1 2
    </Knots>
    <Orders N="2">
      4 4
    </Orders>
    <Coefficients N="8">
      87.5 237.5 0
      14.0625 -7.8125 0
      0 0 0
      -1.5625 1.5625 0
      100 231.25 0
      9.375 -3.125 0
      -4.6875 4.6875 0
      1.5625 -1.5625 0
    </Coefficients>
  </Spline13Core>
</Spline13>
```

7.2.3.6 NURBS Curve

Nurbs13 describes a NURBS curve in 3D space as shown in Figure 29.

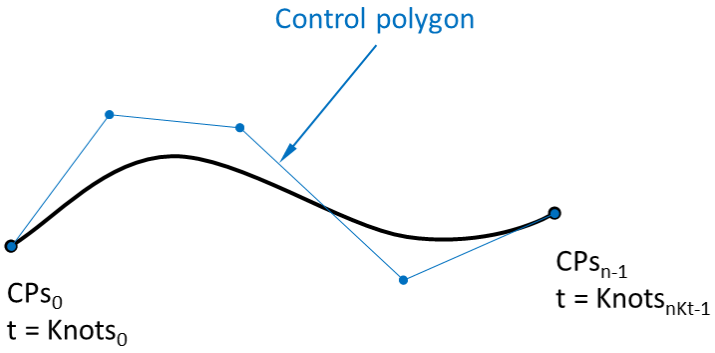


Figure 29 – 3D NURBS Curve

A NURBS curve is a freeform curve built on the B-spline basis functions and defined by its order (= degree + 1), a knot vector (an increasing sequence of real numbers which divides the parametric space in the intervals called knot spans), and an array of control points with an optional set of associated weights (positive real numbers). If the weights are not defined or equal, the curve is a polynomial one (otherwise rational).

Function:

$$Nurbs13(t)\colon R_1 \to R_3$$

$$Nurbs13(t) = \frac{\sum_{i=0}^{n-1} N_{i,Degree}\,(t) Weights_i CPs_i}{\sum_{i=0}^{n-1} N_{i,Degree}\,(t) Weights_i}$$

$$Degree = Order - 1$$

$$N_{i,j}(t) - the\ bspline\ basis\ functions$$

$$N_{i,0}(t) = \begin{cases} 1, & t \in [Knots_i, Knots_{i+1}) \\ 0, & otherwise \end{cases}$$

$$N_{i,p}(t) = \frac{t - Knots_i}{Knots_{i+p} - Knots_i} N_{i,p-1}(t) + \frac{Knots_{i+p+1} - t}{Knots_{i+p+1} - Knots_{i+1}} N_{i+1,p-1}(t)$$

$$t \in [Knots_0, Knots_{nKt-1}], \qquad nKt - number\ of\ knots$$

$$nKt = n + Order, \qquad n - number\ of\ control\ points$$

Fields:

Field Name	Data Type	Description
Nurbs13Core/Order	NaturalType	The order (= degree + 1).
Nurbs13Core/Knots	ArrayDoubleType	The knot vector is an increasing sequence of real numbers. The size of the knot vector is “number of control points” + “order”.
Nurbs13Core/CPs or Nurbs13Core/CPsBinary	ArrayPointType or ArrayBinaryType	The array of 3D control points.
Nurbs13Core/Weights	ArrayDoubleType	The array of weights associated with control points (positive real numbers).

Example:

```
<Nurbs13 id="607">
  <Nurbs13Core domain="0 1">
    <Order>3</Order>
    <Knots N="10">
       0 0 0 0.33333333 0.33333333 0.66666667 0.66666667 1 1 1
    </Knots>
    <CPs N="7">
      -386.61 -291.50 -44.00
      -386.61 -291.50 -54.39
      -377.61 -291.50 -49.19
      -368.61 -291.50 -44.00
      -377.61 -291.50 -38.80
      -386.61 -291.50 -33.60
      -386.61 -291.50 -44.00
    </CPs>
    <Weights N="7">
       1 0.5 1 0.5 1 0.5 1
    </Weights>
  </Nurbs13Core>
</Nurbs13>
```

7.2.3.7 Aggregate Curve

Aggregate13 describes an aggregate curve in 3D space as shown in Figure 30.

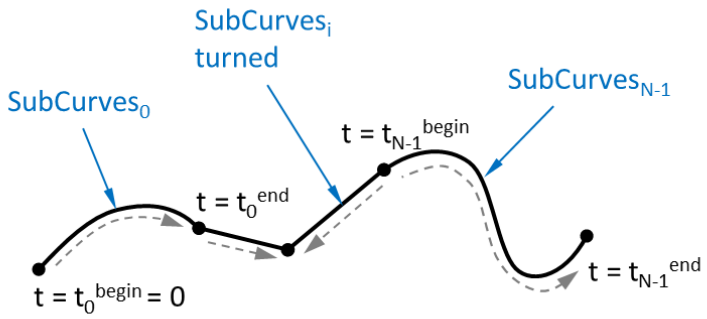


Figure 30 – 3D Aggregate Curve

An aggregate curve is a continuous curve which is defined as a sequence of oriented parametric sub-curves.

Function:

$$Aggregate13(t)\colon R_1 \to R_3$$

$$Aggregate13(t) = SubCurves_i(t_i)$$

$$t \in \left[t_i^{begin}, t_i^{end}\right], \qquad i \in [0, N-1], \qquad N - number\ of\ subcurves$$

$$domain_i^j := domain_j\ of\ SubCurves_i, \qquad j \in \{0,1\}$$

$$t_i^{begin} = \sum_{j<i}\left(domain_j^1 - domain_j^0\right)$$

$$t_i^{end} = t_i^{begin} + (domain_i^1 - domain_i^0)$$

$$t_i = \begin{cases} domain_i^0 + \left(t - t_i^{begin}\right), & SubCurves_i\ is\ not\ turned \\ domain_i^1 - \left(t - t_i^{begin}\right), & SubCurves_i\ is\ turned \end{cases}$$

Fields:

Field Name	Data Type	Description
Aggregate13Core/SubCurves	ArraySubCurve13Type	The array of oriented sub-curves.
Aggregate13Core/SubCurves/SubCurve/@turned	xs:boolean	The orientation of sub-curve.

Example:


```
<Aggregate13 id="419">
  <Aggregate13Core domain="0 4">
    <SubCurves N="4">
      <SubCurve turned="1">
        <Segment13Core domain="0 1">
          <StartPoint>-367.36 -214.75 -36.50</StartPoint>
          <EndPoint>-367.36 -207.50 -36.50</EndPoint>
        </Segment13Core>
      </SubCurve>
      <SubCurve>
        <Nurbs13Core domain="0 1">
          <Order>3</Order>
          <Knots N="6">
            0 0 0 1 1 1
          </Knots>
          <CPs N="3">
            -367.36 -214.75 -36.50
            -366.61 -214.75 -36.50
            -366.61 -214.75 -37.25
          </CPs>
          <Weights N="3">
            1 0.70710678 1
          </Weights>
        </Nurbs13Core>
      </SubCurve>
      <SubCurve>
        <Segment13Core domain="0 1">
          <StartPoint>-366.61 -214.75 -37.25</StartPoint>
          <EndPoint>-366.61 -207.50 -37.25</EndPoint>
        </Segment13Core>
      </SubCurve>
      <SubCurve>
        <Nurbs13Core domain="0 1">
          <Order>4</Order>
          <Knots N="10">
            0 0 0 0 0.91406368 0.91406368 1 1 1 1
          </Knots>
          <CPs N="6">
            -366.61 -207.50 -37.25
            -366.61 -207.50 -36.87
            -366.88 -207.50 -36.56
            -367.29 -207.50 -36.50
            -367.33 -207.50 -36.50
            -367.36 -207.50 -36.50
          </CPs>
        </Nurbs13Core>
      </SubCurve>
    </SubCurves>
  </Aggregate13Core>
</Aggregate13>
```

7.2.4 Parametric Surfaces

This chapter describes parametric surfaces, S(u, v):R2 → R3, where

- R2 is surface parameter space - the 2D real number space, where each point is defined as a pair (u, v)
- R3 is model space - the three-dimensional Euclidean space, where each point is defined as a coordinate triplet (x, y, z) specified in a right-handed Cartesian coordinate system

The correspondence between 2D parameter space and 3D Euclidean space is illustrated in Figure 31.

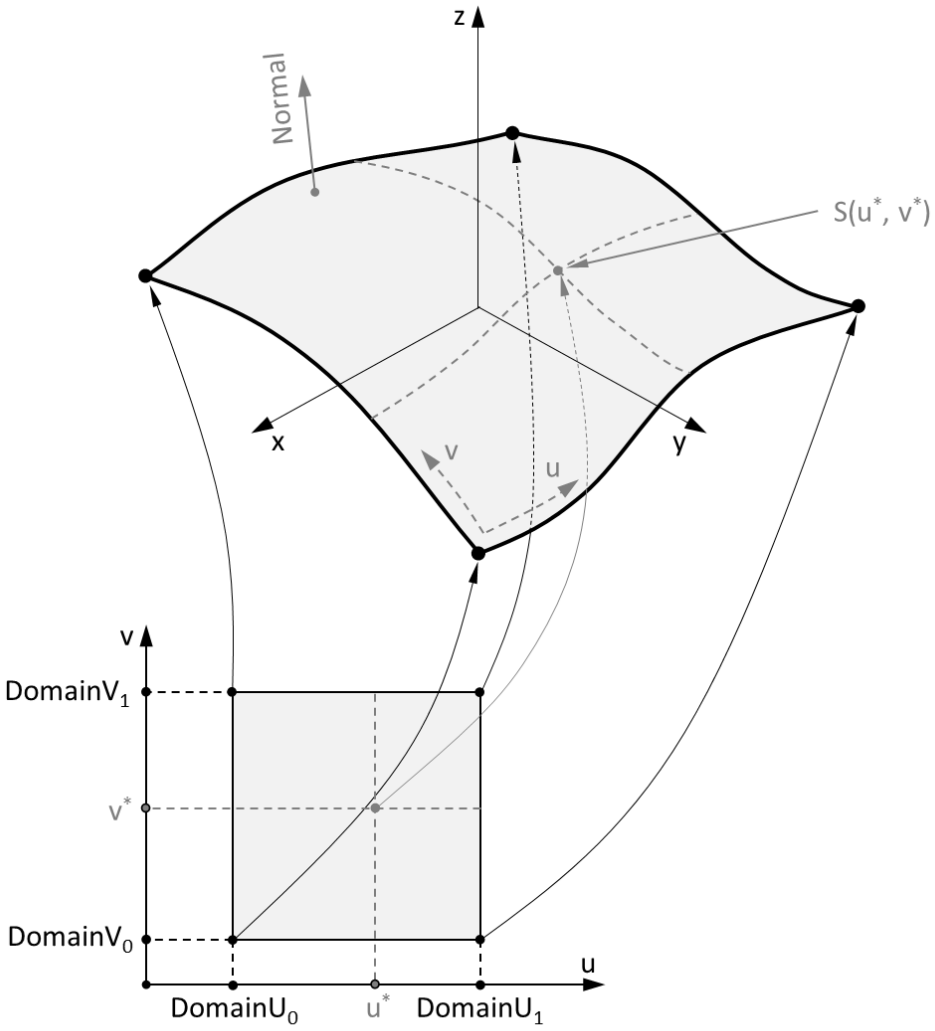


Figure 31 – Parametric Surface

All types of QIF surfaces are derived from SurfaceBaseType as shown in Figure 32.

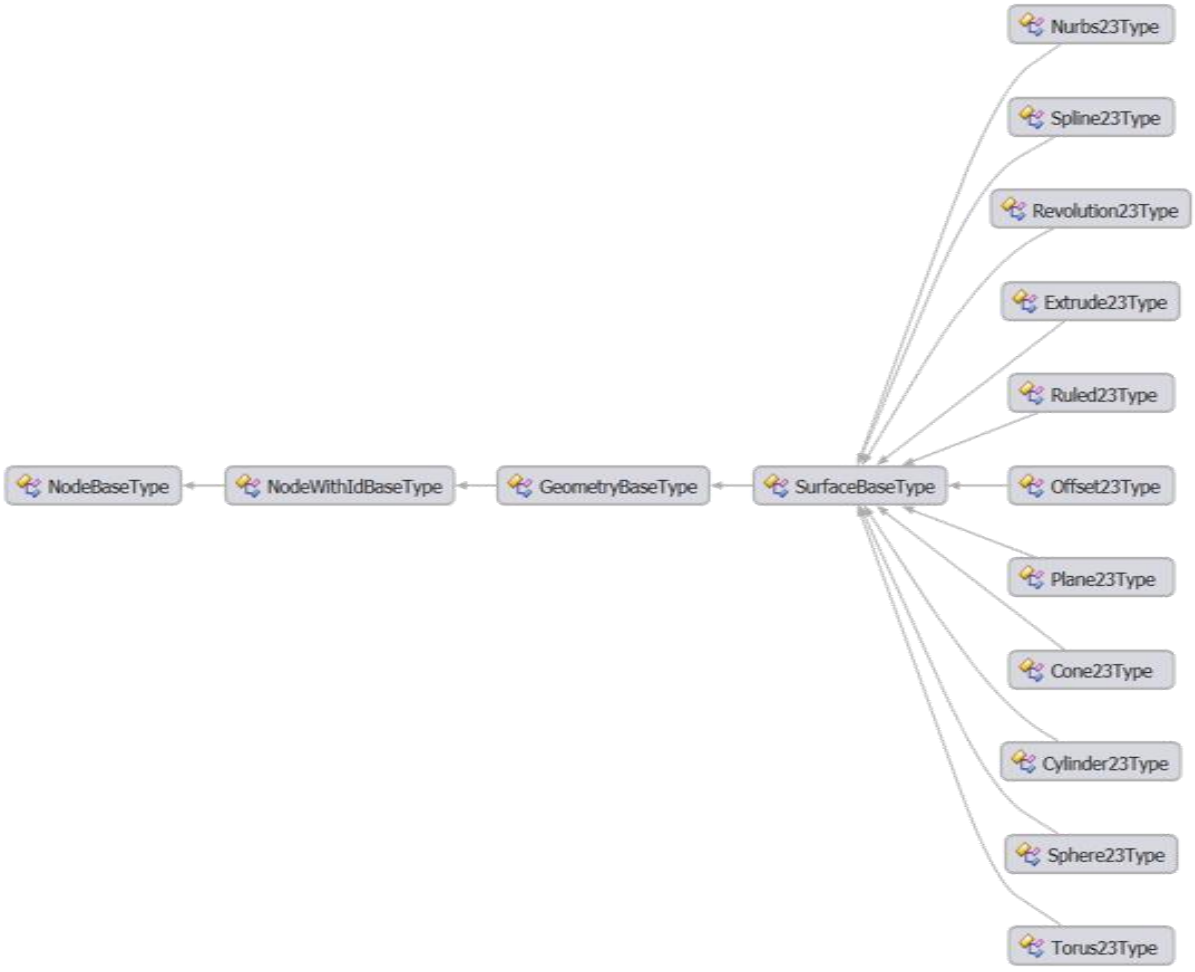


Figure 32 – Parametric Surface Types

The surfaces are normally used as underlying geometry for topological faces.

All parametric surfaces contain the following fields:

Field Name	Data Type	Description
{Curve}/{Curve}Core	{Curve}CoreType	The surface core.
{Curve}/{Curve}Core/@form	Attr23CoreEnumType	This field enumerates values that describe the surface form: 'FREEFORM' - a freeform surface (NURBS, spline etc.); 'CYLINDER' - a cylindrical surface; 'CONE' - a conical surface; 'TORUS' - a toroidal surface; 'SPHERE' - a spherical surface;

		'PLANE' - a plane surface;
{Curve}/Transform	ElementReferenceType	The identifier of a three dimensional transformation matrix.

About the scaling coefficient

The scaling coefficient scaleV normalizes the parameterization in the v-direction, so very large and very small cylinders, cones and tori have reasonable parameterization. Additionally the scaling coefficient preserves approximate angle correspondence in 2D and 3D space as shown in Figure 33.

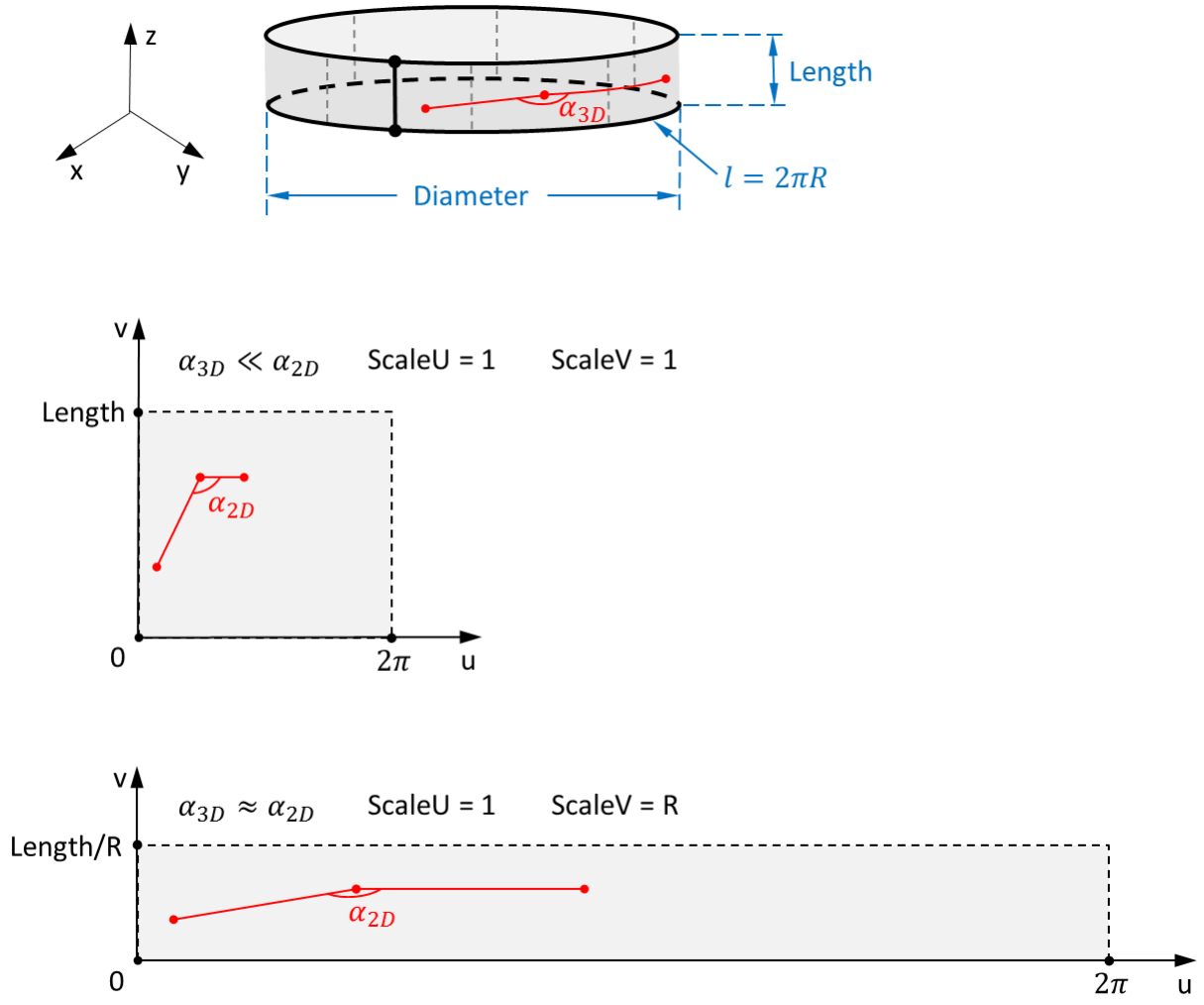


Figure 33 – Scaling coefficient

7.2.4.1 Plane Surface

Plane23 describes a plane parametric surface as shown in Figure 34.

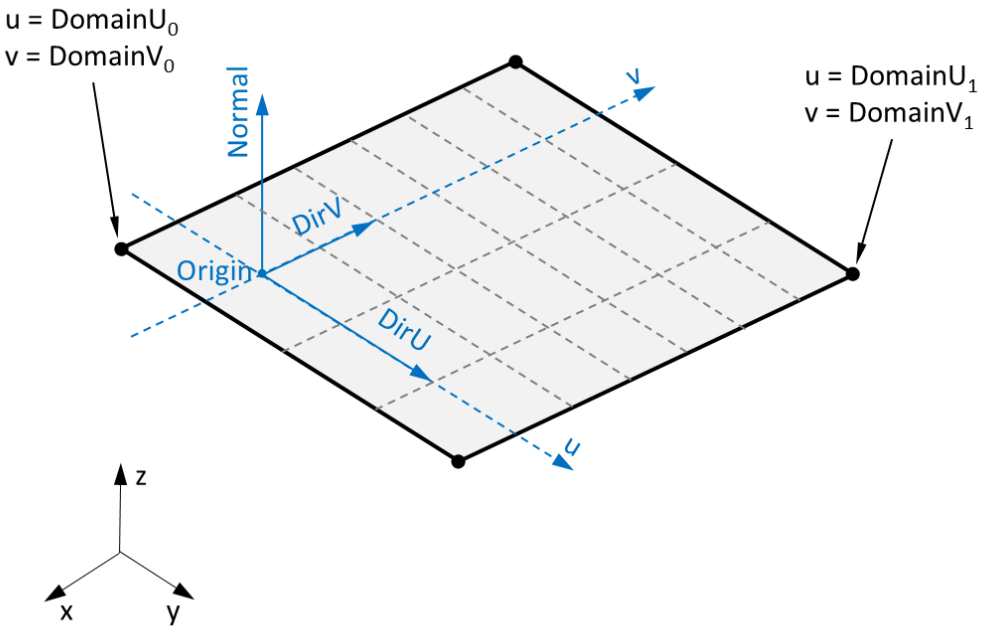


Figure 34 – Plane

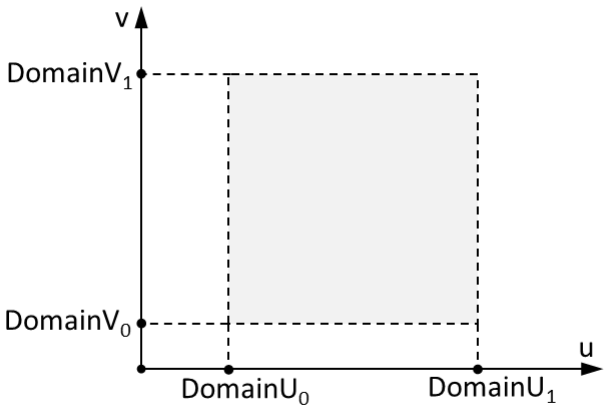


Figure 35 – Plane (Parameter Space)

Plane is an infinite flat surface. The parameterization of the plane is defined by two vectors: DirU and DirV. As seen in Figure 35, domain values (domainU and domainV) 'limit' the plane infinity.

Function:

$$Plane23(u,v)\colon R_2 \rightarrow R_3$$

$$Plane23(u,v) = Origin + uDirU + vDirV$$

$$u \in [domainU_0, domainU_1], \qquad v \in [domainV_0, domainV_1]$$

$$Normal = \|DirU \times DirV\|$$

Fields:

Field Name	Data Type	Description
Plane23Core/@domainU	ParameterRangeType	The parameter domain in the U-direction.
Plane23Core/@domainV	ParameterRangeType	The parameter domain in the V-direction.
Plane23Core/Origin	PointSimpleType	The origin point on the plane.
Plane23Core/DirU	VectorSimpleType	The direction and scaling of U-parameter lines. The DirU vector must not be parallel or anti-parallel to the DirV vector.
Plane23Core/DirV	VectorSimpleType	The direction and scaling of V-parameter lines.

Example:

```
<Plane23 id="233">
  <Plane23Core domainU="-1.0 10.0" domainV="3.0 4.0">
    <Origin>3.0 0.0 5.5</Origin>
    <DirU>2.0 0.0 0.0</DirU>
    <DirV>0.0 3.0 0.0</DirV>
  </Plane23Core>
</Plane23>
```


7.2.4.2 Cylinder Surface

Cylinder23 describes a cylinder parametric surface as shown in Figure 36, Figure 37, and Figure 38.

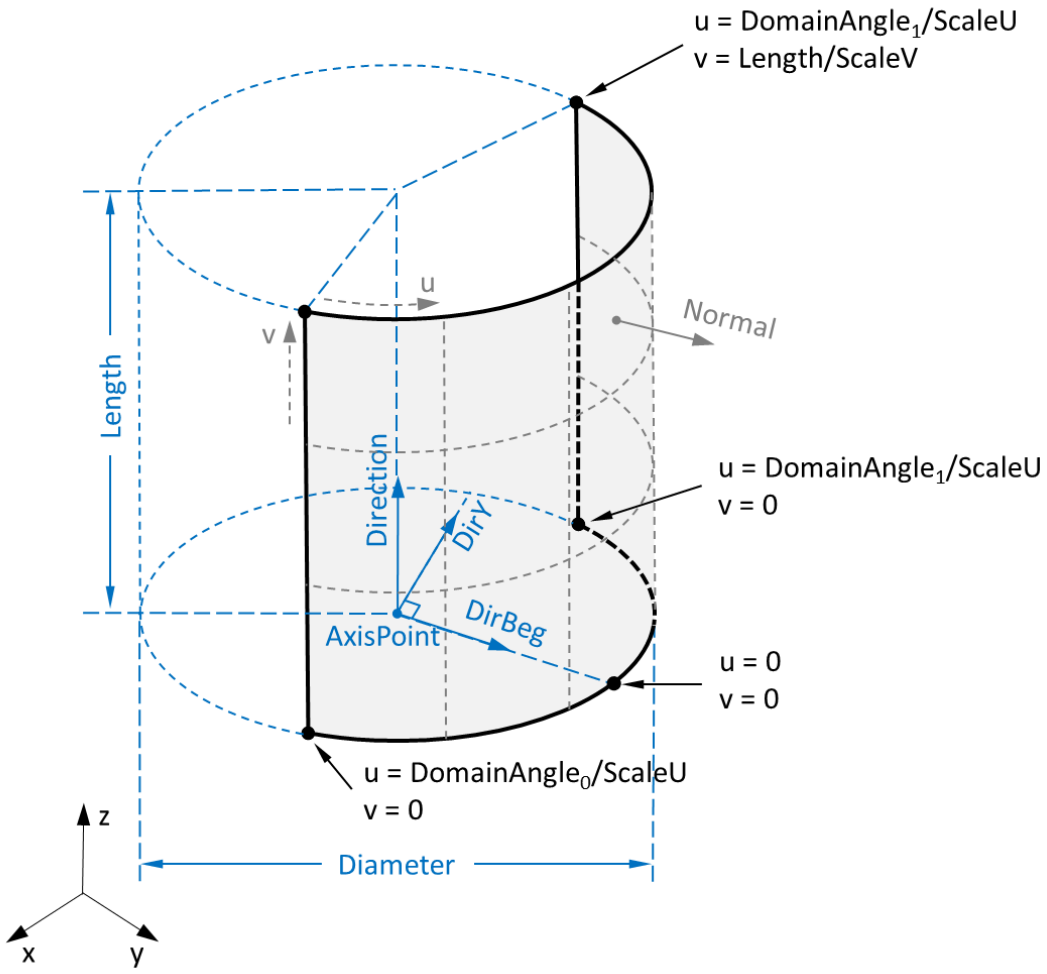


Figure 36 – Cylinder

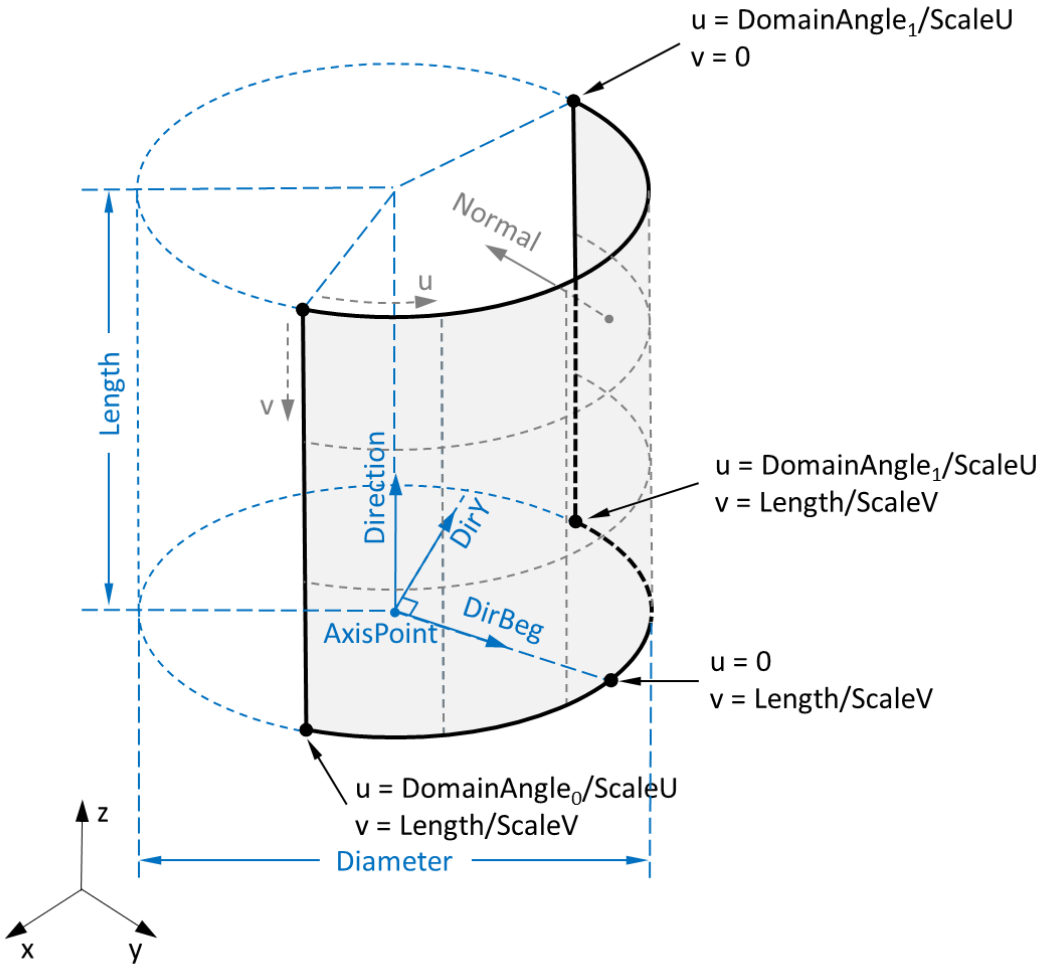


Figure 37 – Cylinder (turnedV = true)

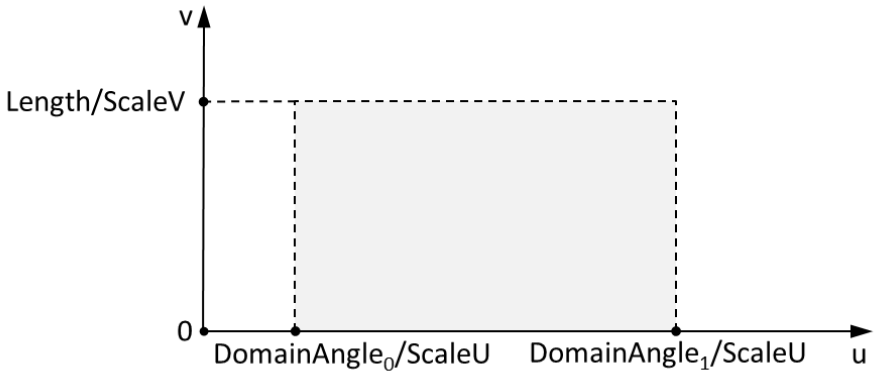


Figure 38 – Cylinder (Parametric Space)

The cylinder is a surface generated by rotating a 3D linear segment around a specified axis on a specified angle. The rotating segment must be parallel to the specified axis.

Function:

$$Cylinder23(u,v)\colon R_2 \to R_3$$

$$Cylinder23(u,v) = AxisPoint + R(\cos(u')DirBeg + \sin(u')DirY) + v'Direction$$

$$R = \frac{Diameter}{2}, \qquad DirY = Direction \times DirBeg$$

$$u' = u * scaleU$$

$$where\ u'\ is\ measured\ in\ radians$$

$$v' = \begin{cases} v * scaleV, & turnedV = false \\ Length - v * scaleV, & turnedV = true \end{cases}$$

$$u \in \left[\frac{domainAngle_0}{scaleU}, \frac{domainAngle_1}{scaleU}\right], \qquad v \in [0, Length/scaleV]$$

$$where\ DomainAngle\ in\ formula\ is\ measured\ in\ radians,$$
$$and\ DomainAngle\ in\ file\ is\ measured\ in\ the\ units\ specified\ in\ the\ element,$$
$$or\ in\ radians\ if\ no\ units\ are\ specified$$

Fields:

Field Name	Data Type	Description
Cylinder23Core/@turnedV	xs:boolean	This flag shows if the v direction of the cylinder must be inverted.
Cylinder23Core/@scaleU	DoublePositiveType	The scaling coefficient of the u direction of the parametric space.
Cylinder23Core/@scaleV	DoublePositiveType	The scaling coefficient of the v direction of the parametric space.
Cylinder23Core/Diameter	xs:double	The cylinder diameter.
Cylinder23Core/Length	xs:double	The height of the cylinder – a distance between the cylinder top and bottom.
Cylinder23Core/Axis/AxisPoint	PointType	The axis origin (the bottom center).
Cylinder23Core/Axis/Direction	UnitVectorType	The axis vector.
Cylinder23Core/Sweep/DirBeg	UnitVectorType	The start direction specifies the position of the cylinder seam (u=0). This vector must lie in a plane normal to the axis of the cone.
Cylinder23Core/Sweep/DomainAngle	AngleRangeType	The sweep angle from the start direction (the surface domain in the u direction).

Example:


```
<Cylinder23 id="132">
  <Cylinder23Core>
    <Diameter>4.4</Diameter>
    <Length>7.1</Length>
    <Axis>
      <AxisPoint>10.0 0.0 0.0</AxisPoint>
      <Direction>0.0 0.0 1.0</Direction>
    </Axis>
    <Sweep>
      <DirBeg>1.0 0.0 0.0</DirBeg>
      <DomainAngle>1.3 1.9</DomainAngle>
    </Sweep>
  </Cylinder23Core>
</Cylinder23>
```

7.2.4.3 Cone Surface

Cone23 describes a cone parametric surface as shown in Figure 39, Figure 40, and Figure 41.

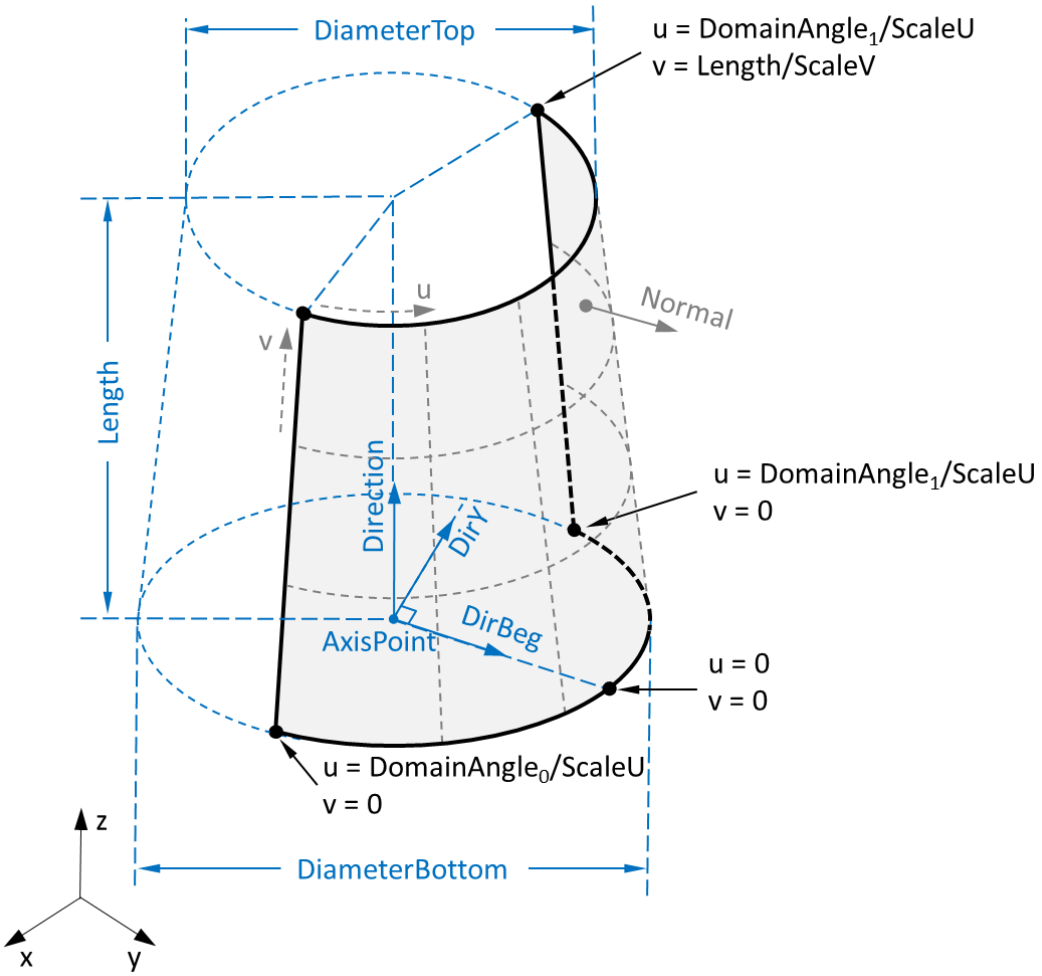


Figure 39 – Cone

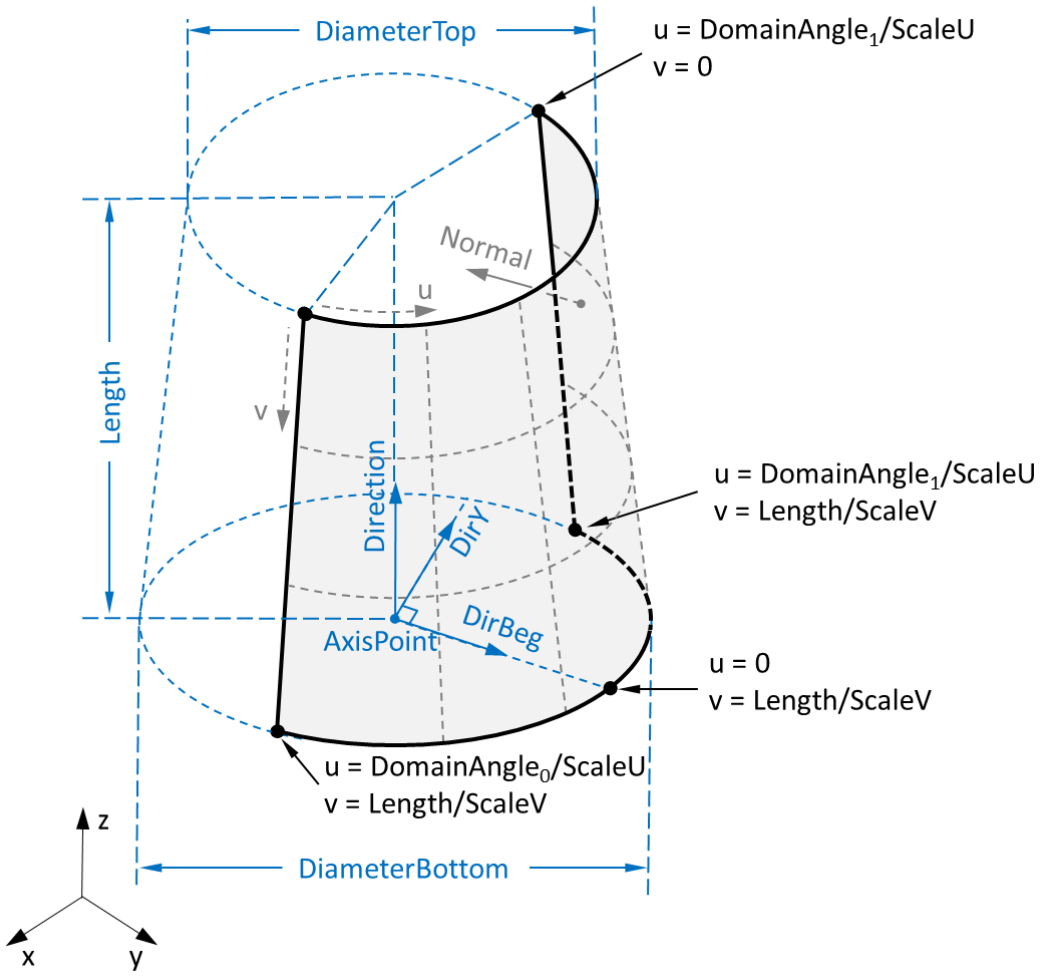


Figure 40 – Cone (turnedV = true)

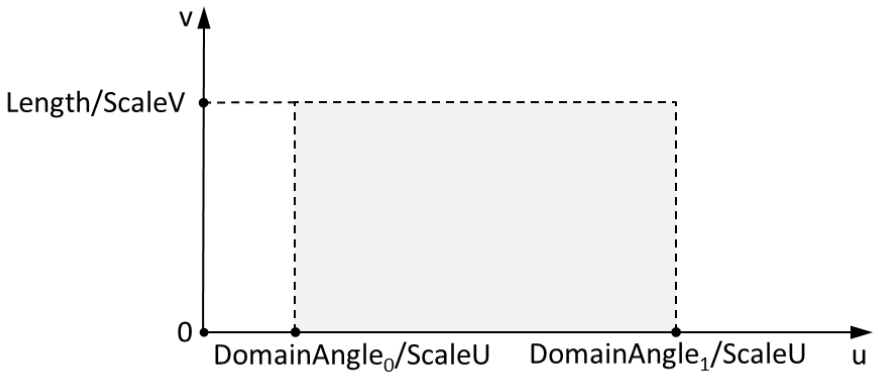


Figure 41 – Cone (Parametric Space)

The cone is a surface generated by rotating a 3D linear segment around a specified axis on a specified angle. The axis and the rotating segment must be coplanar.

Function:

$$Cone23(u,v)\colon\ R_2 \rightarrow R_3$$

$$Cone23(u,v) = AxisPoint + R(v')(\cos(u')DirBeg + \sin(u')DirY) + v' Direction$$

$$RadiusTop = \frac{DiameterTop}{2}, \qquad RadiusBottom = \frac{DiameterBottom}{2}$$

$$R(v) = RadiusBottom + v\frac{RadiusTop - RadiusBottom}{Length}$$

$$DirY = Direction \times DirBeg$$

$$u' = u * scaleU$$

$$where\ u'\ is\ measured\ in\ radians$$

$$v' = \begin{cases} v * scaleV, & turnedV = false \\ Length - v * scaleV, & turnedV = true \end{cases}$$

$$u \in [DomainAngle_0/scaleU, DomainAngle_1/scaleU], \qquad v \in [0, Length/scaleV]$$

$$where\ DomainAngle\ in\ formula\ is\ measured\ in\ radians,$$
$$and\ DomainAngle\ in\ file\ is\ measured\ in\ the\ units\ specified\ in\ the\ element,$$
$$or\ in\ radians\ if\ no\ units\ are\ specified$$

Fields:

Field Name	Data Type	Description
Cone23Core/@turnedV	xs:boolean	This flag shows if the v direction of the cone must be inverted.
Cone23Core/@scaleU	DoublePositiveType	The scaling coefficient of the u direction of the parametric space.
Cone23Core/@scaleV	DoublePositiveType	The scaling coefficient of the v direction of the parametric space.
Cone23Core/DiameterBottom	xs:double	The diameter at the bottom.
Cone23Core/DiameterTop	xs:double	The diameter at the top.
Cone23Core/Length	xs:double	The cone height – a distance between the cone top and bottom.
Cone23Core/Axis/AxisPoint	PointType	The axis origin (the bottom center).
Cone23Core/Axis/Direction	UnitVectorType	The axis vector.
Cone23Core/Sweep/DirBeg	UnitVectorType	The start direction specifies the position of the cone seam (u=0). This vector must lie in a plane normal to the axis of the cone.

Cone23Core/Sweep/DomainAngle	AngleRangeType	The sweep angle from the start direction (the surface domain in the u direction).

Example:

```
<Cone23 id="132">
  <Cone23Core>
    <DiameterBottom>4.4</DiameterBottom>
    <DiameterTop>2.1</DiameterTop>
    <Length>7.1</Length>
    <Axis>
      <AxisPoint>10.0 0.0 0.0</AxisPoint>
      <Direction>0.0 0.0 1.0</Direction>
    </Axis>
    <Sweep>
      <DirBeg>1.0 0.0 0.0</DirBeg>
      <DomainAngle>1.3 1.9</DomainAngle>
    </Sweep>
  </Cone23Core>
</Cone23>
```

7.2.4.4 Sphere Surface

Sphere23 describes a sphere parametric surface as shown in Figure 42, Figure 43, and Figure 44.

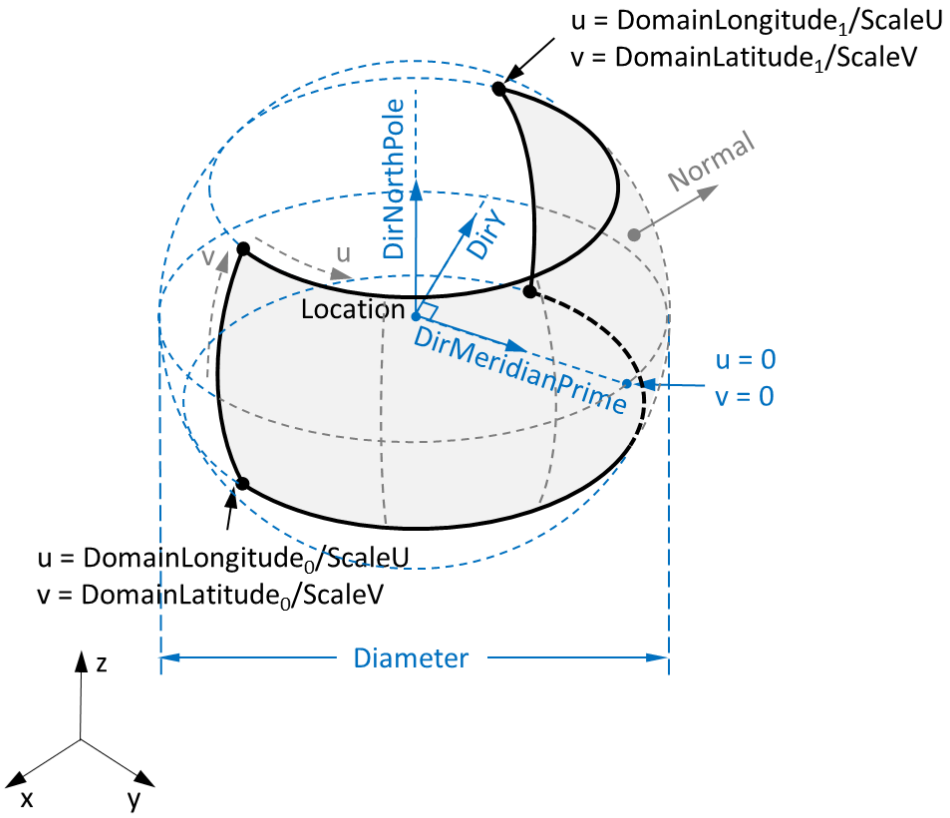


Figure 42 – Sphere

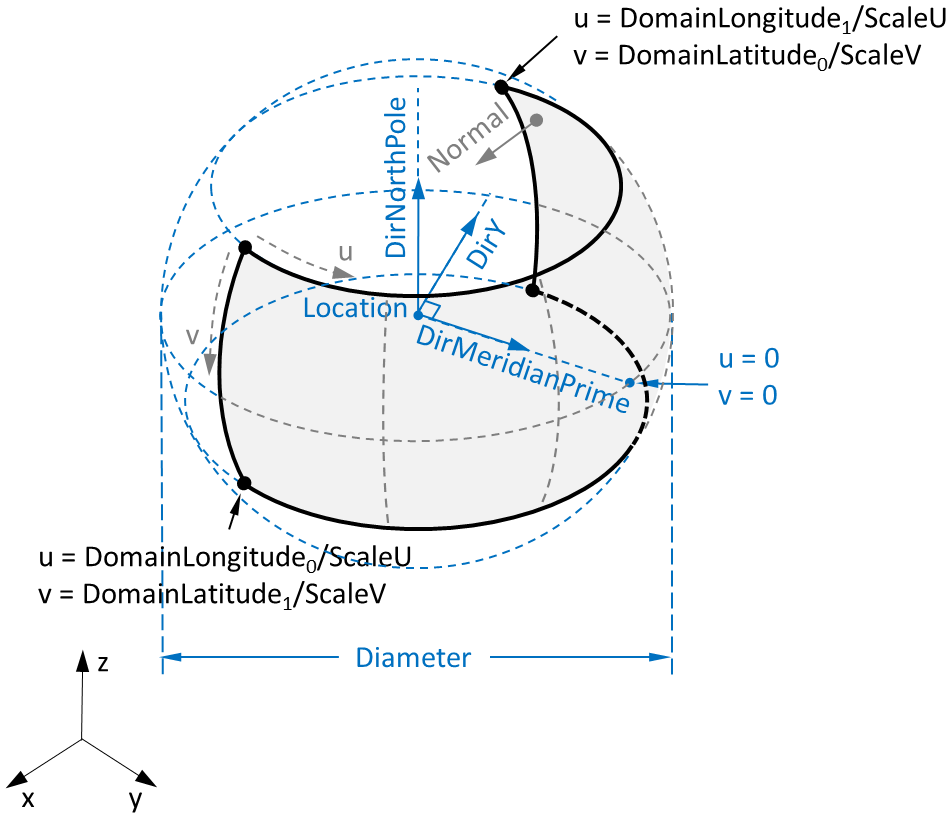


Figure 43 – Sphere (turnedV = true)

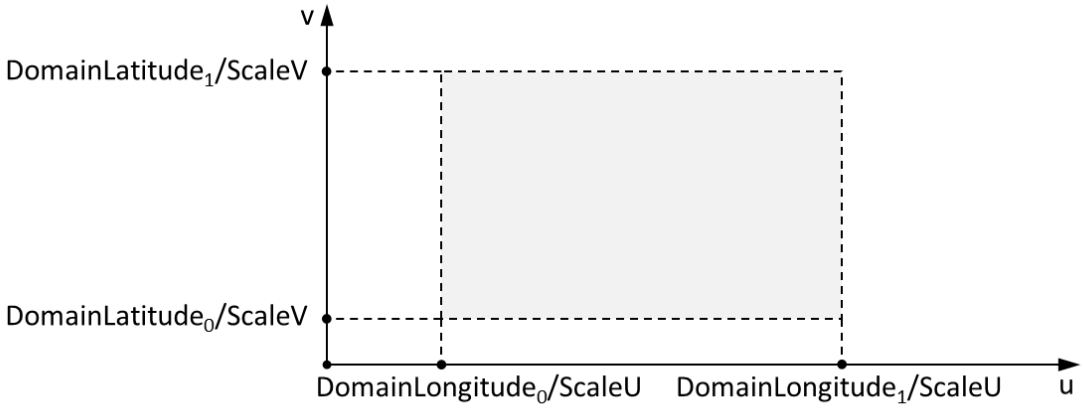


Figure 44 – Sphere (Parametric Space)

The sphere is a surface defined by its center point and radius.

Function:

$$Sphere23(u,v)\colon\ R_2 \rightarrow R_3$$

$$Sphere23(u,v) = Location + R\sin(v')\,DirNorthPole + R\cos(v')(\cos(u')DirMeridianPrime + \sin(u')DirY)$$

$$R = \frac{Diameter}{2}$$

$$DirY = DirNorthPole \times DirMeridianPrime$$

$$u' = u * scaleU$$

$$where\ u'\ is\ measured\ in\ radians$$

$$v' = \begin{cases} v * scaleV, & turnedV = false \\ -v * scaleV, & turnedV = true \end{cases}$$

$$where\ v'\ is\ measured\ in\ radians$$

$$u \in [DomainLongitude_0/scaleU, DomainLongitude_1/scaleU]$$

$$where\ DomainLongitude\ in\ formula\ is\ measured\ in\ radians,\ and\ DomainLongitude\ in\ file\ is\ measured\ in\ the\ units\ specified\ in\ the\ element,\ or\ in\ radians\ if\ no\ units\ are\ specified$$

$$v \in [DomainLatitude_0/scaleV, DomainLatitude_1/scaleV]$$

$$where\ DomainLatitude\ in\ formula\ is\ measured\ in\ radians,\ and\ DomainLatitude\ in\ file\ is\ measured\ in\ the\ units\ specified\ in\ the\ element,\ or\ in\ radians\ if\ no\ units\ are\ specified$$

Fields:

Field Name	Data Type	Description
Sphere23Core/@turnedV	xs:boolean	This flag shows if the v direction of the sphere must be inverted.
Sphere23Core/@scaleU	DoublePositiveType	The scaling coefficient of the u direction of the parametric space.
Sphere23Core/@scaleV	DoublePositiveType	The scaling coefficient of the v direction of the parametric space.
Sphere23Core/Diameter	xs:double	The sphere diameter.

Sphere23Core/Location	PointSimpleType	The sphere center.
Sphere23Core/LatitudeLongitudeSweep/DirMeridianPrime	UnitVectorType	The direction of the prime meridian vector. The longitude is 0 on the PrimeMeridianVector. This vector must be perpendicular to the north pole vector.
Sphere23Core/LatitudeLongitudeSweep/DomainLatitude	AngleRangeType	The latitude domain (the u direction). The latitude end angle must be greater than the latitude start angle.
Sphere23Core/LatitudeLongitudeSweep/DomainLongitude	AngleRangeType	The longitude domain (the u direction).
Sphere23Core/LatitudeLongitudeSweep/DirNorthPole	UnitVectorType	The direction of the north pole vector.

Example:

```
<Sphere23 id="132">
  <Sphere23Core>
    <Diameter>7.4</Diameter>
    <Location>2.0 3.0 2.3</Location>
    <LatitudeLongitudeSweep>
      <DirMeridianPrime>1.0 0.0 0.0</AxisPoint>
      <DomainLatitude>-0.1 0.4</DomainLatitude>
      <DomainLongitude>0.2 1.4</ DomainLongitude>
      <DirNorthPole>0.0 0.0 1.0</DirNorthPole>
    </LatitudeLongitudeSweep >
  </Sphere23Core>
</Sphere23>
```

7.2.4.5 Torus Surface

Torus23 describes a torus parametric surface as shown in Figure 45, Figure 46, and Figure 47.

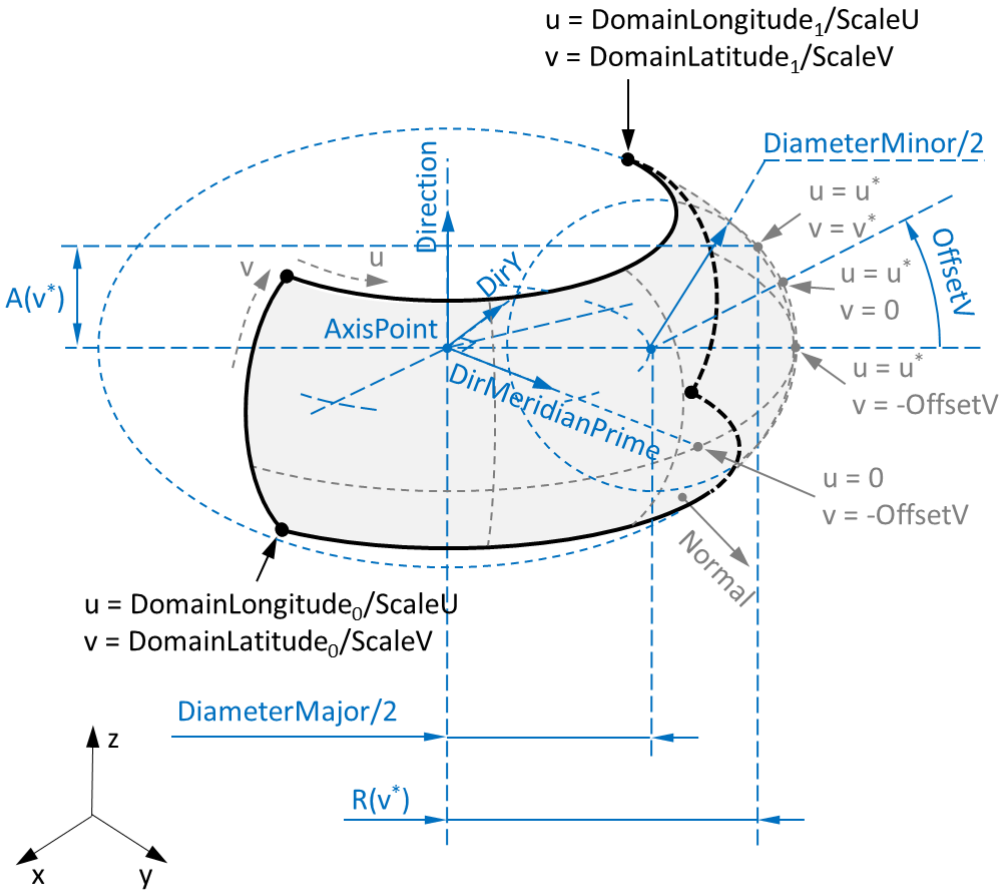


Figure 45 – Torus

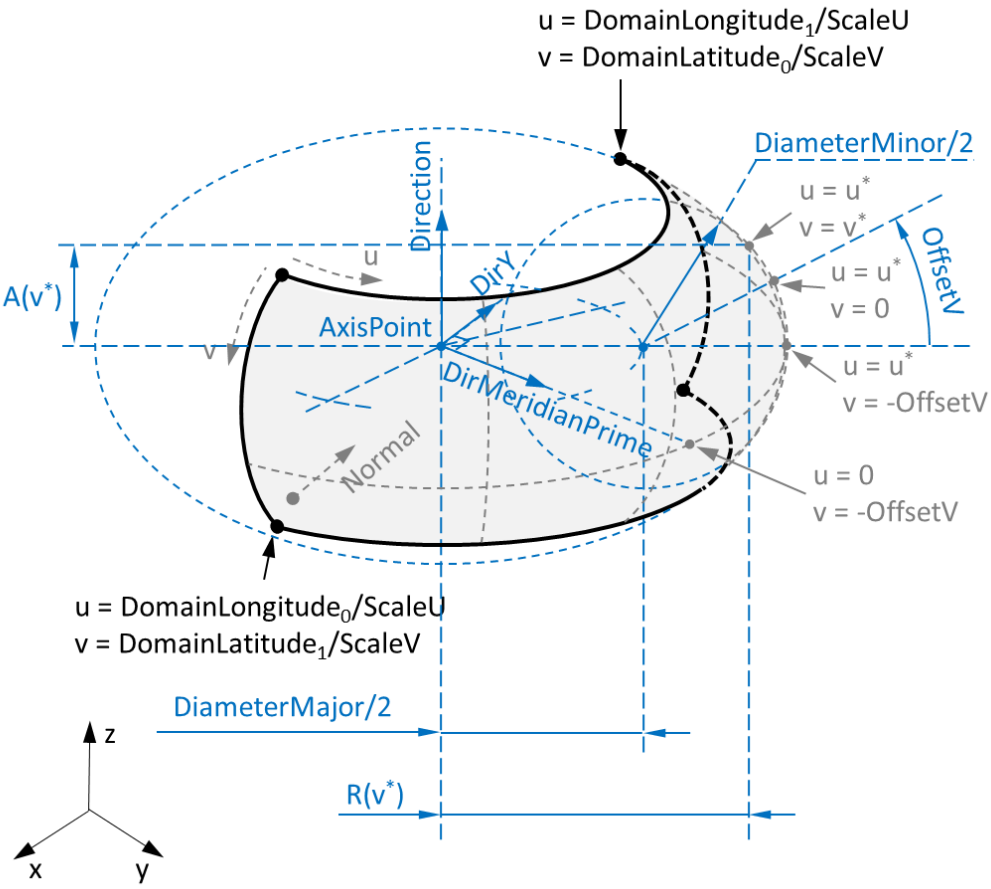


Figure 46 – Torus (turnedV = true)

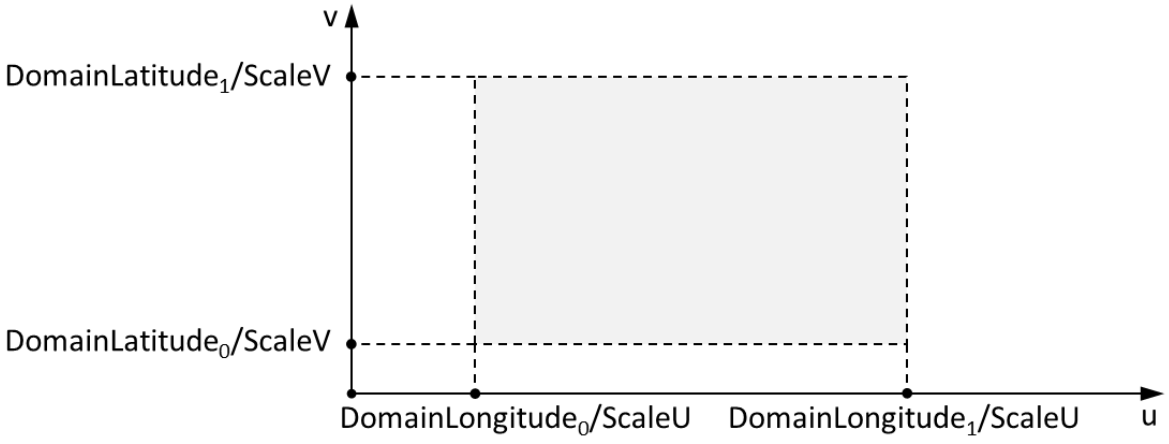


Figure 47 – Torus (Parametric Space)

The torus is a surface generated by rotating a 3D circular arc around an axis coplanar with the arc on a specified angle.

Function:

$$Torus23(u,v)\colon\ R_2 \rightarrow R_3$$

$$Torus23(u,v) = AxisPoint + A(v) + R(v)(\cos(u')DirMeridianPrime + \sin(u')\,DirY)$$

$$A(v) = RadiusMinor\,\sin(v')Direction$$

$$RadiusMinor = \frac{DiameterMinor}{2}, \qquad RadiusMajor = \frac{DiameterMajor}{2}$$

$$R(v) = RadiusMajor + RadiusMinor\,\cos(v')$$

$$DirY = Direction \times DirMeridianPrime$$

$$u' = u * scaleU$$

$$where\ u'\ is\ measured\ in\ radians$$

$$v' = \begin{cases} OffsetV + v * scaleV, & turnedV = false \\ OffsetV - v * scaleV, & turnedV = true \end{cases}$$

$$where\ v'\ is\ measured\ in\ radians$$

$$u \in [DomainLongitude_0/scaleU, DomainLongitude_1/scaleU]$$

$$where\ DomainLongitude\ in\ formula\ is\ measured\ in\ radians,$$
$$and\ DomainLongitude\ in\ file\ is\ measured\ in\ the\ units\ specified\ in\ the\ element,$$
$$or\ in\ radians\ if\ no\ units\ are\ specified$$

$$v \in [DomainLatitude_0/scaleV, DomainLatitude_1/scaleV]$$

$$where\ DomainLatitude\ in\ formula\ is\ measured\ in\ radians,$$
$$and\ DomainLatitude\ in\ file\ is\ measured\ in\ the\ units\ specified\ in\ the\ element,$$
$$or\ in\ radians\ if\ no\ units\ are\ specified$$

Fields:

Field Name	Data Type	Description
Torus23Core/@turnedV	xs:boolean	This flag shows if the v direction of the torus must be inverted.
Torus23Core/@offsetV	xs:double	The offsetting distance of the v direction of the parametric space.

Torus23Core/@scaleU	DoublePositiveType	The scaling coefficient of the u direction of the parametric space.
Torus23Core/@scaleV	DoublePositiveType	The scaling coefficient of the v direction of the parametric space.
Torus23Core/DiameterMinor	xs:double	The torus minor diameter.
Torus23Core/DiameterMajor	xs:double	The torus major diameter.
Torus23Core/Axis/AxisPoint	PointType	The axis origin (the bottom center).
Torus23Core/Axis/Direction	UnitVectorType	The axis vector.
Torus23Core/LatitudeLongitudeSweep/DirMeridianPrime	UnitVectorType	The direction of the prime meridian vector. The longitude is 0 on the PrimeMeridianVector. This vector must be perpendicular to the north pole vector.
Torus23Core/LatitudeLongitudeSweep/DomainLatitude	AngleRangeType	The latitude domain (the v direction).
Torus23Core/LatitudeLongitudeSweep/DomainLongitude	AngleRangeType	The longitude domain (the u direction). Regardless of the values of the longitude domain, the longitude sweep is in the positive direction.

Example:

```
<Torus23 id="344">
  <Torus23Core>
    <DiameterMinor>8.0</DiameterMinor>
    <DiameterMajor>24.0</DiameterMajor>
    <Axis>
      <AxisPoint>10.0 3.0 0.0</AxisPoint>
      <Direction>0.0 0.0 1.0</Direction>
    </Axis>
    <LatitudeLongitudeSweep>
      <DirMeridianPrime>1.0 0.0 0.0</DirMeridianPrime>
```

```
      <DomainLatitude>0.0 1.57</DomainLatitude>
      <DomainLongitude>0.0 3.14</DomainLongitude>
    </LatitudeLongitudeSweep>
  </Torus23Core>
</Torus23>
```


7.2.4.6 Extrude Surface

Extrude23 describes an extruded parametric surface as shown in Figure 48 and Figure 49.

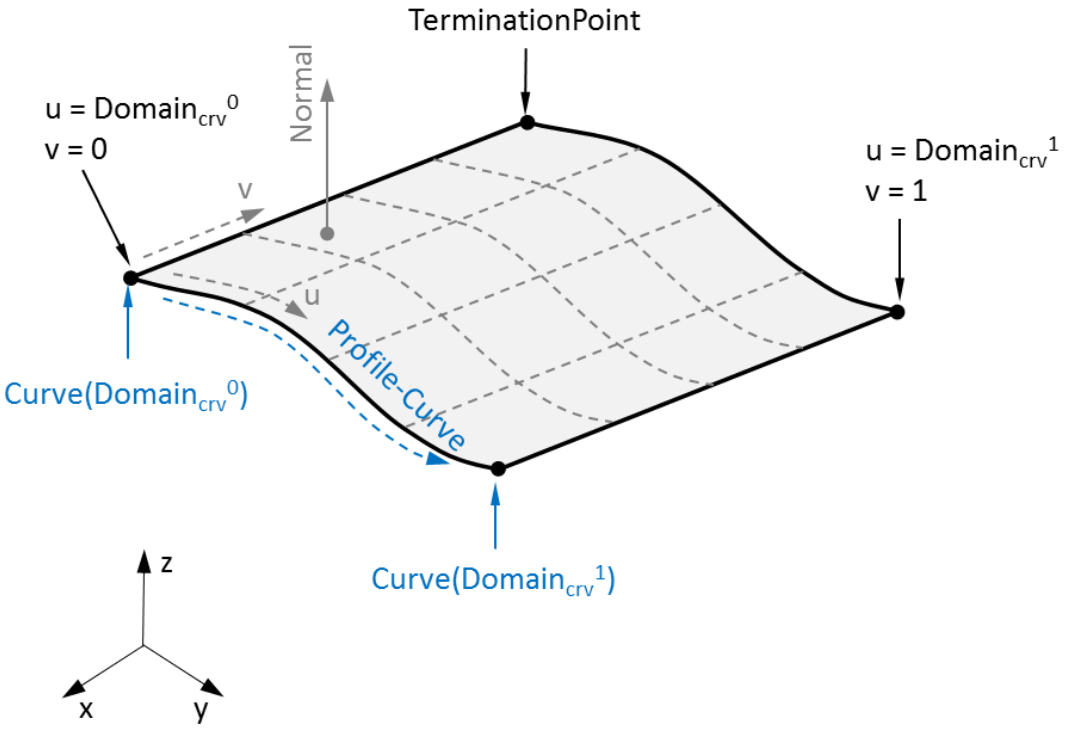


Figure 48 – Extrude Surface

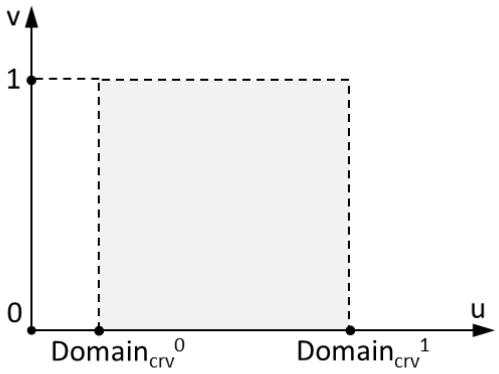


Figure 49 – Extrude Surface (Parametric Space)

The extrude surface is an extrusion of the profile-curve along the extrusion direction for a distance specified by the termination point.

Function:

$$Extrude23(u,v)\colon R_2 \rightarrow R_3$$

$$Extrude23(u,v) = Curve(u) + (TerminationPoint - Curve(domain^0_{crv}))v$$

$$u \in [domain^0_{crv}, domain^1_{crv}], \qquad v \in [0,1]$$

Fields:

Field Name	Data Type	Description
Extrude23Core/TerminationPoint	PointSimpleType	The termination point. Together with the curve start point it specifies the extrusion direction and distance.
Extrude23Core/Curve	Curve13CoreType	The curve to be used as the profile of extrusion.

Example:

```
<Extrude23 id="357">
  <Extrude23Core >
    <TerminationPoint>20.4 11.2 -0.9</TerminationPoint>
    <Curve>
      <ArcCircular13Core domain="0 3.14159265358979">
        <Radius>0.25</Radius>
        <Center>20.0 11.0 0.0</Center>
        <DirBeg>1.0 0.0 0.0</DirBeg>
        <Normal>0.0 0.0 1.0</Normal>
      </ArcCircular13Core>
    </Curve>
  </Extrude23Core>
</Extrude23>
```


7.2.4.7 Ruled Surface

Ruled23 describes a ruled parametric surface as shown in Figure 50, Figure 51, and Figure 52.

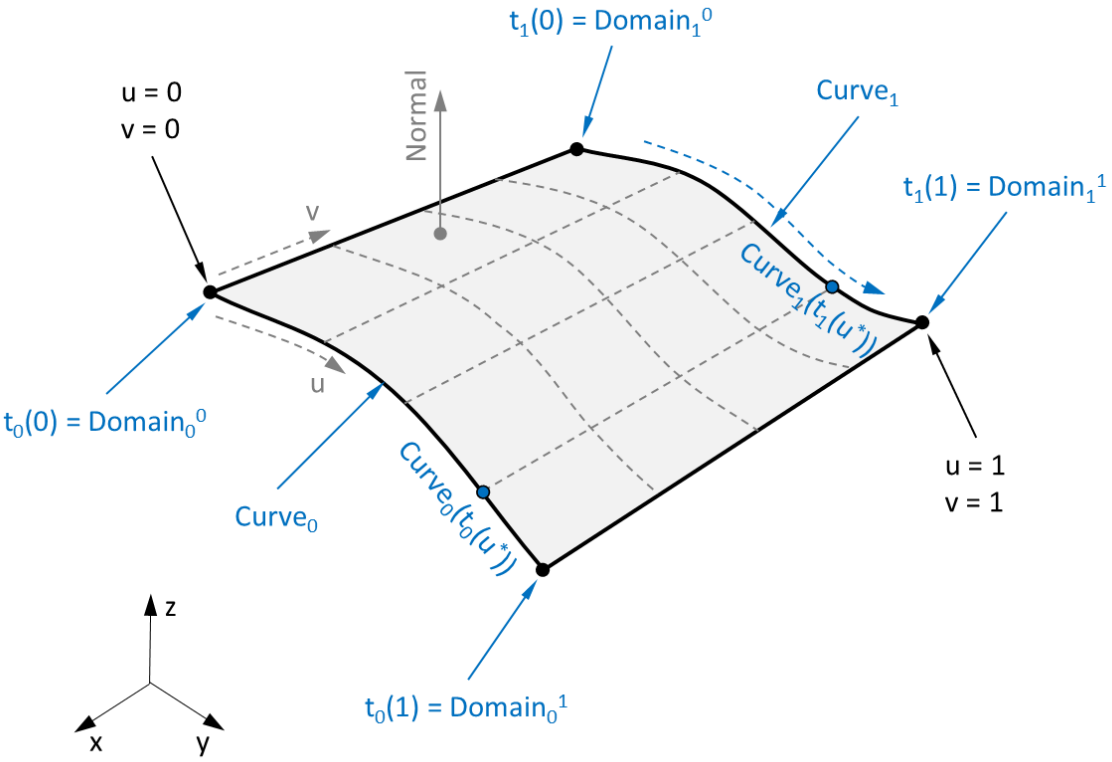


Figure 50 – Ruled Surface

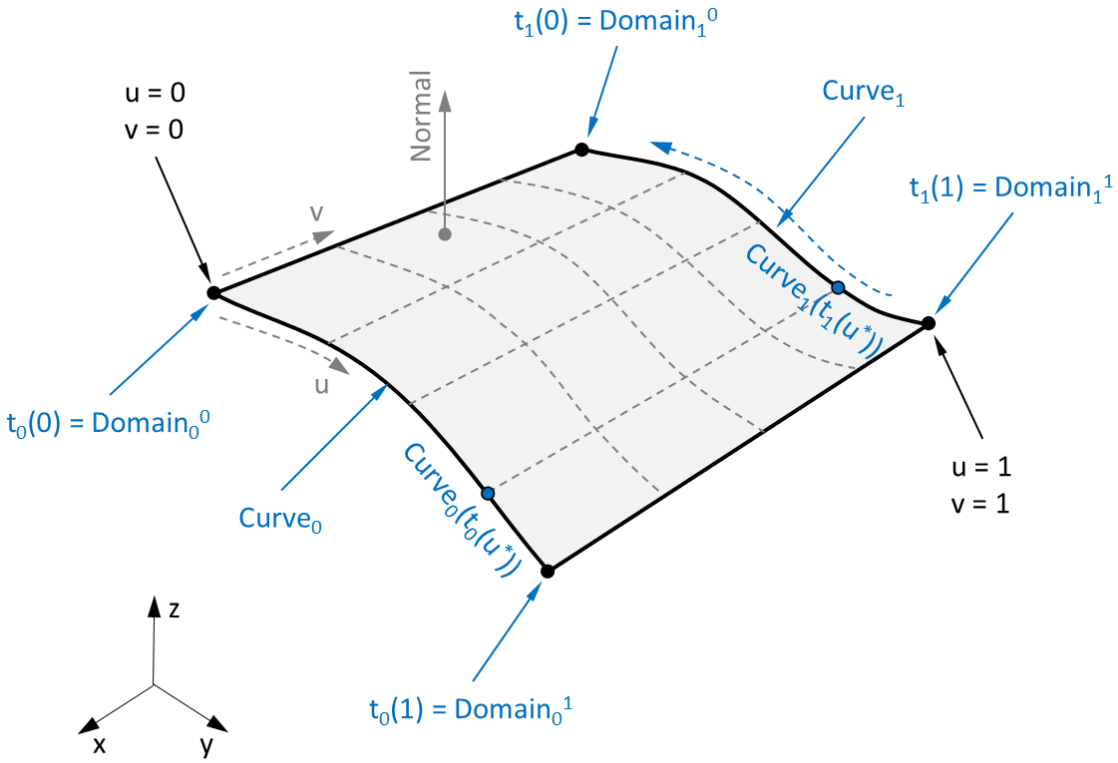


Figure 51 – Ruled Surface (turnedSecondCurve = true)

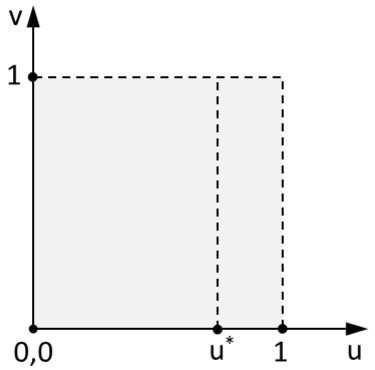


Figure 52 – Ruled Surface (Parametric Space)

The ruled surface is generated by connecting corresponding points on two 3D curves by a set of linear segments.

Function:

$Ruled23(u,v)\colon R_2 \to R_3$

$Ruled23(u,v) = Curve_0(t_0(u))(1-v) + Curve_1(t_1(u))v$

$domain_c - domain\ of\ curve\ c, \qquad c \in \{0,1\}$

$u \in [0,1], \qquad v \in [0,1]$

$t_0(u) = domain_0^0 + u(domain_0^1 - domain_0^0)$

$$t_1(u) = \begin{cases} domain_1^0 + u(domain_1^1 - domain_1^0), & turnedSecondCurve = false \\ domain_1^1 + u(domain_1^0 - domain_1^1), & turnedSecondCurve = true \end{cases}$$

Fields:

Field Name	Data Type	Description
Ruled23Core/@turnedSecondCurve	xs:boolean	This flag shows if the second curve is turned.
Ruled23Core/Curve[0]	Curve13CoreType	The first curve.
Ruled23Core/Curve[1]	Curve13CoreType	The second curve.

Example:

```
<Ruled23 id="361">
  <Ruled23Core>
    <Curve>
      <ArcCircular13Core domain="2.214 4.068">
        <Radius>0.625</Radius>
        <Center>14.8 11.0 0</Center>
        <DirBeg>1.0 0.0 0.0</DirBeg>
        <Normal>0.0 0.0 1.0</Normal>
      </ArcCircular13Core>
    </Curve>
    <Curve>
      <Segment13Core domain="0 1">
        <StartPoint>13.5 11.5 0.1</StartPoint>
        <EndPoint>13.5 10.5 0.3</EndPoint>
      </Segment13Core>
    </Curve>
  </Ruled23Core>
</Ruled23>
```

7.2.4.8 Surface of Revolution

Revolution23 describes a parametric surface of revolution as shown in Figure 53, Figure 54, and Figure 55.

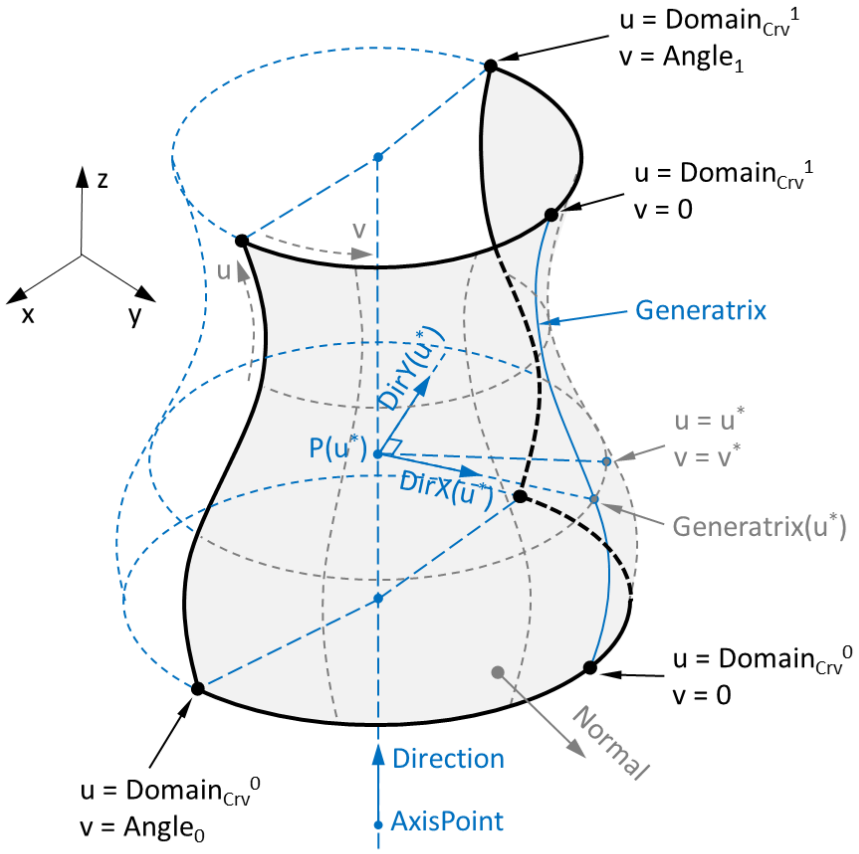


Figure 53 – Surface Of Revolution

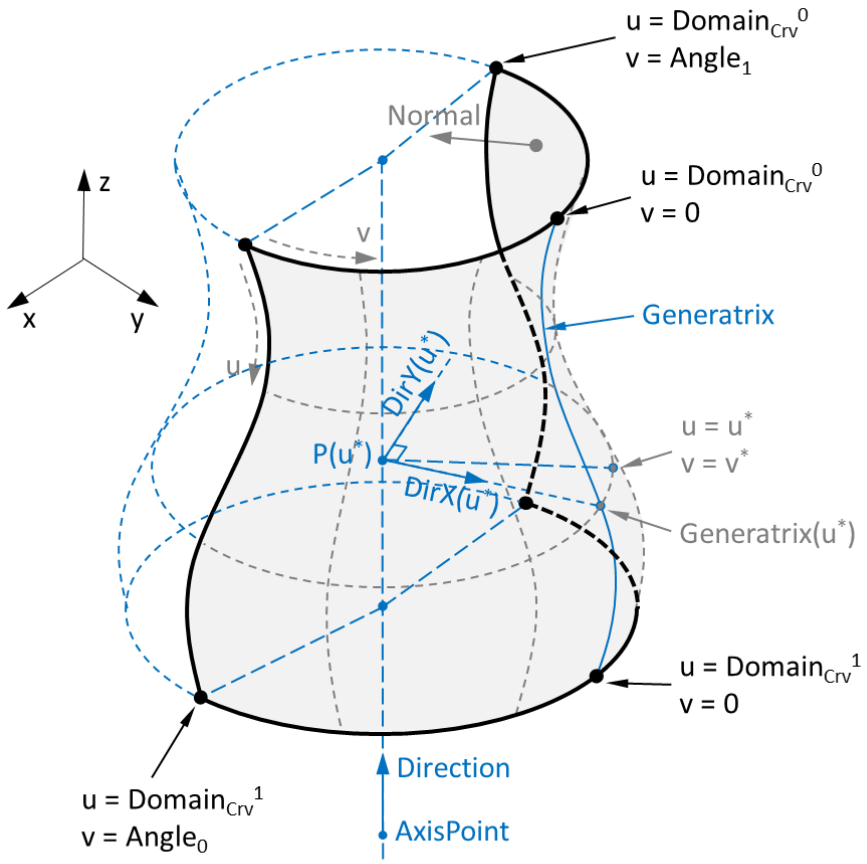


Figure 54 – Surface Of Revolution (turned Generatrix)

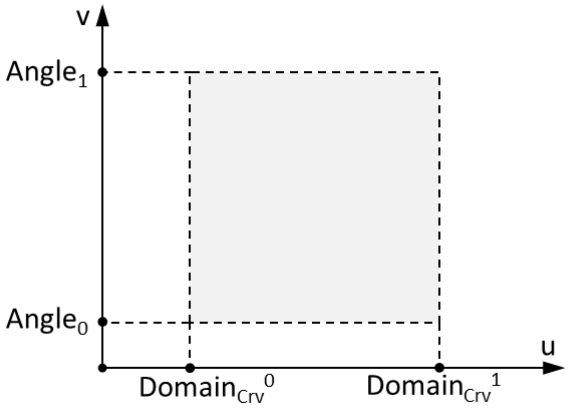


Figure 55 – Surface Of Revolution (Parametric Space)

The surface of revolution is generated by rotating a 3D generatrix curve around a specified axis on a specified angular range.

Function:

$$Revolution23(u,v)\colon\ R_2 \rightarrow R_3$$

$$Revolution23(u,v) = P(u) + R(u)\,(\cos(v)\,DirX(u) + \sin(v)\,DirY(u))$$

$$P(u) = AxisPoint + \big((Generatrix(u) - AxisPoint)\cdot Direction\big)Direction$$

$$R(u) = \|Generatrix(u) - P(u)\|$$

$$DirX(u) = \frac{Generatrix(u) - P(u)}{\|Generatrix(u) - P(u)\|}$$

$$DirY(u) = Direction \times DirX(u)$$

$$u \in [domain^0_{crv}, domain^1_{crv}], \qquad v \in [angle_0, angle_1]$$

Fields:

Field Name	Data Type	Description
Revolution23Core/@angle	ParameterRangeType	This field specifies start and end rotation angles.
Revolution23Core/Axis/AxisPoint	PointType	The axis origin.
Revolution23Core/Axis/Direction	UnitVectorType	The axis direction.
Revolution23Core/Generatrix	Curve13CoreType	The 3D curve to be rotated around the axis.

Example:

```
<Revolution23 id="354">
  <Revolution23Core angle="0 6.28318">
    <Axis>
      <AxisPoint>18.0 10.5 0</AxisPoint>
      <Direction>0.0 1.0 0.0</Direction>
    </Axis>
    <Generatrix>
      <ArcCircular13Core domain="1.965 4.317">
        <Radius>0.4</Radius>
        <Center>18.6 11.1 0.0</Center>
        <DirBeg>1.0 0.0 0.0</DirBeg>
        <Normal>0.0 0.0 1.0</Normal>
      </ArcCircular13Core>
    </Generatrix>
  </Revolution23Core>
</Revolution23>
```


7.2.4.9 Spline Surface

Spline23 describes a parametric spline surface as shown in Figure 56 and Figure 57.

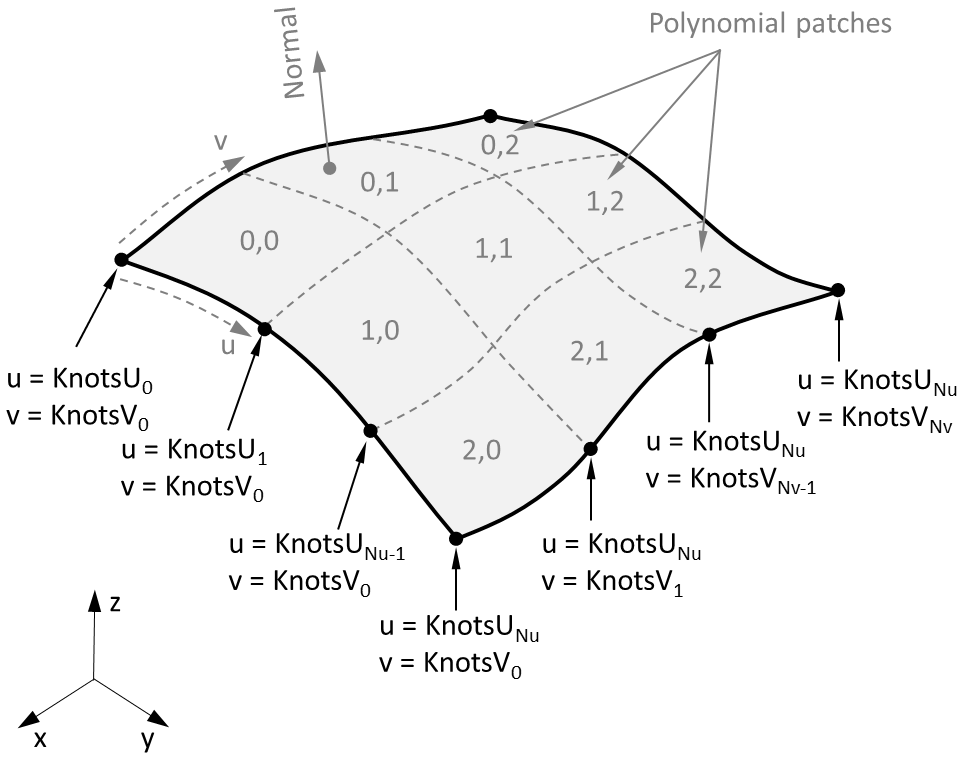


Figure 56 – Spline Surface

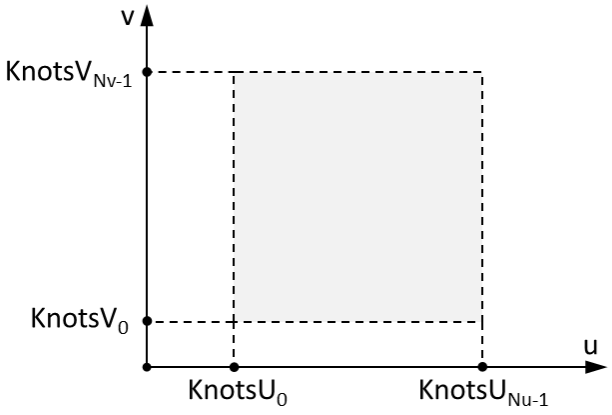


Figure 57 – Spline Surface (Parameter Space)

A spline surface is a grid of parametric polynomial patches.

Function:

$$Spline23(u,v)\colon R_2 \rightarrow R_3$$

$$Spline23(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{DegreeU_p} \sum_{j=0}^{DegreeV_q} (C_{i,j}^{p,q})_x (u_p)^i (v_q)^j \\ \sum_{i=0}^{DegreeU_p} \sum_{j=0}^{DegreeV_q} (C_{i,j}^{p,q})_y (u_p)^i (v_q)^j \\ \sum_{i=0}^{DegreeU_p} \sum_{j=0}^{DegreeV_q} (C_{i,j}^{p,q})_z (u_p)^i (v_q)^j \end{bmatrix}$$

$$Coefficients = \{\underbrace{C^{0,0} \dots C^{N_u-1,0}}_{N_u}, \dots, \underbrace{C^{0,N_v-1} \dots C^{N_u-1,N_v-1}}_{N_u}\}$$

$$DegreeU_p = OrdersU_p - 1$$

$$DegreeV_q = OrdersV_q - 1$$

$$NC - number\ of\ coefficients$$

$$NC^{p,q} - number\ of\ coefficients\ of\ polynomial\ patch\ (p,q)$$

$$NC^{p,q} = OrdersU_p OrdersV_q$$

$$NC = \sum_{p=0}^{N_u-1} \sum_{q=0}^{N_v-1} NC^{p,q}$$

$$C^{p,q} - coefficients\ of\ polynomial\ patch\ (p,q)$$

$$C^{p,q} = \{\underbrace{c_{0,0} \dots c_{DegreeU_p,0}}_{OrdersU_p}, \dots, \underbrace{c_{0,DegreeV_q} \dots c_{DegreeU_p,DegreeV_q}}_{OrdersU_p}\}$$

$$N_u, N_v - number\ of\ polynomials\ in\ the\ u\ and\ v\ directions$$

$$p \in [0, N_u], \qquad N_u = number\ of\ knots\ in\ the\ u\ direction - 1$$

$$q \in [0, N_v], \qquad N_v = number\ of\ knots\ in\ the\ v\ direction - 1$$

$$u_p - parameter\ of\ polynomials\ p\ in\ the\ u\ direction$$

$$u_p \in \begin{cases} [KnotsU_p, KnotsU_{p+1}], & normalized = false \\ [0,1], & normalized = true \end{cases}$$

$$u_p = \begin{cases} u - KnotsU_p, & normalized = false \\ (u - KnotsU_p)/(KnotsU_{p+1} - KnotsU_p), & normalized = true \end{cases}$$

$$v_q - parameter\ of\ polynomials\ p\ in\ the\ v\ direction$$

$$v_q \in \begin{cases} [KnotsV_q, KnotsV_{q+1}], & normalized = false \\ [0,1], & normalized = true \end{cases}$$

$$v_q = \begin{cases} v - KnotsV_q, & normalized = false \\ (v - KnotsV_q)/(KnotsV_{q+1} - KnotsV_q), & normalized = true \end{cases}$$

$$u \in [KnotsU_i, KnotsU_{i+1}], \qquad i \in [0, N_u - 1]$$

$$v \in [KnotsV_j, KnotsV_{j+1}], \qquad j \in [0, N_v - 1]$$

Fields:

Field Name	Data Type	Description
Spline13Core/@normalized	xs:boolean	This flag shows if the spline surface is normalized. A value of 1 (or true) means the surface is normalized. A value of 0 (or false) means the surface is not normalized.
Spline23Core/KnotsU	ArrayDoubleType	The knot vector in the u direction (the u spline breakpoints).
Spline23Core/KnotsV	ArrayDoubleType	The knot vector in the v direction (the v spline breakpoints).
Spline23Core/OrdersV	ArrayNaturalType	The orders of the polynomial patches in the u direction. The order is 'the degree of the polynomial' + 1. The size of this array is 'the number of the u spline breakpoints' - 1.
Spline23Core/OrdersV	ArrayNaturalType	The orders of the polynomial patches in the v direction. The order is 'the degree of the polynomial' + 1. The size of this array is 'the number of the v spline breakpoints' - 1.
Spline13Core/Coefficients	ArrayPointType	The coefficients of the polynomial patches. For each patch the number of coefficients equals 'the u polynomial order of the patch' * 'the v polynomial order of the patch'. The total size of this array is the sum of all patch coefficients.

Example:

```
<Spline23 id="101">
  <Spline23Core>
    <KnotsU N="3">
      0 1 2
```

```
    </KnotsU>
    <KnotsV N="2">
      0 1
    </KnotsV>
    <OrdersU N="2">4 4</OrdersU>
    <OrdersV N="2">4 4</OrdersV>
    <Coefficients N="32">
      -29.7 -20 13
      47.1 18 33
      -21.6 -18 -90
      4.45 6.75 49
      -15.9 60 -48
      2.7 -54 0
      -2.7 54 132.3
      7.65 -20.25 -83.7
      30.9 -60 12
      6.3 54 117
      -13.05 -54 -314.1
      0.6 20.25 147.9
      -20.3 34 -5
      -14.1 -60 -129
      19.35 60 241.8
      -6.2 -20.75 -100.45
      0.25 -13.25 5
      17.25 2.25 0
      -8.25 2.25 57
      25.75 -6.25 -49
      -8.25 39.75 0.6
      20.25 -6.75 13.5
      20.25 -6.75 -118.8
      -53.25 18.75 77.7
      24.75 -39.75 -37.2
      -18 6.75 -67.5
      -11.25 6.75 129.6
      45.3 -18.75 -54.9
      -21.25 13.25 7.35
      6 -2.25 53.25
      0.75 -2.25 -59.55
      -8.3 16.25 14.95
    </Coefficients>
  </Spline23Core>
</Spline23>
```


7.2.4.10 NURBS Surface

Nurbs23 describes a NURBS parametric surface as shown in Figure 58 and Figure 59.

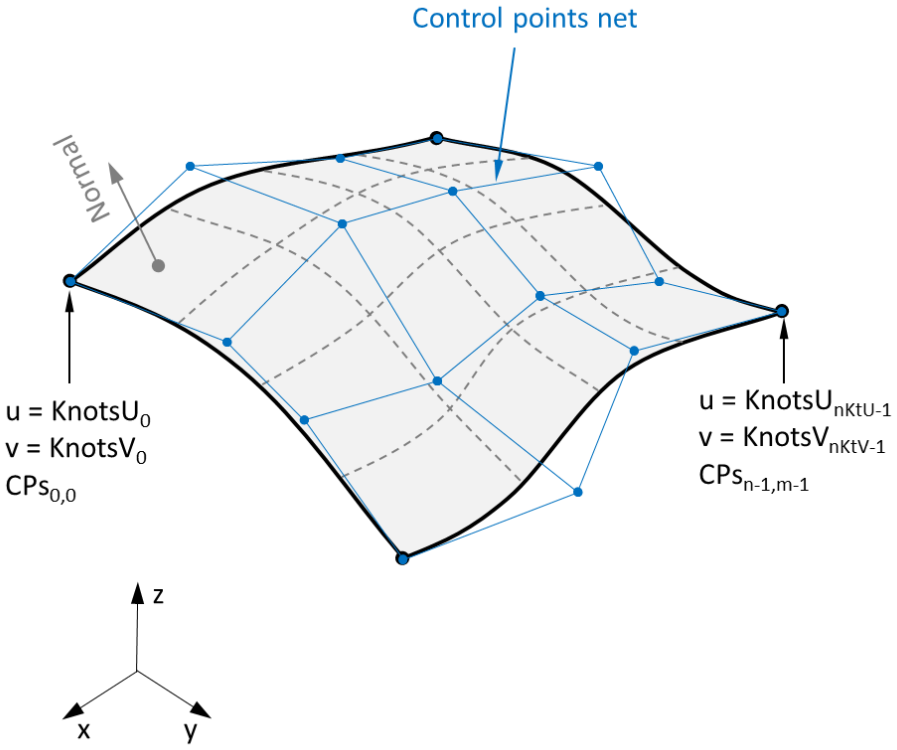


Figure 58 – NURBS Surface

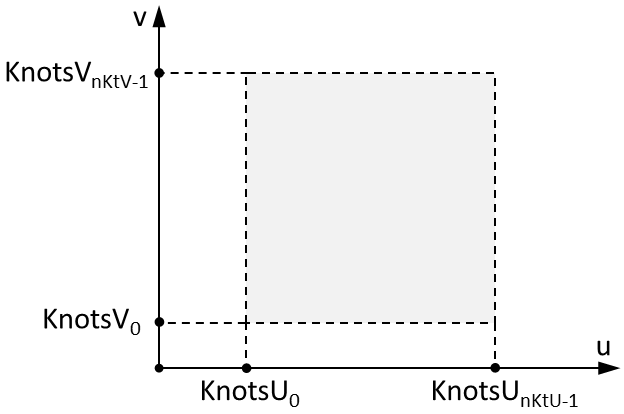


Figure 59 – NURBS Surface (Parameter Space)

A NURBS surface is a bivariate vector-valued piecewise rational function built on the B-spline basis functions and defined by its orders in the u and v directions, two knot vectors (an

increasing sequence of real numbers which divides the parametric space in the intervals called knot spans), and a bidirectional control net with an optional set of associated weights (positive real numbers). If the weights are not defined or equal, the curve is polynomial (otherwise rational).

Function:

$$Nurbs23(u,v)\colon R_2 \rightarrow R_3$$

$$Nurbs23(u,v) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} N_{i,DegreeU}\,(u)N_{j,DegreeV}\,(v)Weights_{i,j}\,CPs_{ij}}{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} N_{i,DegreeU}\,(u)N_{j,DegreeV}\,(v)Weights_{i,j}}$$

$$DegreeU = OrderU - 1, \qquad DegreeV = OrderV - 1$$

$$N_{i,j}(t) - the\ bspline\ basis\ functions$$

$$N_{i,0}(t) = \begin{cases} 1, & t \in [Knots_i, Knots_{i+1}) \\ 0, & otherwise \end{cases}$$

$$N_{i,p}(t) = \frac{t - Knots_i}{Knots_{i+p} - Knots_i} N_{i,p-1}(t) + \frac{Knots_{i+p+1} - t}{Knots_{i+p+1} - Knots_{i+1}} N_{i+1,p-1}(t)$$

$$u \in [KnotsU_0, KnotsU_{nKtU-1}], \qquad n - number\ of\ control\ points\ in\ the\ u\ direction$$

$$v \in [KnotsV_0, KnotsV_{nKtV-1}], \qquad m - number\ of\ control\ points\ in\ the\ v\ direction$$

$$nKtU = n + OrderU, \qquad nKtV = m + OrderV$$

$$CPs = \{\underbrace{CPs_{0,0}, \dots, CPs_{n-1,0}}_{n}, \dots, \underbrace{CPs_{0,m-1}, \dots, CPs_{n-1,m-1}}_{n}\}$$

$$Weights = \{\underbrace{Weights_{0,0}, \dots, Weights_{n-1,0}}_{n}, \dots, \underbrace{Weights_{0,m-1}, \dots, Weights_{n-1,m-1}}_{n}\}$$

Fields:

Field Name	Data Type	Description
Nurbs23Core/OrderU	NaturalType	The order in the u direction (= degree + 1).
Nurbs23Core/OrderV	NaturalType	The order in the v direction (= degree + 1).
Nurbs23Core/KnotsU	ArrayDoubleType	The knot vector in the u direction. The size of the knot vector is 'number of control points in the u direction' + 'order in the u direction'.
Nurbs23Core/KnotsV	ArrayDoubleType	The knot vector in the v direction. The size of the knot vector is 'number of control points in the v direction' + 'order in the v direction'.
Nurbs23Core/CPs or Nurbs23Core/CPsBinary	ArrayPointType or ArrayBinaryType	The array of control points. The size of this array is ('number of the u-knots' - 'the u-order') * ('number of the v-knots' - 'the v-order').

Nurbs23Core/Weights	ArrayDoubleType	The array of weights associated with control points (positive real numbers).

Example:

```
<Nurbs23 id="102">
  <Nurbs23Core>
    <OrderU>4</OrderU>
    <OrderV>4</OrderV>
    <KnotsU N="8">
      0 0 0 0 1 1 1 1
    </KnotsU>
    <KnotsV N="8">
      0 0 0 0 1 1 1 1
    </KnotsV>
    <CPs N="16">
      -1.165 -0.787 0.511
      -0.551 -0.551 0.944
      -0.216 -0.551 0.196
      0.009 -0.521 0.196
      -1.377 0 -0.118
      -0.748 0 0.314
      -0.413 0 0.145
      -0.098 0 0.204
      -1.181 0 -0.590
      -0.511 0 0.354
      -0.206 0 -0.098
      0.118 0 -0.275
      -1.377 0.551 -1.102
      -0.826 0 -0.826
      -0.511 0 -0.944
      -0.177 0 -0.954
    </CPs>
  </Nurbs23Core>
</Nurbs23>
```

7.2.4.11 Offset Surface

Offset23 describes an offset parametric surface as shown in Figure 60 and Figure 61.

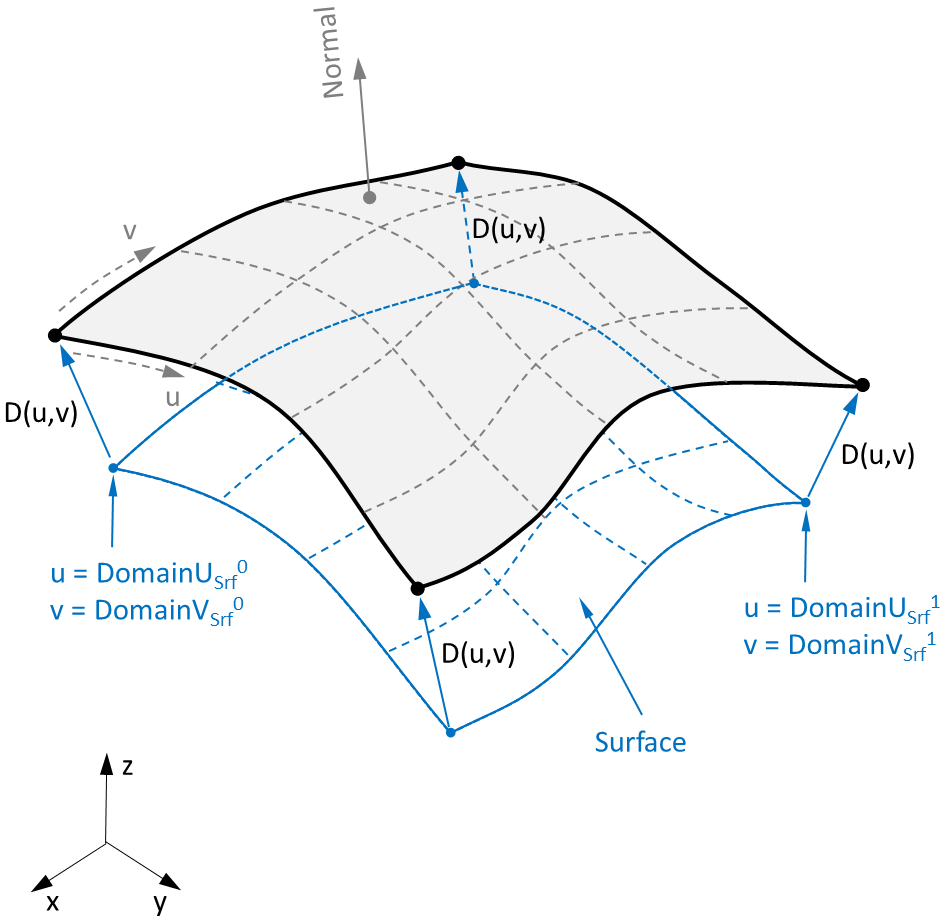


Figure 60 – Offset Surface

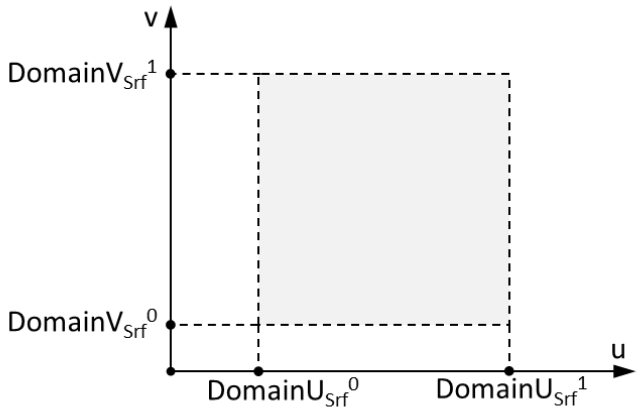


Figure 61 – Offset Surface (Parametric Space)

The offset surface is constructed from an existing surface by moving every surface point in the normal direction by an offset distance.

Function:

$$Offset23(u,v)\colon R_2 \rightarrow R_3$$

$$Offset23(u,v) = Surface(u,v) + D(u,v)$$

$$D(u,v) = N_{srf}(u,v)Distance$$

$$N_{srf}(u,v) = \frac{Surface_u^{'} \times Surface_v^{'}}{\|Surface_u^{'} \times Surface_v^{'}\|}$$

$$u \in \left[domainU_{srf}^{0}, domainU_{srf}^{1}\right], \qquad v \in \left[domainV_{srf}^{0}, domainV_{srf}^{1}\right]$$

Fields:

Field Name	Data Type	Description
Offset23Core/Distance	xs:double	The offset distance.
Offset23Core/Surface	SurfaceCoreType	The base surface for offsetting.

Example:

```
<Offset23 id="97">
  <Offset23Core>
    <Distance>0.2</Distance>
    <Surface>
      <Nurbs23Core>
        <OrderU>4</OrderU>
        <OrderV>4</OrderV>
        <KnotsU N="8">
```

```
          0 0 0 0 1 1 1 1
        </KnotsU>
        <KnotsV N="8">
          0 0 0 0 1 1 1 1
        </KnotsV>
        <CPs N="16">
          -1.165 -0.787 0.511
          -0.551 -0.551 0.944
          -0.216 -0.551 0.196
          0.009 -0.521 0.196
          -1.377 0 -0.118
          -0.748 0 0.314
          -0.413 0 0.145
          -0.098 0 0.204
          -1.181 0 -0.590
          -0.511 0 0.354
          -0.206 0 -0.098
          0.118 0 -0.275
          -1.377 0.551 -1.102
          -0.826 0 -0.826
          -0.511 0 -0.944
          -0.177 0 -0.954
        </CPs>
      </Nurbs23Core>
    </Surface>
  </Offset23Core>
</Offset23>
```


7.2.5 Mesh Curves

7.2.5.1 Triangulation Path

PathTriangulation describes a path of triangulation edges. The way the identification of triangle edges relate to the identification of triangle vertices is illustrated in Figure 62.

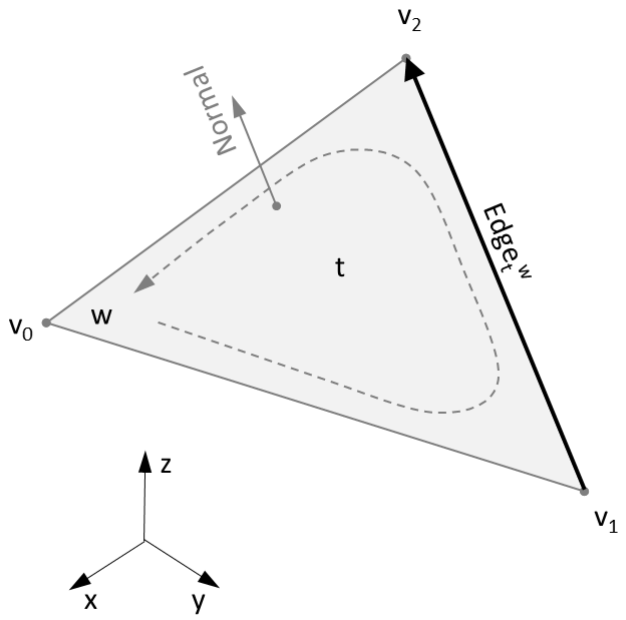


Figure 62 – The edge ‘w’ of triangle ‘t’

$$E_t^w - the\ edge\ 'w'\ of\ the\ triangle\ 't'\ is\ opposite\ to\ the\ vertex\ 'w'$$

$$E_t^w = \begin{cases} Start\ point = v_1 = Vertices_a, & a = Triangles_t^{(w+1)\ mod\ 3} \\ End\ point = v_2 = Vertices_b, & b = Triangles_t^{(w+2)\ mod\ 3} \end{cases}$$

$$Triangles_t^w = vertex\ w\ in\ triangle\ t$$

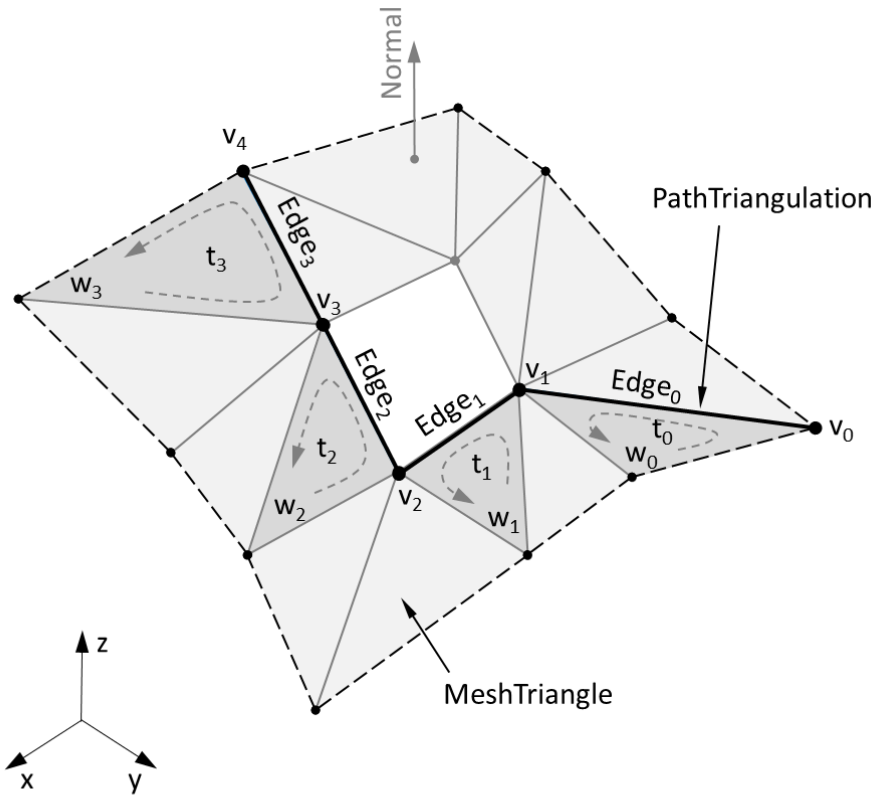


Figure 63 – Triangulation Path

$$Edges = \{\underbrace{t_0, w_0}_{Edge_0}, \underbrace{t_1, w_1}_{Edge_1}, \underbrace{t_2, w_2}_{Edge_2}, \underbrace{t_3, w_3}_{Edge_3}\}$$

$$V = \{v_0, v_1, v_2, v_3, v_4\}$$

The triangulation path is a polyline formed from the mesh edges as shown in Figure 63.

Function:

$$Edges = \{\underbrace{t_0, w_0}_{Edge_0}, \ldots, \underbrace{t_{n-1}, w_{n-1}}_{Edge_{n-1}}\}, \qquad n - number\ of\ edges$$

$$Pairs\ (t_i, w_i)\ specifies\ edges\ E_{t_i}^{w_i}$$

$$V - sequence\ of\ vertices\ in\ the\ path$$

$$V = \{v_0, \ldots, v_n\}$$

$$v_i = Vertices_j, \qquad j = \begin{cases} Triangles_0^{(w_0+1)\ mod\ 3} & i = 0 \\ Triangles_i^{(w_i+1)\ mod\ 3} = Triangles_{i-1}^{(w_{i-1}+2)\ mod\ 3}, & 0 < i < n \\ Triangles_{n-1}^{(w_{n-1}+2)\ mod\ 3}, & i = n \end{cases}$$

Fields:

Field Name	Data Type	Description
PathTriangulationCore/Edges or PathTriangulationCore/EdgesBinary	ArrayI2Type or ArrayBinaryType	The array of triangle edges which forms a triangulation path. This is an array of pairs of positive integers, where the first value is a triangle index and the second value is a vertex index opposite to the edge.
MeshTriangle	ElementReferenceType	The identifier of a triangle mesh.

Example:

```
<PathTriangulation id="32">
  <PathTriangulationCore>
    <Edges N="2">
      1 0
      0 2
    </Edges>
  </PathTriangulationCore>
  <MeshTriangle>
    <Id>98</Id>
  </MeshTriangle>
</PathTriangulation>

<MeshTriangle id="98">
  <MeshTriangleCore>
    <Triangles N="2">
      0 1 2
      2 3 0
    </Triangles>
    <Neighbours N="2">
      -1 1 -1
      -1 0 -1
    </Neighbours>
    <Vertices N="4">
      0.0 0.0 0.0
      10.0 0.0 0.0
      10.0 5.0 0.0
      0.0 5.0 0.0
    </Vertices>
  </MeshTriangleCore>
</MeshTriangle>
```

7.2.6 Mesh Surfaces

7.2.6.1 Triangulation Mesh Surface

MeshTriangle23 describes a triangulation mesh surface. Figure 64 illustrates how two triangles may be sewn together.

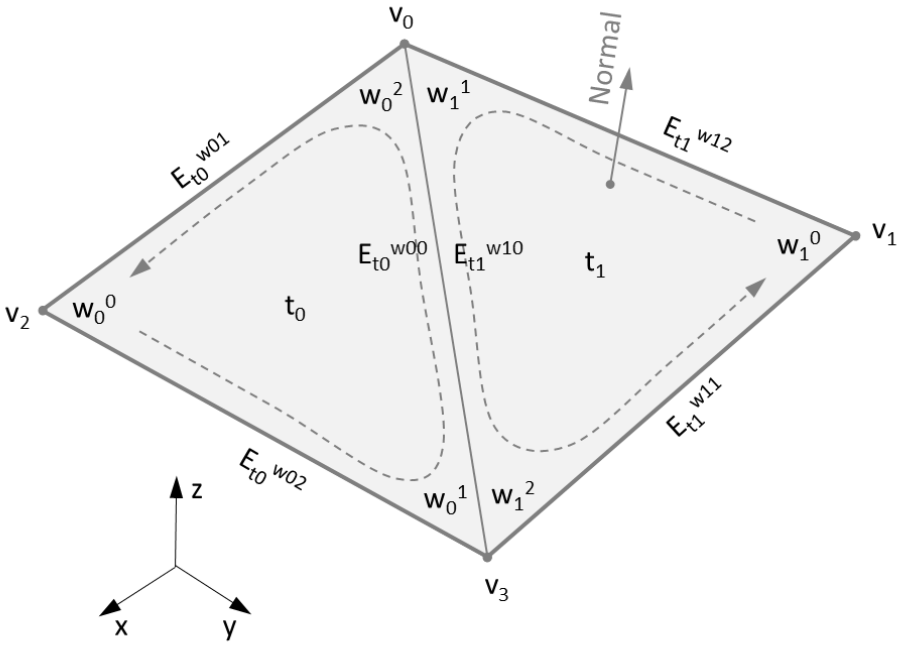


Figure 64 – Two sewn triangles

$$t_0 = Neighbours_{t_1}^{w_1^0}, \qquad t_1 = Neighbours_{t_0}^{w_0^0}$$

$$w_t^i - index\ of\ vertex\ of\ triangle\ t$$

$$w_t^i \in \{0,1,2\}, \qquad i \in \{0,1,2\}, \qquad t - index\ of\ triangle$$

$$w_t^1 = (w_t^0 + 1)\ mod\ 3, \qquad w_t^2 = (w_t^0 + 2)\ mod\ 3$$

$$\begin{cases} v_0 = Vertices_i = Vertices_j, & i = Triangles_{t_0}^{w_0^2}, j = Triangles_{t_1}^{w_1^1} \\ v_1 = Vertices_i, & i = Triangles_{t_1}^{w_1^0} \\ v_2 = Vertices_i, & i = Triangles_{t_0}^{w_0^0} \\ v_3 = Vertices_i = Vertices_j, & i = Triangles_{t_0}^{w_0^1}, j = Triangles_{t_1}^{w_1^2} \end{cases}$$

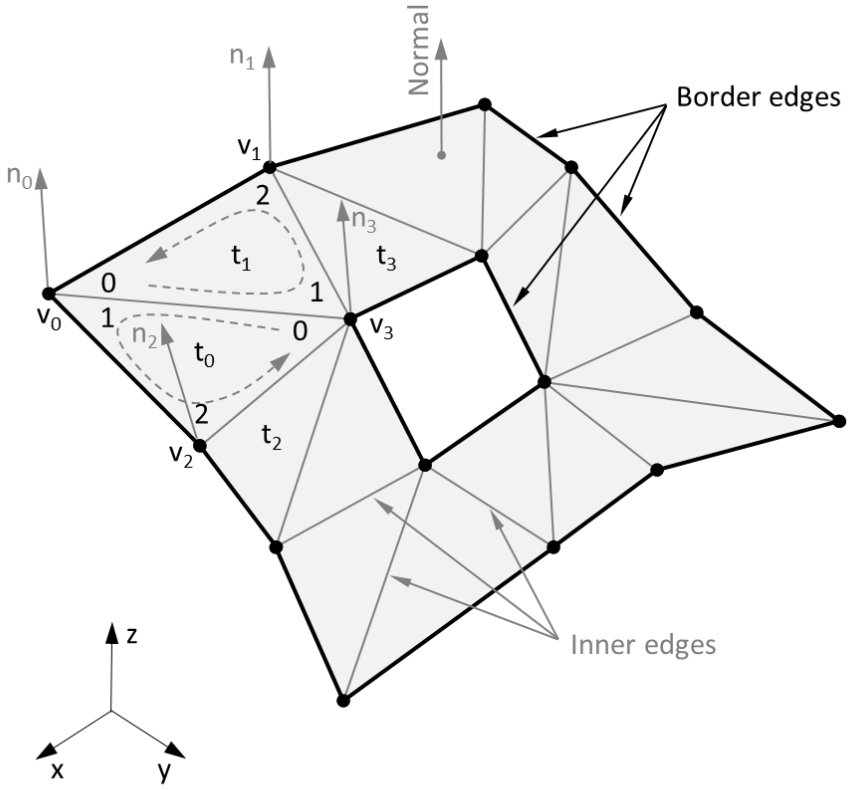


Figure 65 – Triangle Mesh

$$Triangles_{t_0} = \begin{bmatrix} v_3 \\ v_0 \\ v_2 \end{bmatrix}, \qquad Triangles_{t_1} = \begin{bmatrix} v_0 \\ v_3 \\ v_1 \end{bmatrix}$$

$$Neighbours_{t_0} = \begin{bmatrix} -1 \\ t_2 \\ t_1 \end{bmatrix}, \qquad Neighbours_{t_1} = \begin{bmatrix} t_3 \\ -1 \\ t_0 \end{bmatrix}$$

A mesh surface is a set of triangles connected by their common edges as shown in Figure 65.

Function:

$$Triangles_t^w = the\ vertex\ 'w'\ in\ the\ triangle\ 't'$$

$$E_t^w - the\ edge\ \ 'w'\ in\ the\ triangle 't'\ is\ opposite\ to\ the\ vertex\ 'w'$$

$$E_t^w = \begin{cases} Start\ point = Vertices_a, & a = Triangles_t^{(w+1)\ mod\ 3} \\ End\ point = Vertices_b, & b = Triangles_t^{(w+2)\ mod\ 3} \end{cases}$$

$$Neighbours_t^w = \begin{cases} neighbour\ triangle\ index, & E_t^w\ is\ a\ border\ edge \\ -1, & otherwise \end{cases}$$

$$w - vertex\ index\ in\ a\ triangle, \qquad w \in \{0,1,2\}$$

$$N_t - unit\ normal\ of\ the\ triangle\ 't'$$

$$\begin{cases} N_t = \dfrac{(Vertices_{v_1} - Vertices_{v_0}) \times (Vertices_{v_2} - Vertices_{v_0})}{\left\|(Vertices_{v_1} - Vertices_{v_0}) \times (Vertices_{v_2} - Vertices_{v_0})\right\|} \\ v_i = Triangles_t^i \\ i \in \{0,1,2\} \end{cases}$$

Fields:

Field Name	Data Type	Description
MeshTriangleCore/Triangles or MeshTriangleCore/TrianglesBinary	ArrayI3Type or ArrayBinaryType	The array of indices of the triangle vertices. The number of array elements corresponds to the number of triangles in the mesh. Each element of this array is a triplet of integer numbers: index of the first vertex, index of the second vertex and index of the third vertex. All three vertex indices of a triangle must be different and must lie in the range [0, ..., number of vertices - 1].
MeshTriangleCore/Neighbours or MeshTriangleCore/NeighboursBinary	ArrayI3Type or ArrayBinaryType	The array of indices of the triangle neighbors. The number of array elements corresponds to the number of triangles in the mesh. Each element of this array is a triplet of integer numbers: index of a neighbor triangle opposite to the first triangle vertex, index of a neighbor triangle opposite to the second triangle vertex, index of a neighbor triangle opposite to the third triangle vertex. There is a special index value "-1" which shows that there is no neighbor. The neighbor indices must lie in the range [-1, ..., number of triangles - 1].
MeshTriangleCore/Vertices or MeshTriangleCore/VerticesBinary	ArrayPointType or ArrayBinaryType	The array of 3D points. The number of array elements corresponds to the number of vertices in the mesh. Each element of this array is a triplet of real numbers: the X-coordinate, the Y-coordinate and the Z-coordinate.
MeshTriangleCore/Normals or MeshTriangleCore/NormalsBinary	ArrayUnitVectorType or ArrayBinaryType	The mesh normals - an array of unit vectors. The number of array elements corresponds to the number of vertices in the mesh. Each element of this array is a triplet of real numbers: the X-component, the Y-component and the Z-

		component.

Example:

```
<MeshTriangle id="98">
  <MeshTriangleCore>
    <Triangles N="2">
      0 1 2
      2 3 0
    </Triangles>
    <Neighbours N="2">
      -1 1 -1
      -1 0 -1
    </Neighbours>
    <Vertices N="4">
      0.0 0.0 0.0
      10.0 0.0 0.0
      10.0 5.0 0.0
      0.0 5.0 0.0
    </Vertices>
  </MeshTriangleCore>
</MeshTriangle>
```

7.3 Topology

QIF topology elements define boundaries of geometry objects and specify connectivity relations. In QIF they are separated by type and organized in the following sets:

- VertexSet
- EdgeSet
- LoopSet
- FaceSet
- ShellSet
- BodySet
- PointCloudSet

As seen in Figure 66, all topology types are derived from TopologyBaseType.

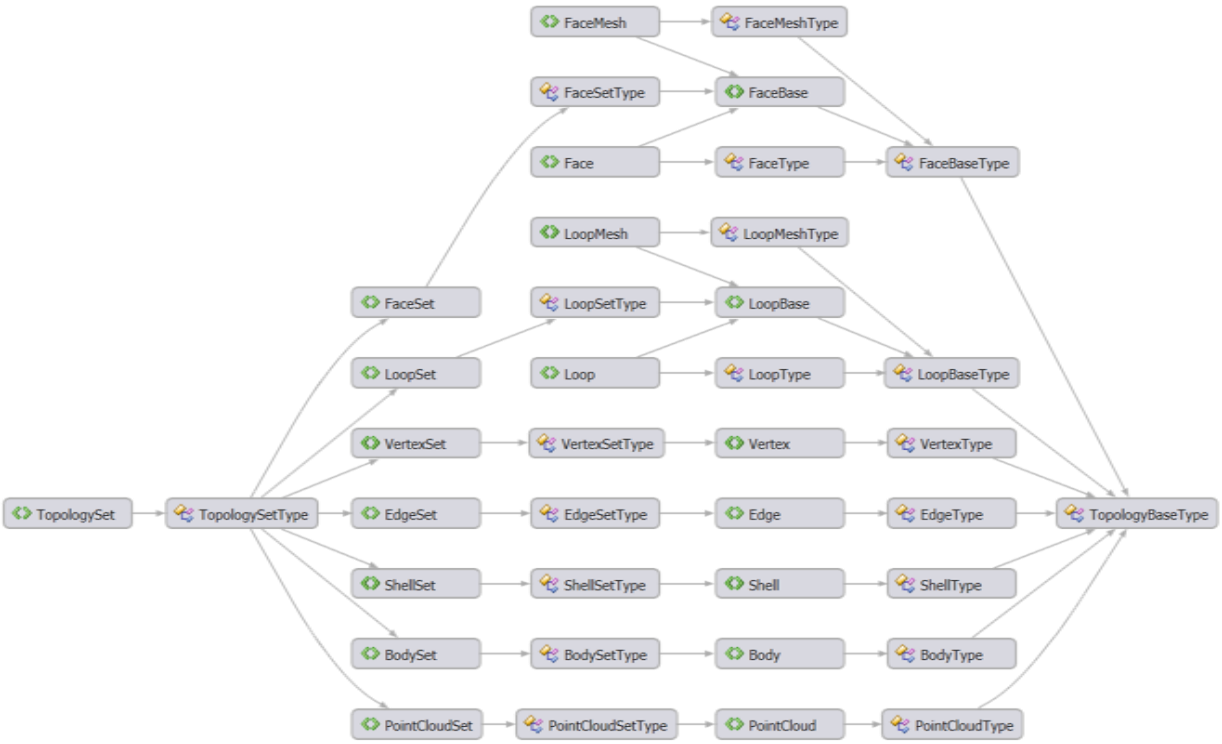


Figure 66 – Topology Types

The main topological items are:

- Body - a set of oriented shells built on collections of connected face elements
- Shell - a set of connected faces
- Face - a bounded portion of a surface
- Loop - a circuit of edges bounding a face
- Edge - a bounded piece of a 3D curve
- Vertex - lies at a point and can be used to bound an edge

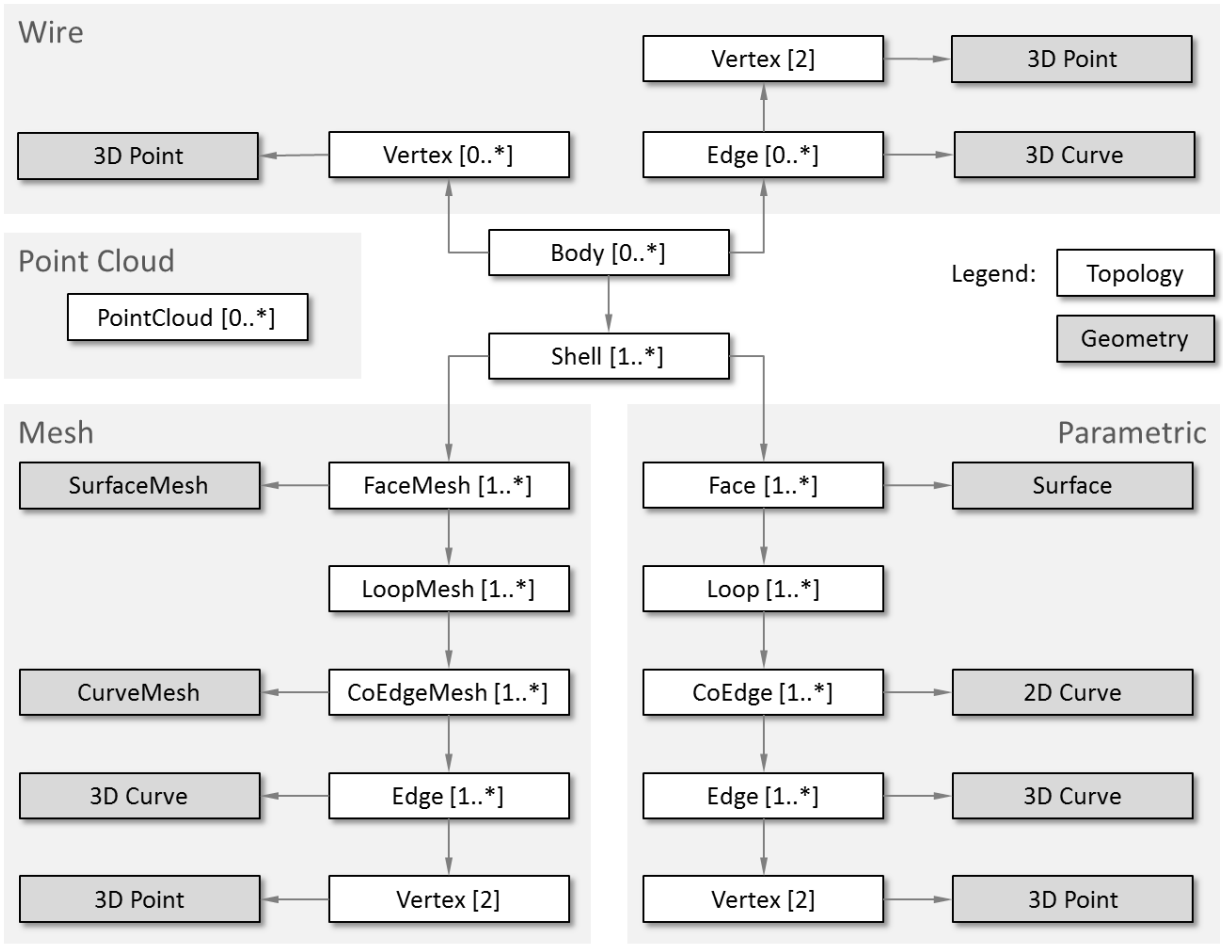


Figure 67 – Boundary Representation

All topology objects contain the following fields:

Field Name	Data Type	Description
@id	QIFIdType	The unique model entity identifier.
@label	xs:string	The model entity "nameplate". Normally it can be seen at the entity item in the model tree.
@color	ColorType	The RGB color property of a model entity.
@transparency	xs:double	The transparency property of a model entity.
@hidden	xs:boolean	The visibility property of a model entity in the graphical window.
@size	xs:double	A recommended size for visualization of an infinite drawable element.
Attributes	AttributesType	User-defined attributes (typified, binary array, or XML structured).

7.3.1 Vertex

Vertex describes a topological vertex as illustrated in Figure 68.

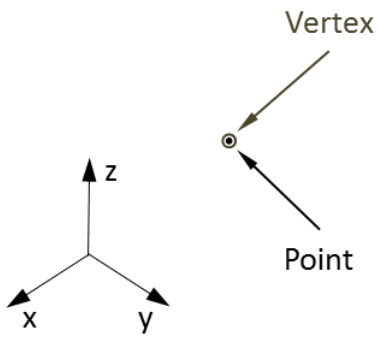


Figure 68 – Vertex

A vertex lies at a 3D point and is normally used to bound an edge.

Fields:

Field Name	Data Type	Description
@tolerance	xs:double	The tolerance value which is calculated as the maximum distance from the vertex underlying 3D point to the ends of all neighboring edges that are terminated in the neighborhood of this vertex. This value can be defined only for the case of the tolerant body.
Point	ElementReferenceType	The identifier of a 3D point – the underlying geometry of the vertex.

Example:

```
<Vertex id="401">
  <Point>
    <Id>398</Id>
  </Point>
</Vertex>
```


7.3.2 Edge

Edge describes a topological edge as shown in Figure 69.

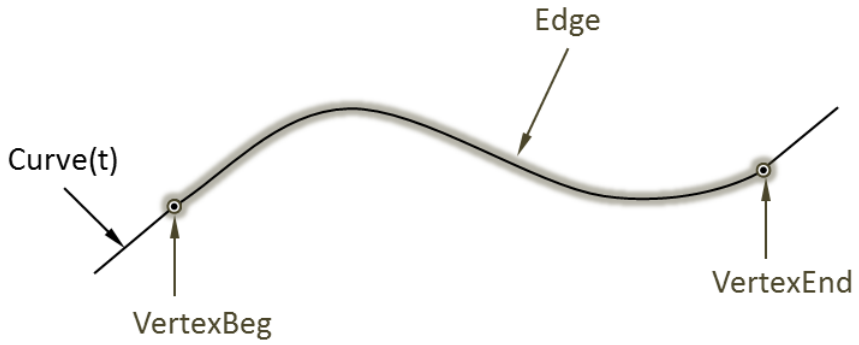


Figure 69 – Edge

An edge is a bounded piece of a 3D curve.

Fields:

Field Name	Data Type	Description
@tolerance	xs:double	The tolerance value calculated as the maximum distance between the 2D parametric co-edges of the neighboring faces. The edge tolerance for the case of tolerant body.
Curve	ElementReferenceType	The identifier of a 3D curve that is the underlying geometry of this edge.
VertexBeg	ElementReferenceType	The identifier of the vertex which bounds this edge at the beginning of the edge. The 'underlying' parameter of VertexBeg must be less than the 'underlying' parameter of VertexEnd. Or, in other words, the edge always follows the natural parameterization of the underlying 3D curve. If there is a need to pass an edge in the opposite (to the natural parameterization of the underlying curve) direction then the corresponding flag must be defined at the loop level.
VertexEnd	ElementReferenceType	The identifier of the vertex which bounds this edge at the end of the edge. The 'underlying' parameter of VertexEnd must be bigger than the 'underlying' parameter of VertexBeg. Or, in other words, the edge always follows the natural parameterization of the underlying 3D curve. If there is a need to pass an edge in the opposite (to the natural parameterization of the underlying curve) direction then the corresponding flag must be defined at the loop level.

Example:

```
<Edge id="405">
```

```
  <Curve>
    <Id>390</Id>
  </Curve>
  <VertexBeg>
    <Id>566</Id>
  </VertexBeg>
  <VertexEnd>
    <Id>401</Id>
  </VertexEnd>
</Edge>
```


7.3.3 Loop

Loop describes a topological loop as shown in Figure 70.

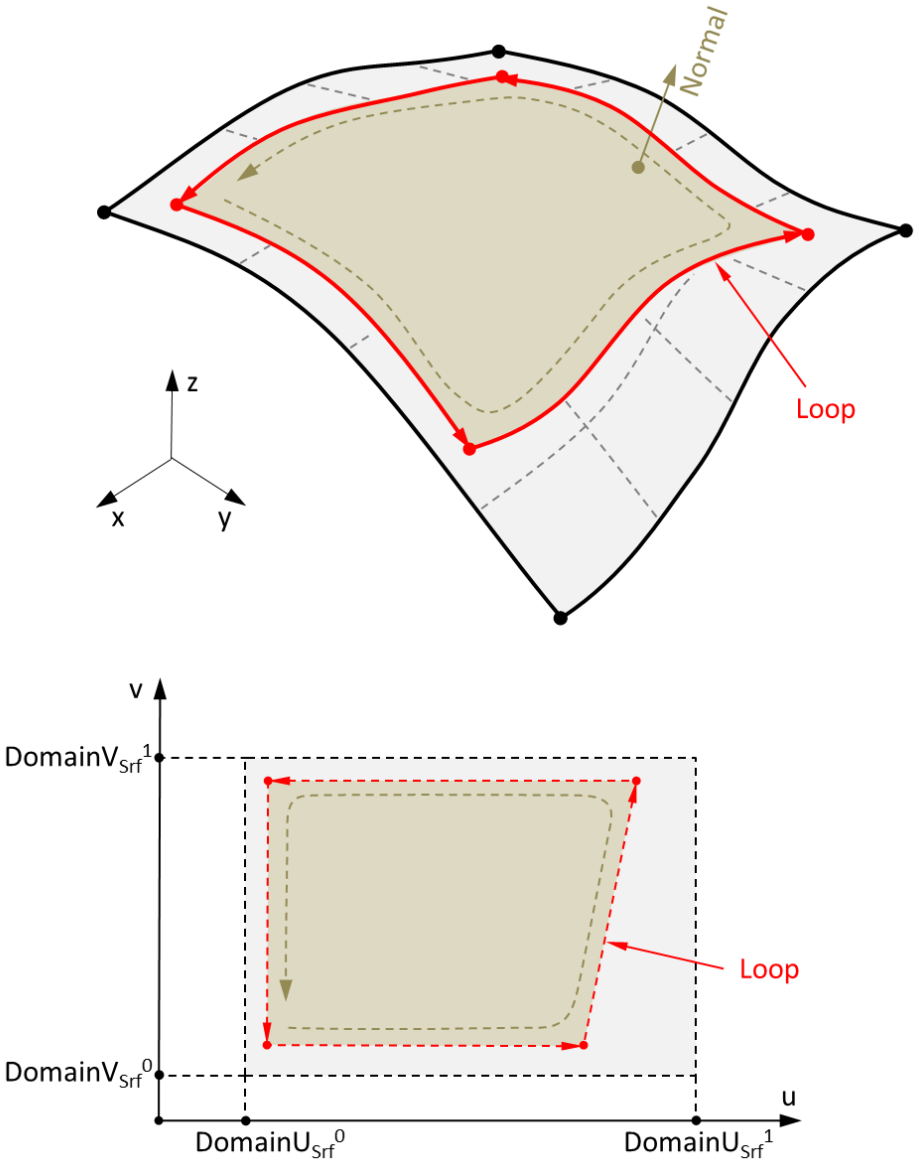


Figure 70 – Loop

A loop is a circuit of edges (in 3D space) with corresponding CoEdges (in surface parameter space) bounding a face. CoEdges are shown in Figure 71.

Fields:

Field Name	Data Type	Description
@form	LoopFormEnumType	The loop type which can take one of the following values: 'UNKNOWN', 'OUTER', 'INNER' or

		'SLIT'.
CoEdges	CoEdgesType	An array of co-edges which form the loop.
CoEdges/CoEdge/EdgeOriented/@turned	xs:boolean	This flag shows if the referenced edge must be inverted.
CoEdges/CoEdge/EdgeOriented/Id	QIFReferenceType	A reference to an edge.
CoEdges/CoEdge/Curve12	ElementReferenceType	A reference to a 2D curve which is a projection of the edge underlying 3D Curve onto the face underlying surface.

Example:

```
<Loop id="113" form="OUTER">
  <CoEdges N="4">
    <CoEdge>
      <EdgeOriented>
        <Id>405</Id>
      </EdgeOriented>
      <Curve12>
        <Id>520</Id>
      </Curve12>
    </CoEdge>
    <CoEdge>
      <EdgeOriented>
        <Id>469</Id>
      </EdgeOriented>
      <Curve12>
        <Id>532</Id>
      </Curve12>
    </CoEdge>
    <CoEdge>
      <EdgeOriented turned="1">
        <Id>511</Id>
      </EdgeOriented>
      <Curve12>
        <Id>547</Id>
      </Curve12>
    </CoEdge>
    <CoEdge>
      <EdgeOriented>
        <Id>567</Id>
      </EdgeOriented>
      <Curve12>
        <Id>563</Id>
      </Curve12>
    </CoEdge>
  </CoEdges>
</Loop>
```


7.3.3.1 Co-Edges

A CoEdge defines a parameter space curve (i.e. the projection of an underlying 3D Curve of an Edge onto an underlying surface of a face) which represents a part of the face trimming loop and additionally includes the corresponding edge orientation flag.

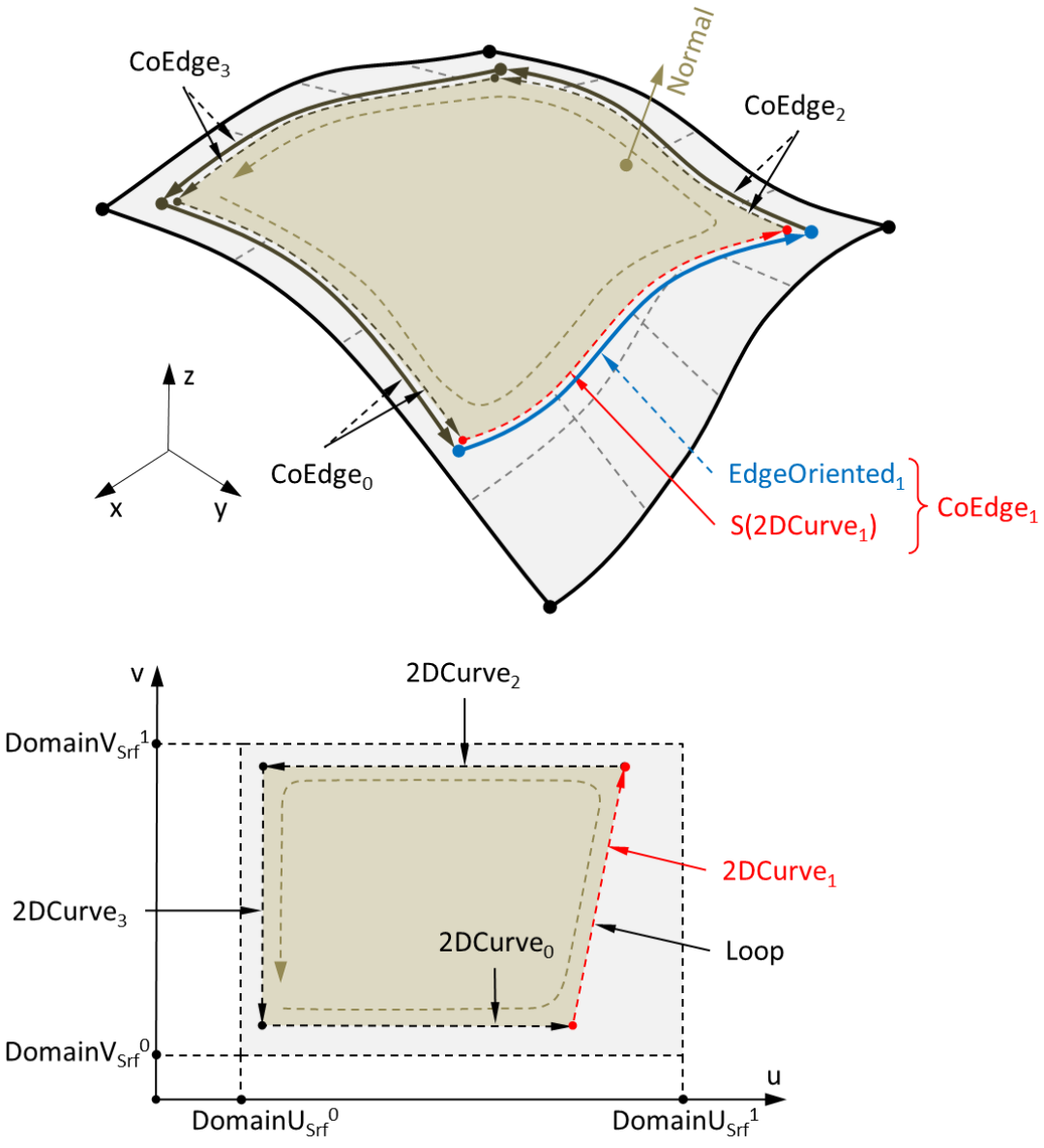


Figure 71 – Co-Edge

7.3.3.2 Loop forms

The QIF data model allows defining of the three loop forms: 'OUTER' (shown in Figure 72), 'INNER' (shown in Figure 73) and ‘SLIT’ (shown Figure 74).

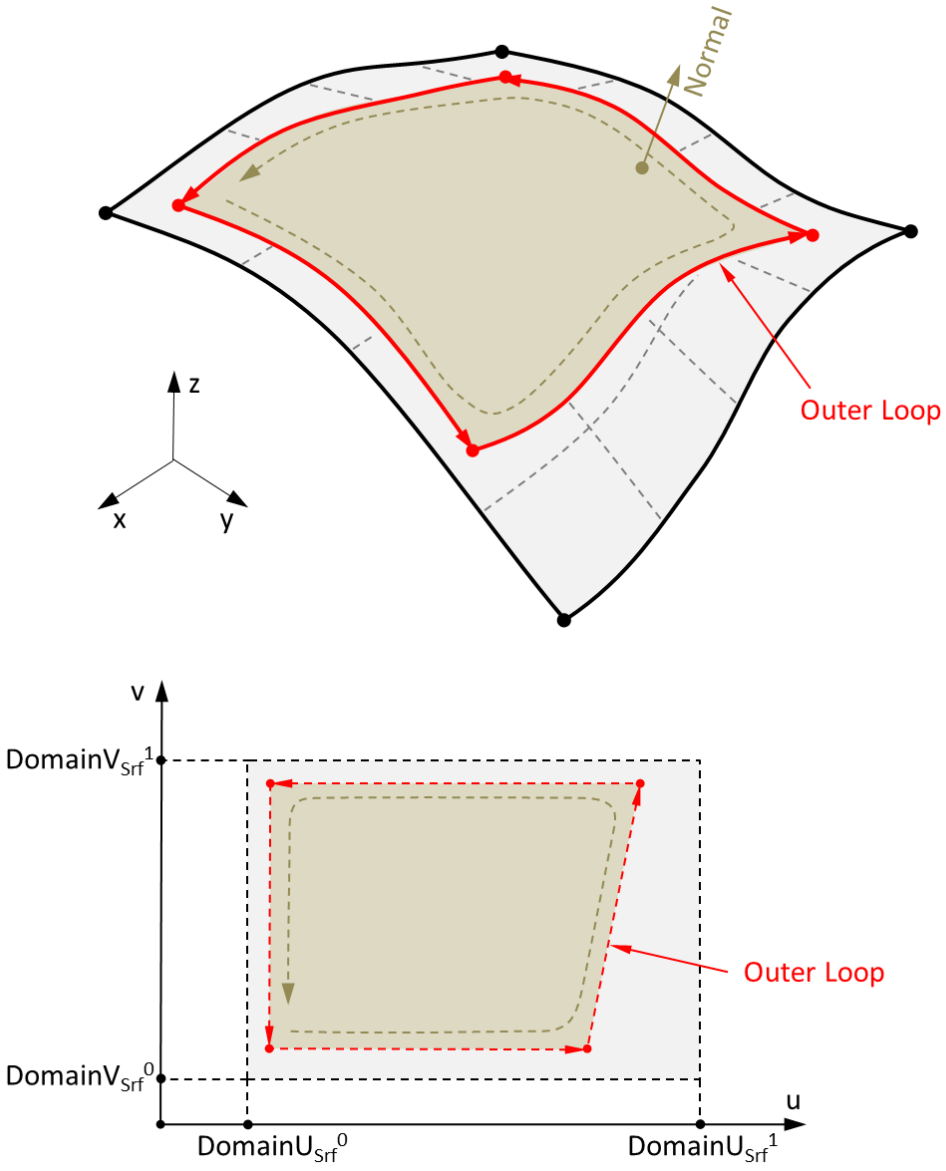


Figure 72 – Outer Loop

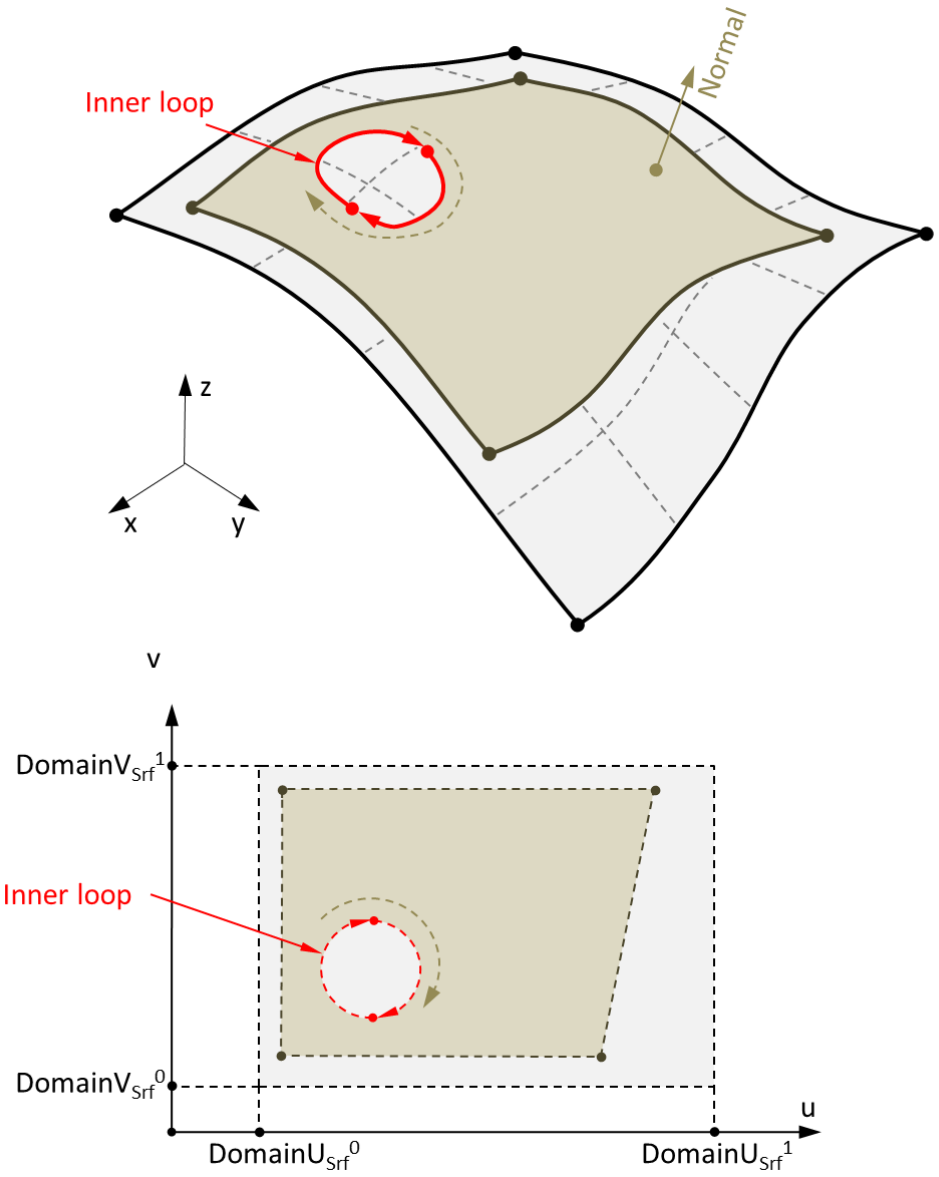


Figure 73 – Inner Loop

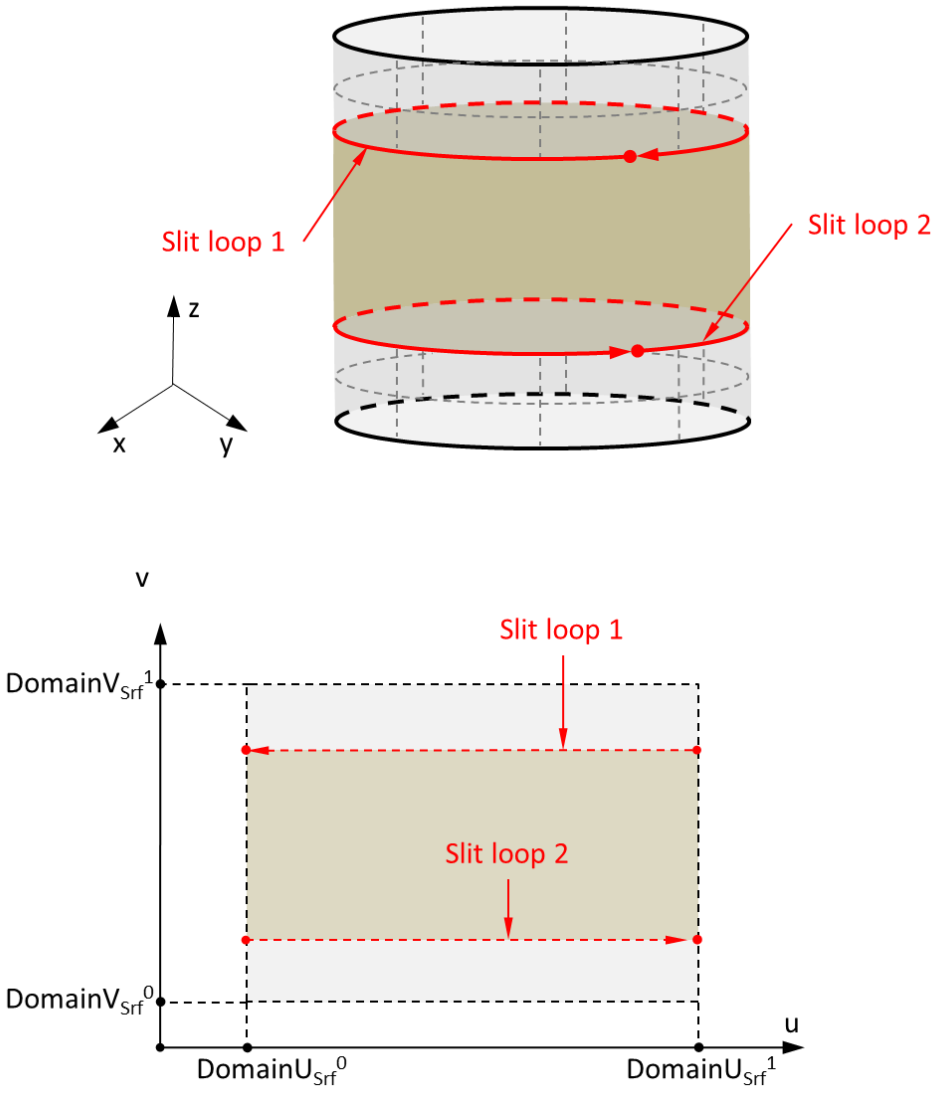


Figure 74 – Slit Loop

7.3.4 Mesh Loop

LoopMesh describes a topological mesh loop as shown in Figure 75.

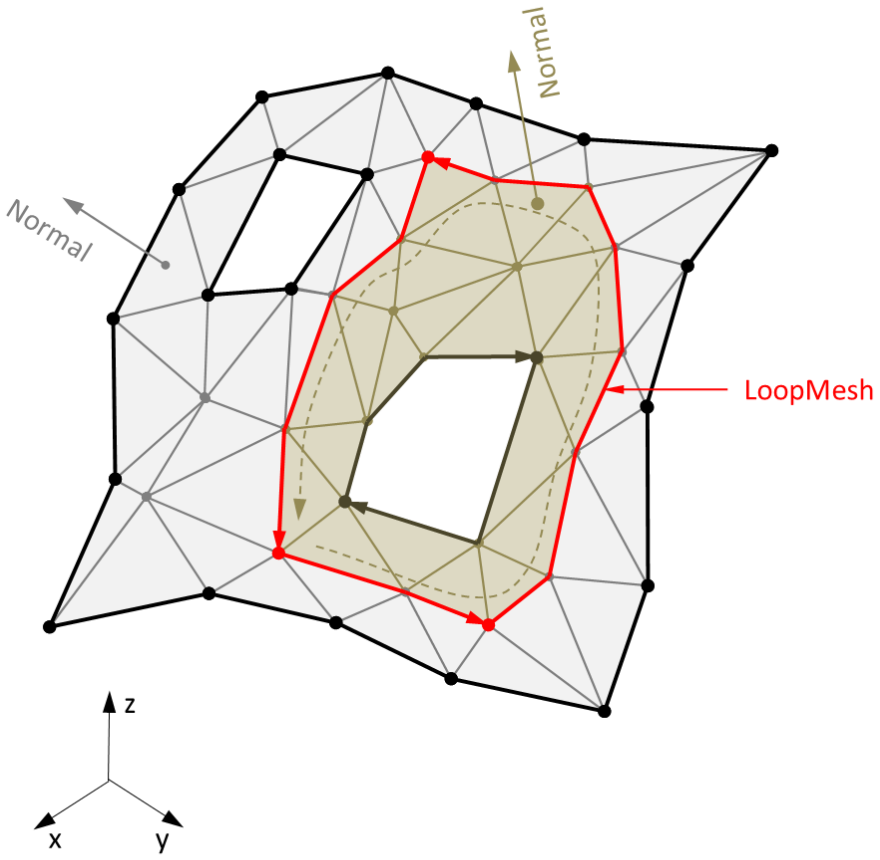


Figure 75 – Mesh Loop

A mesh loop is a circuit of CoEdges bounding a mesh face as shown in Figure 76.

Fields:

Field Name	Data Type	Description
CoEdgesMesh	CoEdgesMeshType	An array of mesh co-edges that forms a mesh loop.
CoEdgesMesh/CoEdgeMesh/EdgeOriented/@turned	xs:boolean	This flag shows if the referenced edge must be reversed from the origin edge orientation.
CoEdgesMesh/CoEdgeMesh/EdgeOriented/Id	QIFReferenceType	A reference to an edge with a given orientation.

CoEdgesMesh/CoEdge/CurveMesh	ElementReferenceType	A reference to a mesh curve. This is the projection of the underlying 3D Curve of an oriented Edge onto an underlying mesh surface of a mesh face.

Example:

```
<LoopMesh id="113">
  <CoEdgesMesh N="4">
    <CoEdgeMesh>
      <EdgeOriented>
        <Id>405</Id>
      </EdgeOriented>
      <CurveMesh>
        <Id>520</Id>
      </CurveMesh>
    </CoEdgeMesh>
    <CoEdgeMesh>
      <EdgeOriented>
        <Id>469</Id>
      </EdgeOriented>
      <CurveMesh>
        <Id>532</Id>
      </CurveMesh>
    </CoEdgeMesh>
    <CoEdgeMesh>
      <EdgeOriented turned="1">
        <Id>511</Id>
      </EdgeOriented>
      <CurveMesh>
        <Id>547</Id>
      </CurveMesh>
    </CoEdgeMesh>
    <CoEdgeMesh>
      <EdgeOriented>
        <Id>567</Id>
      </EdgeOriented>
      <CurveMesh>
        <Id>563</Id>
      </CurveMesh>
    </CoEdgeMesh>
  </CoEdgesMesh>
</LoopMesh>
```


7.3.4.1 Mesh Co-Edges

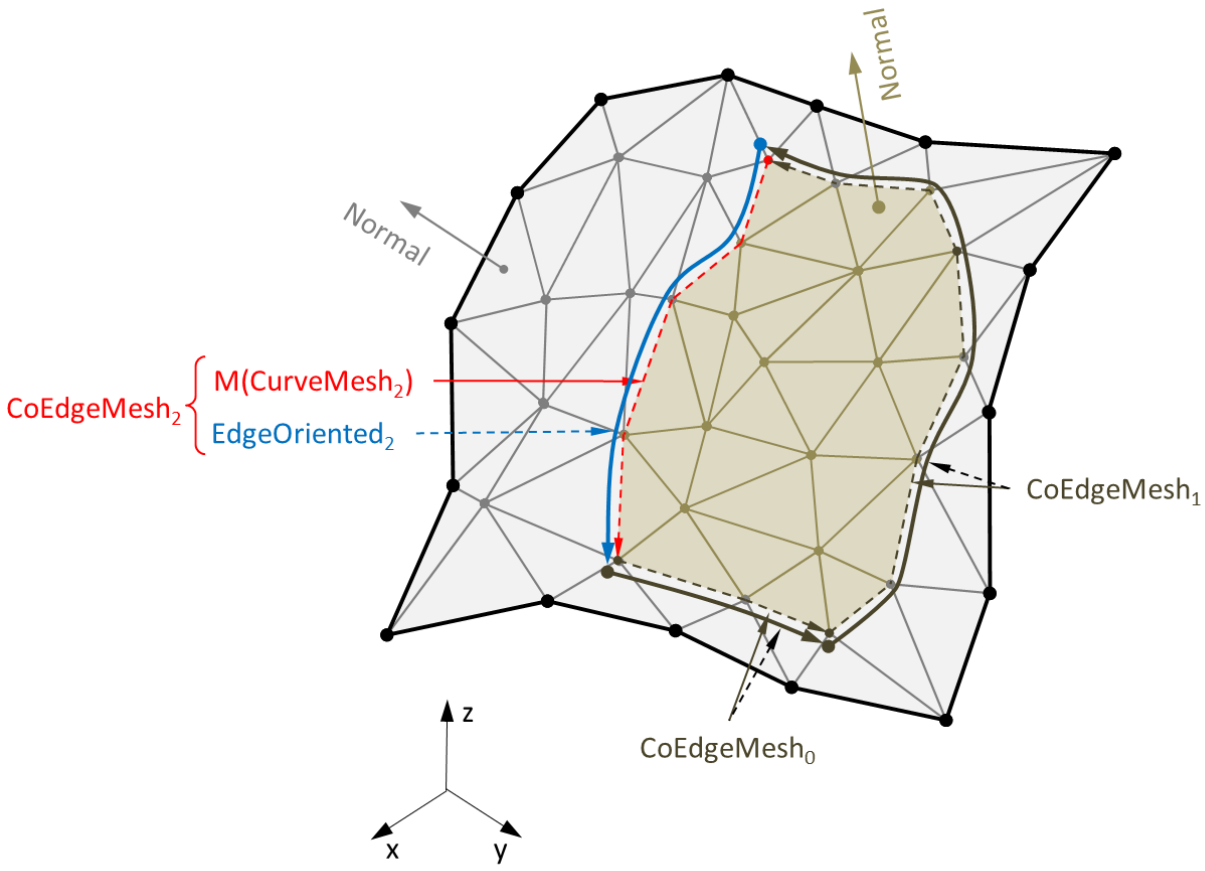


Figure 76 – Mesh Co-Edge

7.3.5 Face

Face describes a topological face as shown in Figure 77.

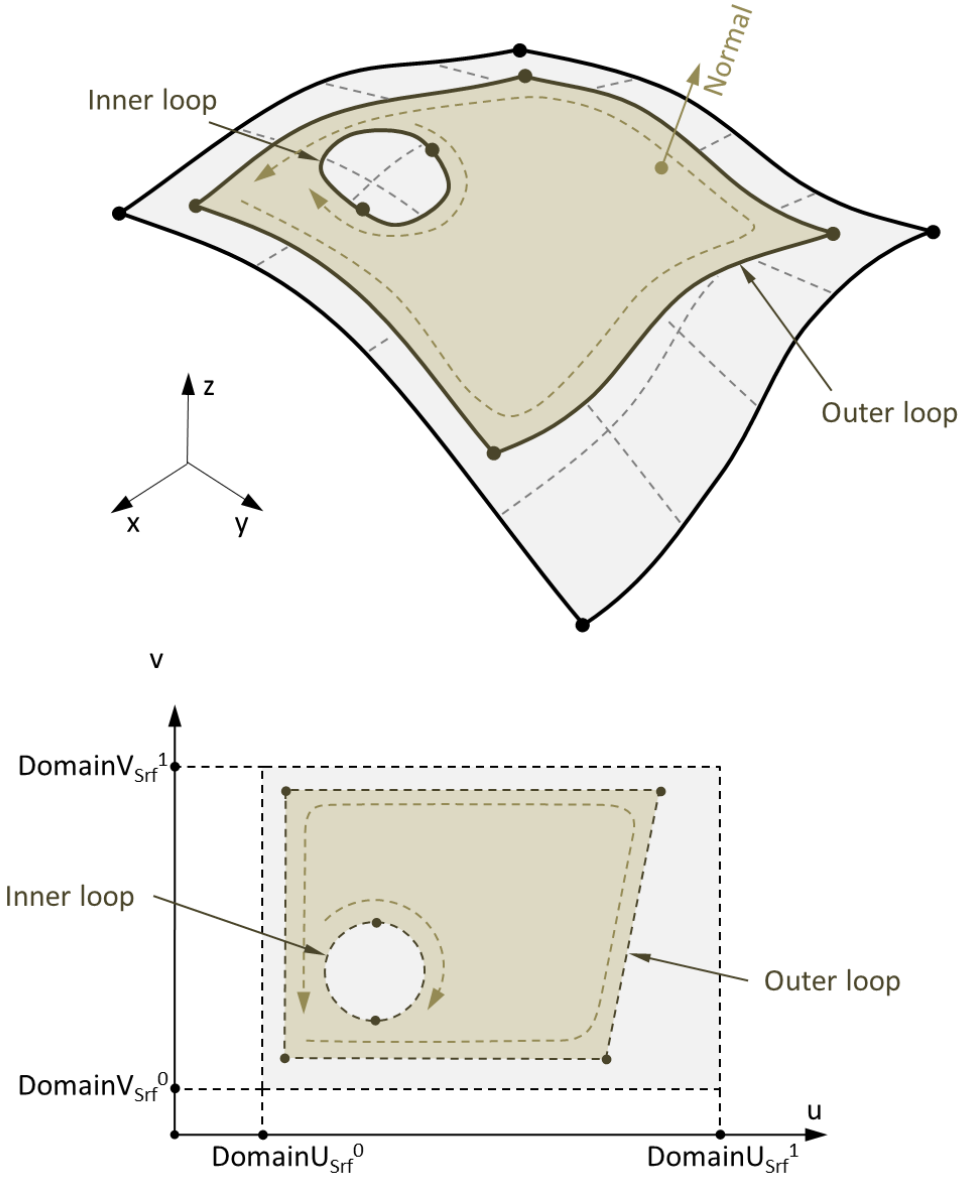


Figure 77 – Face

A face is a bounded portion of a parametric surface.

Fields:

Field Name	Data Type	Description
@turned	xs:boolean	This flag shows if the face normal goes in the opposite direction of the surface normal.

Surface	ElementReferenceType	The identifier of the underlying surface.
LoopIds	ArrayReferenceType	The array of identifiers of the face trimming contours.

Example:

```
<Face id="174" color="191 191 191" turned="1">
  <Surface>
    <Id>102</Id>
  </Surface>
  <LoopIds N="1">
    <Id>113</Id>
  </LoopIds>
</Face>
```

7.3.6 Mesh Face

FaceMesh describes a topological mesh face as shown in Figure 78.

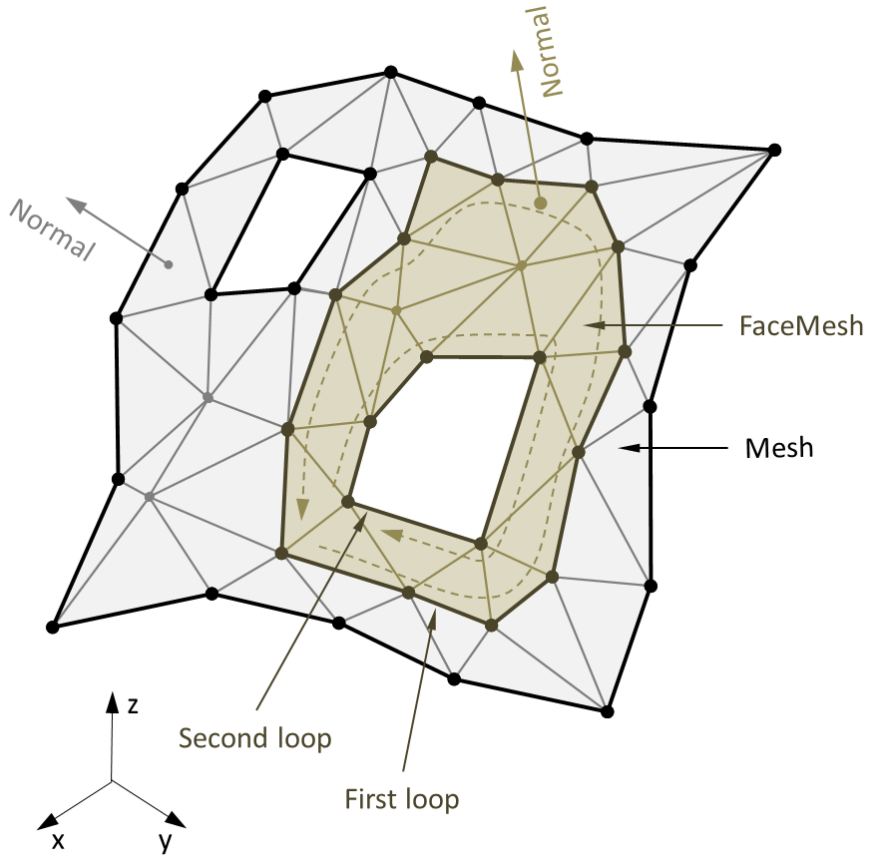


Figure 78 – Mesh Face

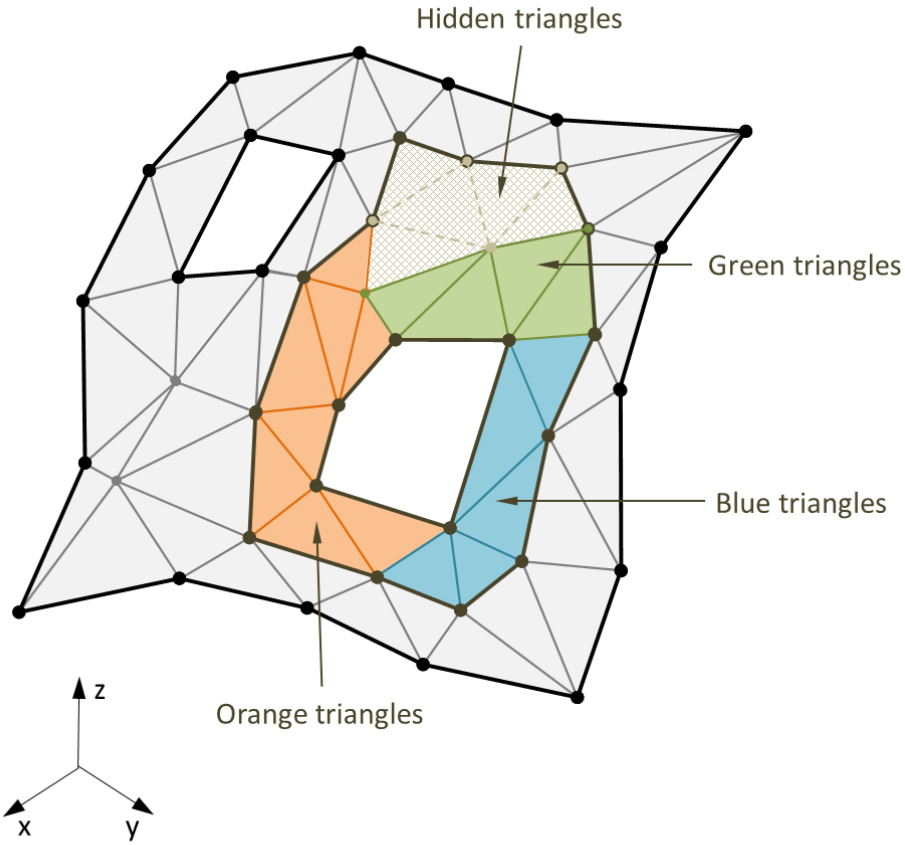


Figure 79 – Mesh Face (Triangle Visibility and Color)

As shown in Figure 79, a mesh face is built on a mesh surface bounded by a set of closed triangulation paths (polylines formed from the triangle edges). The triangles of a mesh face may be colored and they may be hidden. By default, all triangles in a mesh face are visible, and their appearance is the same on both sides.

Fields:

Field Name	Data Type	Description
@turned	xs:boolean	This attribute shows if the face orientation must be reversed from the orientation of the underlying surface. If the value is true, the face orientation must be opposite the surface orientation. If the value is false, the two orientations must be the same.
Mesh	ElementReferenceType	The identifier of the underlying mesh surface.
LoopIds	ArrayReferenceType	An array of identifiers of the face trimming contours.

Triangles or TrianglesBinary	ArrayIntType or ArrayBinaryType	The array of triangle indices of the underlying mesh surface. All elements of this array must be unique and must lie in the range [0, number of triangles in the underlying mesh surface - 1].
TrianglesVisible or TrianglesVisibleBinary or TrianglesHidden or TrianglesHiddenBinary	ArrayIntType or ArrayBinaryType or ArrayIntType or ArrayBinaryType	The indexes from the Triangles array of the triangles that should be visible or hidden. If this element is not used, all triangles are visible. If TrianglesVisible[Binary] is used, all other triangles in the mesh face are hidden. If TrianglesHidden is used, all other triangles in the mesh face are visible.
TrianglesColor or TrianglesColorBinary	ArrayIntType or ArrayBinaryType	An array of unsigned byte values which defines colors of the face interior triangles. Each element of this array is a triplet of unsigned byte numbers representing the RGB color: the red-component, the green-component and the blue-component. The number of array elements corresponds to the number of triangles in the face interior.

Example:

```
<FaceMesh id="234">
  <Mesh>
    <Id>34</Id>
  </Mesh>
  <LoopIds N="1">
    <Id>55</Id>
  </LoopIds>
  <Triangles N="8">
    19 20 21 22 23 40 41 42
  </Triangles>
  <TrianglesHidden N="3">
    0 6 7
  </TrianglesHidden>
  <TrianglesColor N="8">
    255 255 255
    255 255 255
    255 255 255
    128 0 0
    255 10 255
    100 255 255
    255 10 255
    128 0 0
  </TrianglesColor>
</FaceMesh>
```

7.3.7 Shell

Shell describes a topological shell as shown in Figure 80.

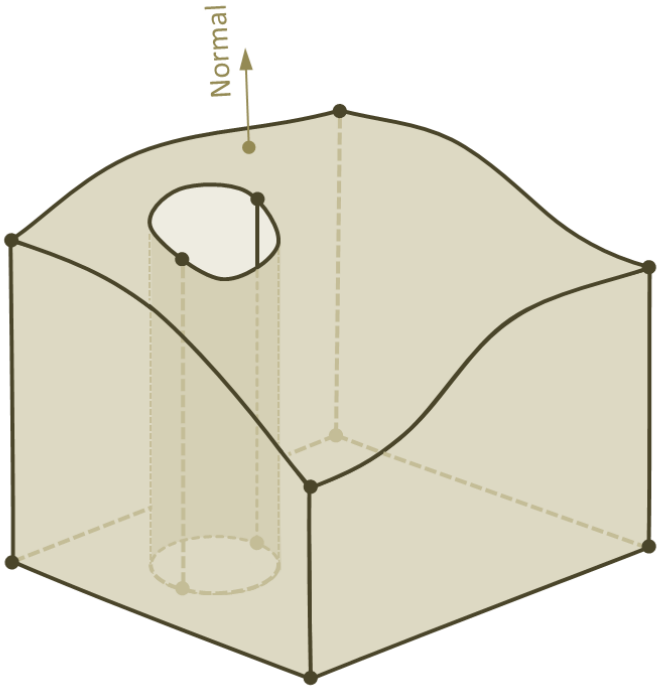


Figure 80 – Shell

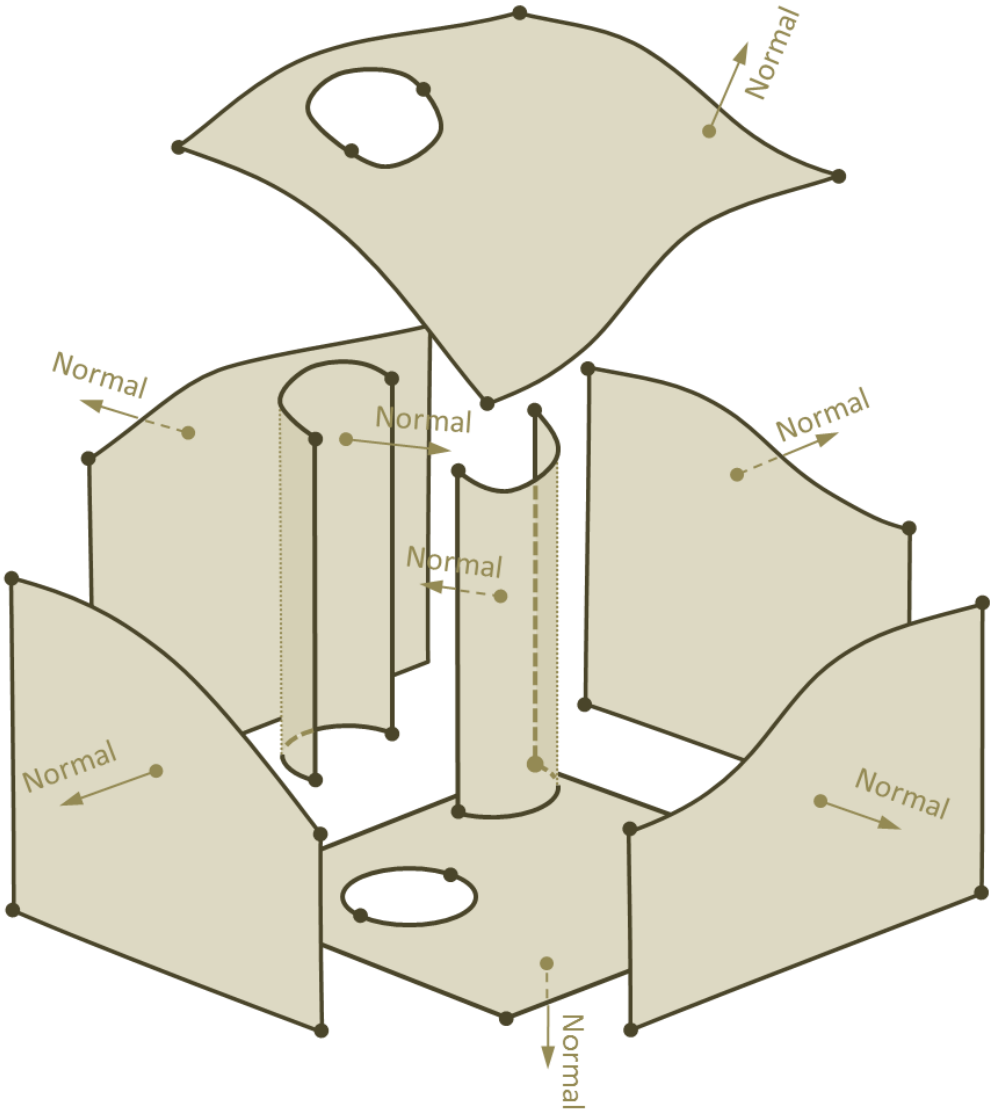


Figure 81 – Shell Faces

A topological shell is a set of connected faces. Figure 81 shows an exploded view of the faces of a shell along with their normal vectors.

Fields:

Field Name	Data Type	Description
@turned	xs:boolean	This flag shows if the shell orientation must be reversed from the orientation of the component faces. If the value is true, the shell orientation must be opposite the faces orientation. If the value is false, the two orientations must be the same.
@closed	xs:boolean	This flag shows if the shell is closed (there are no gaps or open contours).

@form	ShellFormEnumType	The shell type which can take one of the following values: 'UNKNOWN', 'OUTER' or 'INNER'.
FaceIds	ArrayReferenceType	An array of face IDs.

Example:

```
<Shell id="578" closed="1" form="OUTER">
  <FaceIds N="6">
    <Id>176</Id>
    <Id>180</Id>
    <Id>178</Id>
    <Id>174</Id>
    <Id>179</Id>
    <Id>177</Id>
  </FaceIds>
</Shell>
```

7.3.8 Body

Body describes a topological body as shown in Figure 82.

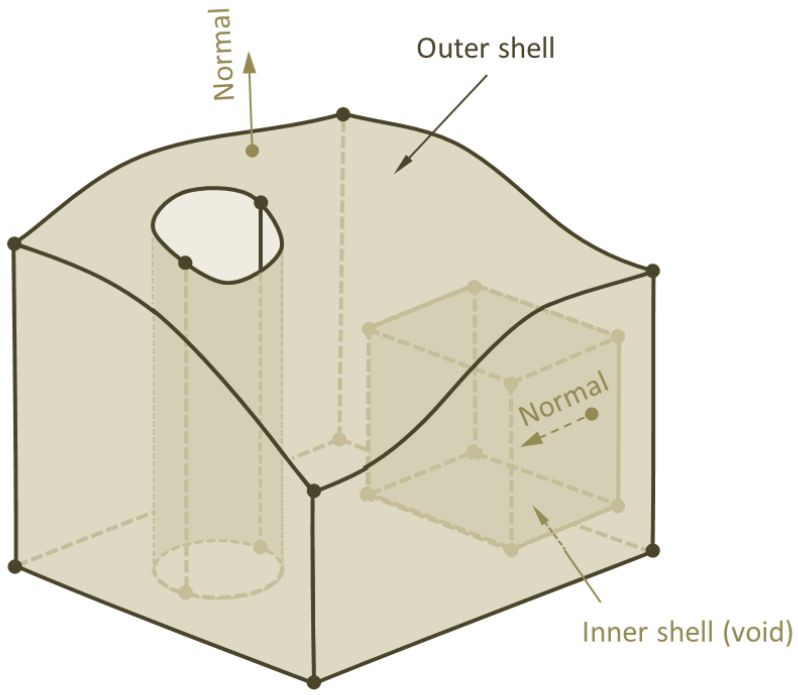


Figure 82 – Body

A body is a solid represented as a set of one outer and a number of inner shells.

Fields:

Field Name	Data Type	Description
ShellIds	ArrayReferenceType	The array of shell identifiers
FaceIds	ArrayReferenceType	The array of face identifiers.
LoopIds	ArrayReferenceType	The array of loop identifiers.
EdgeIds	ArrayReferenceType	The array of edge identifiers.
VertexIds	ArrayReferenceType	The array of vertex identifiers.

Example:

```
<Body id="127" color="0 255 0" form="Solid">
  <ShellIds N="1">
    <Id>578</Id>
  </ShellIds>
  <FaceIds N="6">
    <Id>176</Id>
    <Id>179</Id>
    <Id>180</Id>
    <Id>178</Id>
    <Id>174</Id>
    <Id>177</Id>
  </FaceIds>
  <LoopIds N="6">
    <Id>126</Id>
    <Id>161</Id>
    <Id>171</Id>
    <Id>151</Id>
    <Id>113</Id>
    <Id>139</Id>
  </LoopIds>
  <EdgeIds N="12">
    <Id>539</Id>
    <Id>555</Id>
    <Id>567</Id>
    <Id>575</Id>
    <Id>511</Id>
    <Id>495</Id>
    <Id>487</Id>
    <Id>456</Id>
    <Id>430</Id>
    <Id>405</Id>
    <Id>469</Id>
    <Id>425</Id>
  </EdgeIds>
  <VertexIds N="8">
    <Id>553</Id>
    <Id>566</Id>
    <Id>494</Id>
    <Id>486</Id>
```

```
    <Id>451</Id>
    <Id>401</Id>
    <Id>462</Id>
    <Id>421</Id>
  </VertexIds>
</Body>
```

7.3.9 Point Cloud

PointCloud describes a cloud of points as shown in Figure 83 and Figure 84.

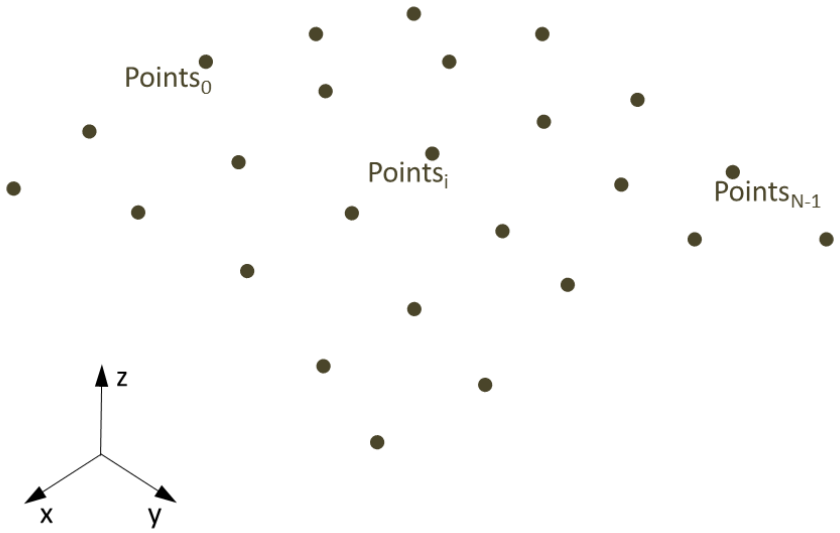


Figure 83 – Cloud of Points

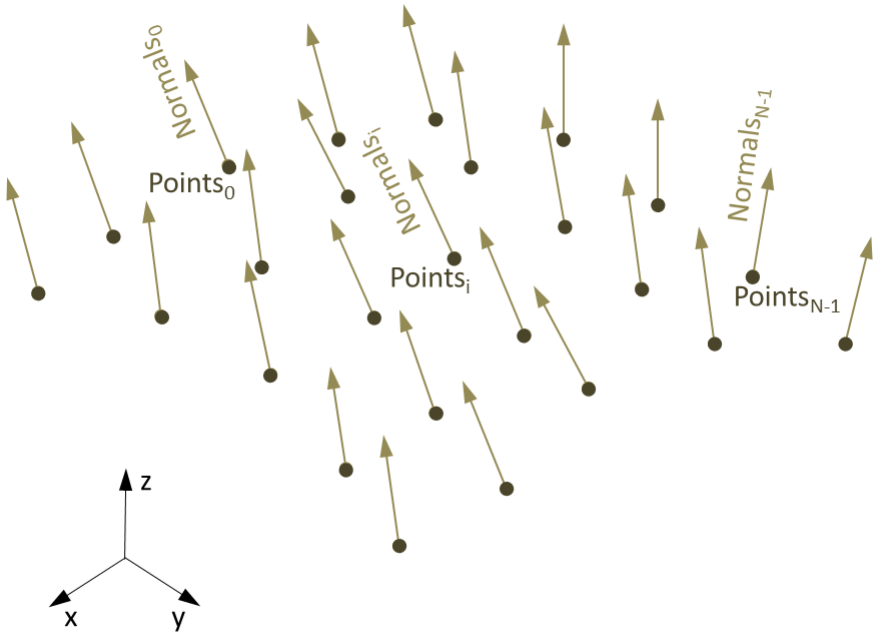


Figure 84 – Cloud of Point with Defined Normals

$N - number\ of\ points$

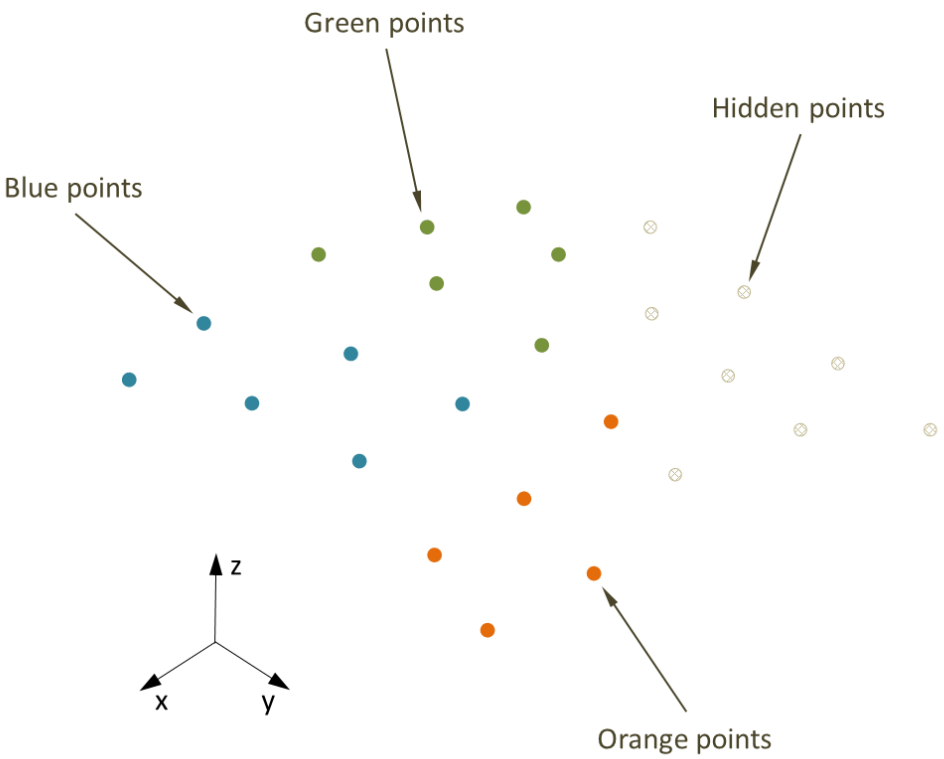


Figure 85 – Point Cloud (Point Visibility and Color)

A point cloud is a collection of 3D points with an optional set of associated normals. The points may be colored and they may be hidden. In order to simplify work with point clouds QIF considers this object as a topological exception. This means that this type of geometric data does not need any extra topological wrapper and it can participate in the model scene as it is.

Fields:

Field Name	Data Type	Description
Points or PointsBinary	ArrayPointType or ArrayBinaryType	The array of 3D points.
Normals or NormalsBinary	ArrayUnitVectorType or ArrayBinaryType	The array of normals. The number of elements in this array must equal the number of 3D points in this point cloud.
PointsVisible or PointsVisibleBinary or PointsHidden or PointsHiddenBinary	ArrayIntType or ArrayBinaryType or ArrayIntType or ArrayBinaryType	The indexes from the Points array of the points that should be visible or hidden. If this element is not used, all points are visible. If PointsVisible[Binary] is used, all other points in the point cloud are hidden. If PointsHidden is used, all other points in the point cloud are visible.
PointsColor	ArrayIntType	An array of unsigned byte values which

or PointsColorBinary	or ArrayBinaryType	defines colors of the points. Each element of this array is a triplet of unsigned byte numbers representing the RGB color: the red-component, the green-component and the blue-component. The number of array elements corresponds to the number of points.

Example:

```
<PointCloud id="3">
  <Points N="20">
    -1.16929133858268 -0.78740157480315 0.511811023622047
    -0.883047171186937 -0.687746062992126 0.638852435112278
    -0.63918051910178 -0.620078740157481 0.622630504520268
    -0.259113444152814 -0.551181102362205 0.374744823563721
    0.00984251968503937 -0.521653543307087 0.196850393700787
    -1.27238261883931 -0.212890055409741 -0.0729075532225137
    -0.980417571260382 -0.19196288735513 0.125172480368966
    -0.731792013652614 -0.177678407483015 0.203090740200685
    -0.334651069850837 -0.162556840888716 0.146355162394824
    -0.0211431904345291 -0.15456401283173 0.0527121609798776
    -1.28273549139691 0.134149897929426 -0.596383785360163
    -0.995118906818745 0.0690377148072541 -0.360411738656125
    -0.750102610630461 0.0254231338366655 -0.245109793374594
    -0.352964367108432 -0.0143654882645842 -0.24891178726116
    -0.0215077282006419 -0.0193205016039662 -0.352580927384077
    -1.37795275590551 0.551181102362205 -1.10236220472441
    -1.12086249635462 0.318970545348498 -0.995051399825022
    -0.896033829104695 0.163312919218431 -0.939413823272091
    -0.514727325750948 0.0204141149023039 -0.927384076990376
    -0.177165354330709 0 -0.954724409448819
  </Points>
  <Normals N="20">
    -0.623765689571477 0.380773422526389 0.682589162828535
    -0.310237069934693 0.562301118777707 0.766531416355309
    0.090750754593406 0.625699548958669 0.774767303758777
    0.577779206607798 0.413801444322832 0.70351940491219
    0.00191672240401954 -0.0146948717641499 0.999890187430l3
    -0.509232347350378 0.675520319522933 0.533249204709822
    -0.326094688542491 0.757232035434036 0.56591686546418
    -0.118001946731718 0.795610989745604 0.594204252394364
    0.205316690584309 0.75203536789529 0.626328876870692
    0.00588691478872499 0.776036953180493 0.630659965062479
    -0.325919814041952 0.783893402891083 0.528476496846074
    -0.0597624697937558 0.921696358846995 0.383280927378786
    0.0741285257671752 0.965747087229765 0.248671520634772
    0.0810453652229759 0.988778769243019 0.125491809573915
    0.0200710836160788 0.993714825343097 0.110127187813973
    0.398014876083999 0.696526033146993 0.597022314125991
    0.588385732990174 0.795976710575262 0.142208668636623
    0.501600088180723 0.865095712095178 0.00260008685018021
```

```
    0.16965400244185 0.984848612494367 -0.0359267299288623
    0 1 0
  </Normals>
</PointCloud>
```

7.3.10 Sewn Faces

Two faces sewn together have at least one common edge between them that corresponds to co-edges defined on each of the two faces. The bypass of the co-edges must be oriented so that they are opposite from each other in order to conform to the normals of each face. If the orientations of the underlying surfaces are not conformal – the face optional attribute "@turned" must be used to specify that the face normal is opposite to its underlying surface normal. Sewn faces are illustrated in Figure 86 and Figure 87.

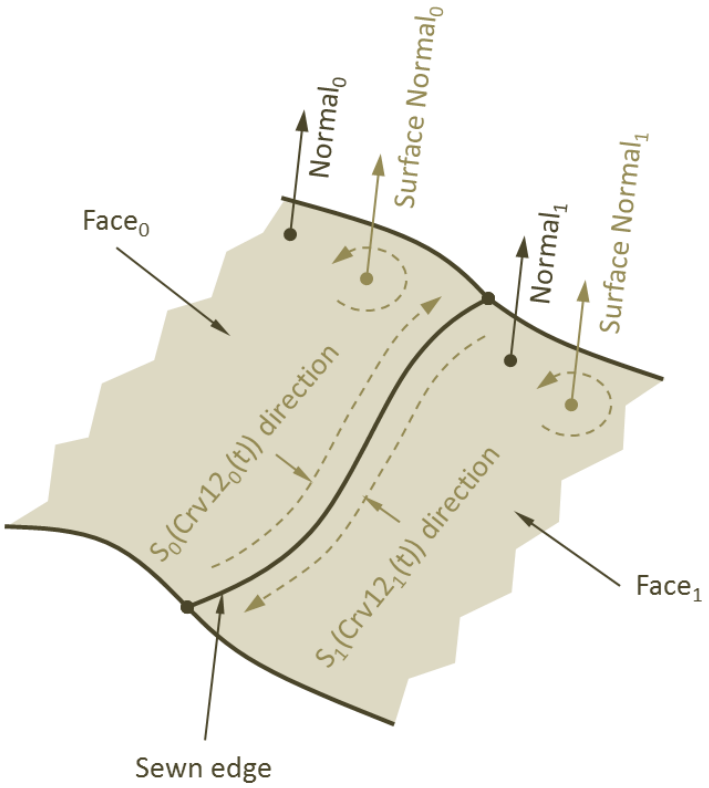


Figure 86 – Sewn Faces (normals of the underlying surfaces are conformed: $Turned_0$ = $Turned_1$ = FALSE)

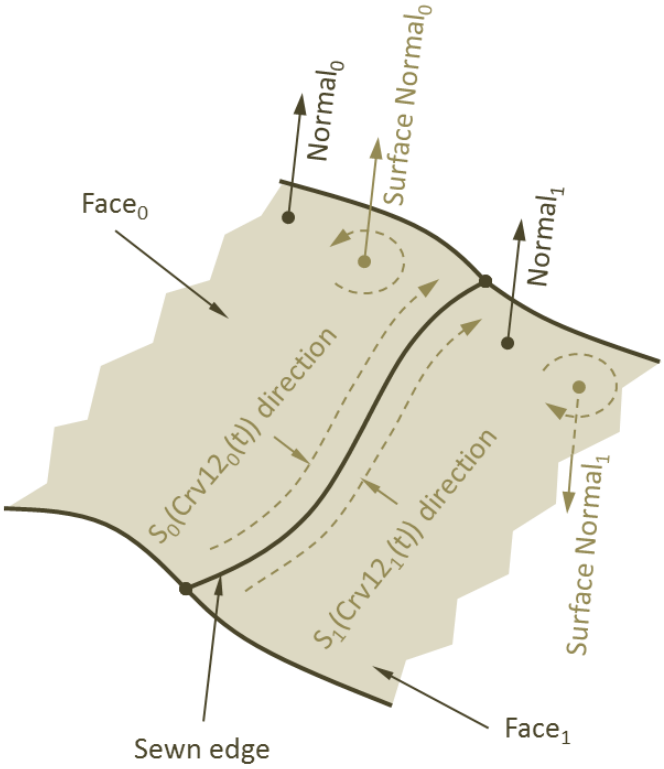


Figure 87 – Sewn Faces (normals of the underlying surfaces are not conformed: Turned$_0$ (FALSE) ≠ Turned$_1$ (TRUE))

One of implications of the face normals conformance requirement is impossibility of a Mobius strip definition.

7.3.11 Tolerant Edges and Vertices

It is highly recommended to maintain one global model tolerance for a whole body. It makes the body definition more robust for geometric algorithms and less problematic to transfer to other environments, systems and formats. There are a number of geometric kernels which allow the loosening of tolerances on some model entities while maintaining topological connectivity. This concept is referred to as tolerant modeling.

QIF can store data from the both types of modelers (exact and tolerant) by use of the optional attribute “@tolerance” which can be specified for edges and vertices.

Tolerant edges must contain the actual tolerance value which is calculated as the maximum distance between the 2D parametric co-edges of the neighbor faces.

Tolerant vertices must contain the actual tolerance value which is calculated as the maximum distance from the vertex underlying 3D point to the ends of all neighboring edges that are terminated in the neighborhood of this vertex.

Figure 88 illustrates tolerant edges and vertices.

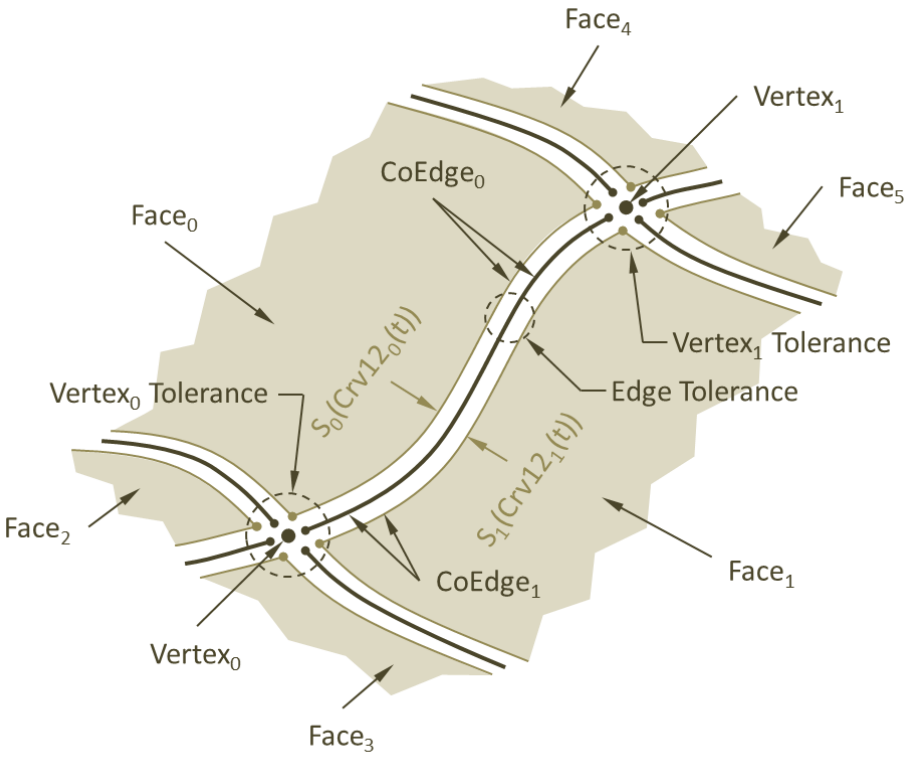


Figure 88 – Tolerant Edges and Vertices

7.4 Product structure

7.4.1 Assembly Graph: Root, Parts, Assemblies and Components

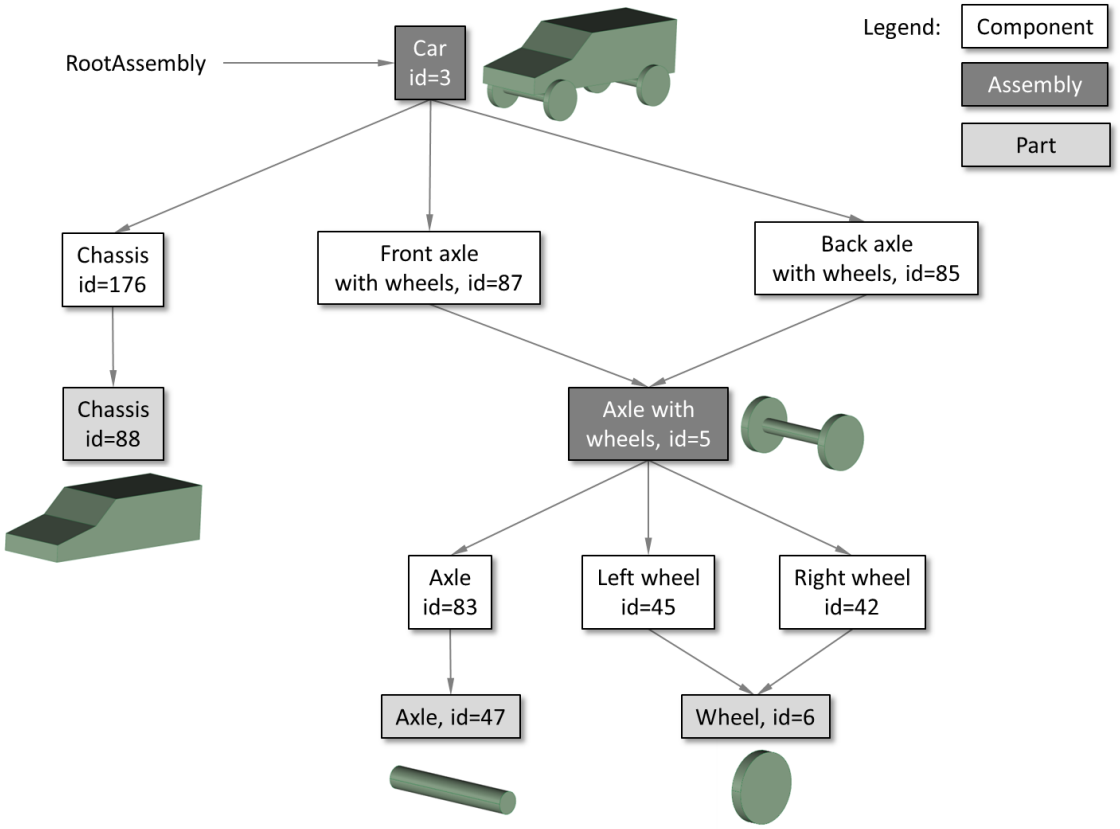


Figure 89 – Product directed acyclic graph

Figure 89 shows the product directed acyclic graph representing the relationships between parts, assemblies and components which describe the product content.

A part defines one body or a number of bodies (also known as a “multi-body part”), and can be instantiated multiple times in the CAD scene. The use of part instances simplifies the need to maintain identical model elements and reduces the total amount of scene data.

An assembly defines one or more bodies and can also include other components (instances of other parts or sub-assemblies). An assembly itself can be instantiated multiple times in the CAD scene. The use of assemblies simplifies the need to maintain identical model elements and reduces the total amount of scene data.

Assembly Fields (product graph relationships):

Field Name	Data Type	Description
ComponentIds	ArrayReferenceType	The array of component identifiers used in the product definition.

Component defines a single instance of a part or an assembly.

Fields of component (product graph relationships):

Field Name	Data Type	Description
Transform	ElementRefernceType	The identifier of the transformation matrix that maps the component into the product.
Part or Assembly	ArrayReferenceType	The identifier of a part or assembly to be instantiated.

Root of the product directed acyclic graph:

Root	Description
Product/RootPart	The starting point of the product directed acyclic graph for a Part. The graph contains only one node because the part cannot contain references to any components or assemblies.
Product/RootAssembly	The starting point of the product directed acyclic graph for an Assembly.
Product/RootComponent	The starting point of the product directed acyclic graph for a Component.

7.4.2 References in Assemblies (AsmPaths)

AsmPath defines information about assembly path, this path is used for identification of an entity within an assembly. The assembly path is a sequence of identifiers of the scene components which contain the locating entity. This sequence shows the "path" from the root to the entity parent component. The id of the parent component must be the last element in this sequence.

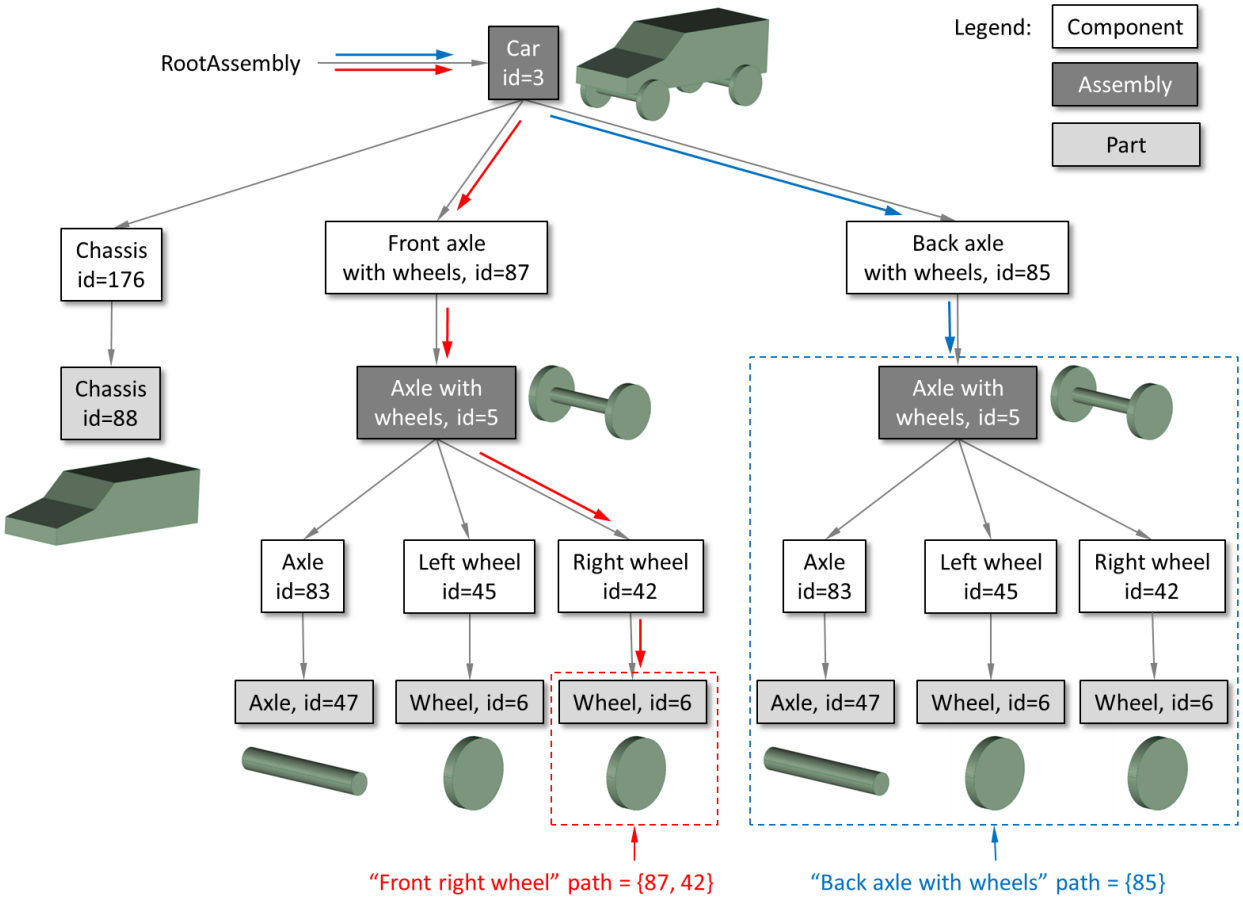


Figure 90 – Unfolded product tree

AsmPath definition:

$$AsmPath = \{ComponentIds_0, \dots, ComponentIds_{n-1}\}$$

$$n - number\ of\ elements\ in\ ComponentIds$$

$$ComponentIds_0 - this\ component\ does\ not\ have\ a\ parent\ component$$

$$ComponentIds_{i-1} - the\ nearest\ parent\ component\ for\ ComponentIds_i, \qquad i \in [1, n-1]$$

AsmPath Fields:

Field Name	Data Type	Description

@id	QIFIdType	The QIF identifier.
ComponentIds	ArrayReferenceType	The array of identifiers of the scene components which contain this entity. This array shows the "path" from the root to the parent component. The id of the parent component must be the last element of this array.

Unfolded Part instances within the Product:

Part	Part id	Instance	AsmPath
Chassis	88	Chassis	176
Axle	47	Front axle	87, 83
Axle	47	Back axle	85, 83
Wheel	6	Front left wheel	87, 45
Wheel	6	Front right wheel	87, 42
Wheel	6	Back left wheel	85, 45
Wheel	6	Back right wheel	85, 42

Unfolded Assembly instances within the Product:

Assembly	Assembly id	Instance	AsmPath
Car	3	Car	<empty>
Axle with wheels	5	Front axle with wheels	87
Axle with wheels	5	Back axle with wheels	85

In QIF all assembly paths, which are effectively used for referencing, are collected in one table (AsmPaths). Every assembly path has its own unique QIFId which is included in one or more referencing elements of QIFReferenceFullType. This provides a clear, unambiguous and compact way of referencing part entities within an assembly of any complexity.

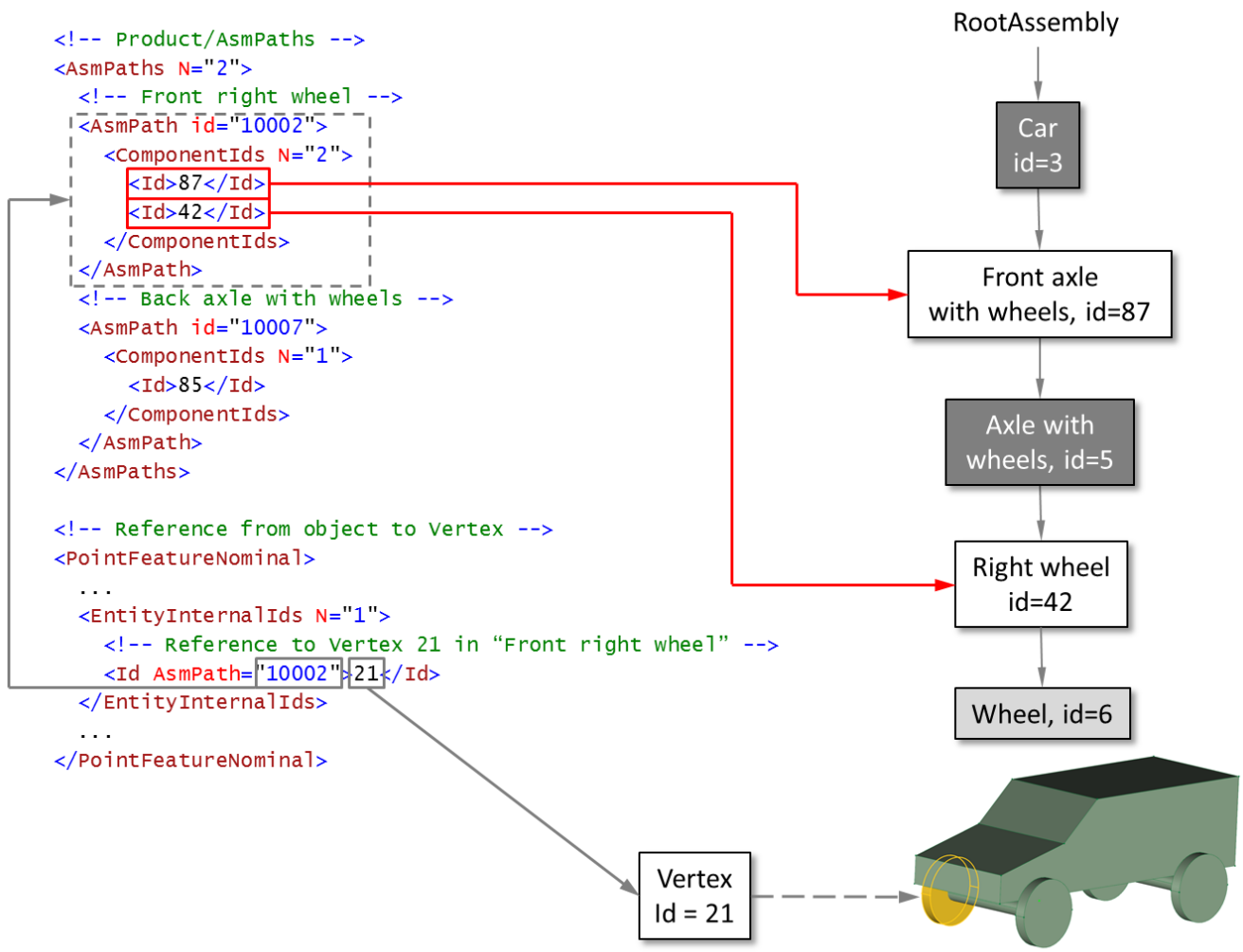


Figure 91 – Reference of a part entity within an assembly

Example:

```
<!-- Product/AsmPaths -->
<AsmPaths N="2">
  <!-- Front right wheel -->
  <AsmPath id="10002">
    <ComponentIds N="2">
      <Id>87</Id>
      <Id>42</Id>
    </ComponentIds>
  </AsmPath>
  <!-- Back axle with wheels -->
  <AsmPath id="10007">
    <ComponentIds N="1">
      <Id>85</Id>
    </ComponentIds>
  </AsmPath>
```

```
</AsmPaths>

<!-- Reference from an object to a Vertex -->
<PointFeatureNominal>
  ...
  <EntityInternalIds N="1">
    <!-- Reference to the Vertex 21 in "Front right wheel" -->
    <Id AsmPath="10002">21</Id>
  </EntityInternalIds>
  ...
</PointFeatureNominal>
```

7.4.3 Positioning of parts, sub-assemblies and bodies

Parts and subassemblies can be instantiated multiple times via components. Positioning of parts/subassemblies in 3D space is defined by transformation specified in the Transform field of component.

$$T(x) - transformation$$

$$T(x)\colon R_3 \to R_3$$

$$PA - part\ or\ subassembly$$

$$AsmPath - assembly\ path\ to\ PA$$

$$AsmPath = \{C_0, \dots, C_{n-1}\}$$

$$TC_i(x) - transformation\ associated\ with\ the\ i-th\ component,$$
$$where\ \ x - a\ 3D\ point\ of\ PA$$

$$TC_i(x) = \begin{cases} transformation\ referenced\ by\ Transform, & if\ the\ Transform\ field\ is\ filled \\ identity\ transformation, & otherwise \end{cases}$$

Positioning of PA geometry inside the parent component C_{n-1} must be calculated by applying transformation $TC_{C_{n-1}}$ to the PA entities. To calculate position and orientation of PA entities in Global Model Space, it is necessary to follow the entire instantiation chain (AsmPath) sequentially applying transformations coming from PA to the assembly root.

$$TPAmodel(x)$$
$$- total\ transformation\ of\ a\ 3D\ point\ defined\ in\ PA\ to\ Global\ Model\ Space$$

$$TPAmodel(x) = \left(TC_{C_0} \circ TC_{C_1} \circ \dots \circ TC_{C_{n-1}}\right)(x), \qquad x - the\ 3D\ point\ defined\ in\ PA$$

A body can be included in only one part or assembly (the DefinitionInternal/BodyIds field in definition of Part/Assembly). Positioning of bodies is defined by the transformation specified in the Transform field.

$$B - a\ body\ referenced\ from\ a\ part\ or\ a\ subassembly\ PA$$

$$TB_i(x) - transformation\ associated\ with\ the\ i-th\ body,\ where\ x - a\ 3D\ point\ of\ B$$

$$TB_i(x) = \begin{cases} transformation\ referenced\ by\ Transform, & Transform\ presented\ in\ the\ i-th\ body \\ identity\ transformation, & otherwise \end{cases}$$

To calculate the position and orientation of the B geometry inside PA it is necessary to apply the TB_B transformation to the B geometry.

To calculate the position and orientation of the B entities in Global Model Space, it is necessary to apply the $TPAmodel$ transformations as described above.

$$TBmodel_i(x)\ - transformation\ of\ a\ 3D\ point\ defined\ in\ the\ i_{th}\ body\ Global\ Model\ Space$$

$$TBmodel_B(x) = (TPAmodel \circ TB_B)(x), \qquad x - a\ 3D\ point\ defined\ in\ B$$

7.4.4 External Definitions of Parts and Assemblies

A part or assembly can also contain external definitions such as files, images, drawings, CAD files of native or neutral formats, physical model or prototype, etc.

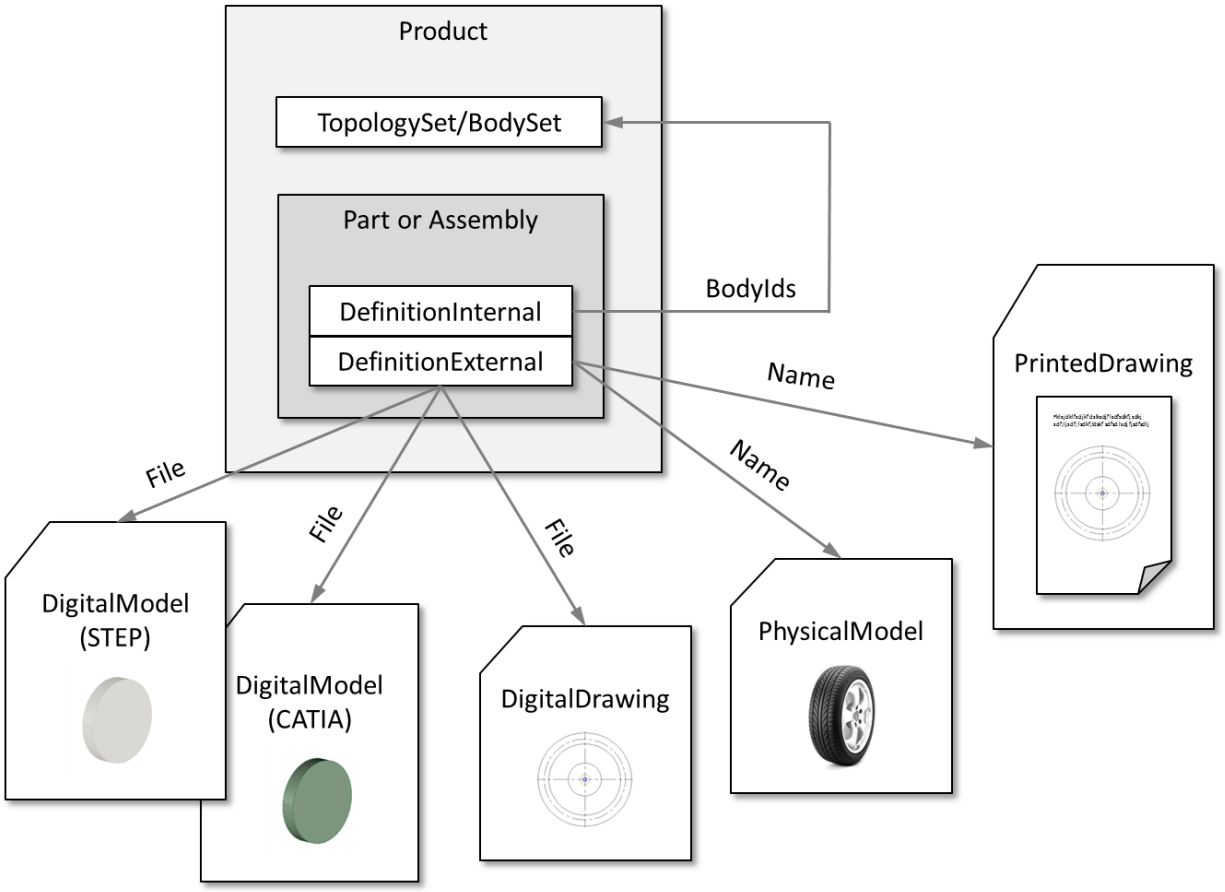


Figure 92 – Multiple representations for parts and assemblies

7.4.4.1 PrintedDrawing

PrintedDrawing defines information about a printed drawing of a product. This may be on paper, mylar, or some other physical media.

Fields:

Field Name	Data Type	Description
Name	xs:string	The name of the model.
Version	xs:string	The version of the model associated with product being inspected.
Description	xs:string	A description of the model.
Author/Name	xs:string	The name of the author.

Author/Organization	xs:string	The name of author's organization.
DrawingNumber	xs:string	The drawing number of the printed drawing associated with product being inspected.
AdditionalChanges	xs:string	The description or references to descriptions of any additional changes to the drawing beyond what is included in the Version.
Location	xs:string	A description of the physical location of the printed drawing.

Example:

```
<PrintedDrawing id="1">
  <Name>Widget Drawing</Name>
  <Version>2</Version>
  <Description>Widget Drawing rev02</Description>
  <DrawingNumber>12345</DrawingNumber>
  <Location>Engineering</Location>
</PrintedDrawing>
```

7.4.4.2 DigitalDrawing

The DigitalDrawing defines an electronic version of a drawing. It has only 2D drawing information. Any geometric dimensioning and tolerancing information is drawn with annotation symbols.

Fields:

Field Name	Data Type	Description
Name	xs:string	The name of the model
File/Name	xs:token	The fully qualified identifier (URI) of the file.
File/Version	xs:token	The version number of the file.
File/Format	DigitalModelFormatType	The file format.
Application/Name	xs:string	The name of the software application wherein the model was most recently edited.
Application/Organization	xs:string	The name of the organization that created the software application wherein the model was most recently edited.
Application/AddonName	xs:string	The name of the software add-on application wherein the model was most recently edited.
Application/AddonOrganization	xs:string	The name of the organization that created the software add-on

		application wherein the model was most recently edited.
Author/Name	xs:string	The name of the author.
Author/Organization	xs:string	The name of author's organization.
ApplicationSource/Name	xs:string	The name of the software application wherein the model was created.
ApplicationSource/Organization	xs:string	The name of the organization that created the software application wherein the model was created.
ApplicationSource/AddonName	xs:string	The name of the software add-on application wherein the model was created.
ApplicationSource/AddonOrganization	xs:string	The name of the organization that created the software add-on application wherein the model was created.
Description	xs:string	A description of the model.
Entities	EntitiesExternalType	A list of instances of the EntityExternalType associated with the model. Only those entities from the model that need to be referenced should be included in this list.

Example:

```
<DigitalDrawing id="5">
  <Name>Drawing</Name>
  <File>
    <Name>c:/models/drawing_133.catdrawing</Name>
    <Format>
      <DigitalModelFormatEnum>CATIA</DigitalModelFormatEnum>
    </Format>
  </File>
  <Description>Sample drawing</Description>
</DigitalDrawing>
```

7.4.4.3 DigitalModel

The DigitalModel defines a digital data model that represents information about an assembly or part.

Fields:

Field Name	Data Type	Description

Name	xs:string	The name of the model
File/Name	xs:token	The fully qualified identifier (URI) of the file.
File/Version	xs:token	The version number of the file.
File/Format	DigitalModelFormatType	The file format.
Application/Name	xs:string	The name of the software application wherein the model was most recently edited.
Application/Organization	xs:string	The name of the organization that created the software application wherein the model was most recently edited.
Application/AddonName	xs:string	The name of the software add-on application wherein the model was most recently edited.
Application/AddonOrganization	xs:string	The name of the organization that created the software add-on application wherein the model was most recently edited.
Author/Name	xs:string	The name of the author person.
Author/Organization	xs:string	The name of author's organization.
ApplicationSource/Name	xs:string	The name of the software application wherein the model was created.
ApplicationSource/Organization	xs:string	The name of the organization that created the software application wherein the model was created.
ApplicationSource/AddonName	xs:string	The name of the software add-on application wherein the model was created.
ApplicationSource/AddonOrganization	xs:string	The name of the organization that created the software add-on application wherein the model was created.
Description	xs:string	A description of the model.
Units	OtherUnitsType	Specifies the units used in the model.
GDT	GDTEnumType	Specifies the presence of geometric dimensioning and tolerancing information in model.
Topology	TopologyEnumType	Specifies the presence of topology information in model.
Entities	EntitiesExternalType	A list of instances of the EntityExternalType associated with

		the model. Only those entities from the model that need to be referenced should be included in this list.

Example:

```
<DigitalModel id="6">
  <Name>Model</Name>
  <File>
    <Name>c:/models/model_12.sat</Name>
    <Format>
      <DigitalModelFormatEnum>ACIS</DigitalModelFormatEnum>
    </Format>
  </File>
  <Description>Sample drawing</Description>
  <Units>
    <LinearUnit>
      <UnitName>mm</UnitName>
    </LinearUnit>
  </Units>
  <GDT>MACHINEREAD</GDT>
  <Topology>PRESENT</Topology>
  <Entities>
    <Entity id="101">
      <EntityId>1903</EntityId>
      <Name>Wheel body</Name>
    </Entity>
    <Entity id="102">
      <EntityId>4324</EntityId>
      <Name>Chassis body</Name>
    </Entity>
  </Entities>
</DigitalModel>
```

7.4.4.4 PhysicalModel

The PhysicalModel is an actual, hands-on, physical, model or prototype of a part that is used to communicate features, datum reference frames, characteristics, or other information. Examples of physical models include actual part instances, part instances made from other materials, and 3D printing and stereolithography models.

Fields:

Field Name	Data Type	Description
Name	xs:string	The name of the model.
Version	xs:string	The version of the model associated with product being inspected.

Description	xs:string	A description of the model.
Author/Name	xs:string	The name of the author.
Author/Organization	xs:string	The name of author's organization.
Location	xs:string	A description of the physical location of the physical model
ModelNumber	xs:string	The model number of the physical model being inspected.

Example:

```
<PhysicalModel id="7">
  <Name>Wheel</Name>
  <Version>2</Version>
  <Location>Warehouse 2, position 24A/10</Location>
  <ModelNumber>23876-2372-33</ModelNumber>
</PhysicalModel>
```

7.4.5 Layers

A layer is a “slice” of the model containing a set of model entities independent of the referencing model hierarchy. The user can specify arbitrary number of layers for one model. A model entity can be included in a single or multiple layers.

Fields:

Field Name	Data Type	Description
@hidden	xs:boolean	Defines the visibility property of model entities in the graphical window.
@color	ColorType	Defines the RGB color property of model entities.
@applyColor	xs:boolean	Shows if the layer color supersedes colors of the model entities associated with this layer.
@index	xs:unsignedInt	The layer index.
ElementIds	ArrayReferenceFullType	An array of entity identifiers presents in this this layer.

Example:

```
<Layer id="33" applyColor="0" hidden="0" label="Layer 1" index="12">
  <ElementIds N="4">
    <Id>23</Id>
    <Id>24</Id>
    <Id>1324</Id>
    <Id>1323</Id>
  </ElementIds>
</Layer>
```

7.4.6 Part Notes

PartNote describes a part (standard) note – a treelike note which is used for annotating of a whole model or a separate model entity. In QIF all part notes are collected in the PartNoteSet. Normally part notes are visible in the model tree, and in some systems are also visualized in the graphical window so the part notes are implemented in QIF as drawable entities with the corresponding set of attributes.

Fields:

Field Name	Data Type	Description
Text	xs:string	A text of the current node of the part note.
PartNoteIds	ArrayReferenceType	An array of identifiers of nested/children part notes.

Example:

```
<PartNote id="1427" label="NIST Test Case 1 CATIA V5R21 RB">
  <PartNoteIds N="7">
    <Id>1428</Id>
    <Id>1429</Id>
    <Id>1430</Id>
    <Id>1431</Id>
    <Id>1432</Id>
    <Id>1433</Id>
    <Id>1434</Id>
  </PartNoteIds>
</PartNote>
<PartNote id="1428" label="Part Number">
  <Text>NIST Test Case 1 CATIA V5R21 RB</Text>
</PartNote>
<PartNote id="1429" label="Revision">
  <Text>B</Text>
</PartNote>
<PartNote id="1430" label="Nomenclature">
  <Text>NIST Complex Test Case 1 V5R21</Text>
</PartNote>
<PartNote id="1431" label="Source">
  <Text>Unknown</Text>
</PartNote>
<PartNote id="1432" label="Modeled By">
  <Text>Rich Eckenrode, RECON Services.com</Text>
</PartNote>
<PartNote id="1433" label="CAGE Code">
  <Text>64JW1</Text>
</PartNote>
<PartNote id="1434" label="Company">
  <Text>RECON Services Incorporated</Text>
</PartNote>
```


7.4.7 Notes

Note describes an annotation note, which can be visualized in the graphical window as a 3D annotation or as a flat-to-screen note.

Fields:

Field Name	Data Type	Description
@id	QIFIdType	The unique model entity identifier.
@label	xs:string	The model entity "nameplate". Normally it can be seen at the entity item in the model tree.
@color	ColorType	The RGB color property of a model entity.
@transparency	xs:double	The transparency property of a model entity.
@hidden	xs:boolean	The visibility property of a model entity in the graphical window.
@size	xs:double	A recommended size for visualization of an infinite drawable element.
@form	NoteFormEnumType	This attribute specifies the note form: '3D' - note defined on the 3D annotation plane; 'SCREEN' - note defined as flat to screen;
Attributes	AttributesType	User-defined attributes (typified, binary array, or XML structured).
EntityInternalIds	ArrayReferenceFullType	An array of identifiers of CAD entities associated with this note.
EntityExternalIds	ArrayReferenceFullType	An array of identifiers of instances of EntityExternalType associated with this note.
Text	xs:string	A text of the note.

Example:

```
<Note id="1455" label="Note_10" form="3D">
  <EntityInternalIds N="1">
    <Id>2195</Id>
  </EntityInternalIds>
  <Text>B</Text>
</Note>
```

7.4.8 Flag notes

NoteFlag describes a flag note, which is visualized by its flag note symbol and can be attached to model entities.The shape of a flag note (flag style) is defined in an associated dispay info object.

Fields:

Field Name	Data Type	Description
@id	QIFIdType	The unique model entity identifier.
@label	xs:string	The model entity "nameplate". Normally it can be seen at the entity item in the model tree.
@color	ColorType	The RGB color property of a model entity.
@transparency	xs:double	The transparency property of a model entity.
@hidden	xs:boolean	The visibility property of a model entity in the graphical window.
@size	xs:double	A recommended size for visualization of an infinite drawable element.
@form	NoteFormEnumType	This attribute specifies the note form: '3D' - note defined on the 3D annotation plane; 'SCREEN' - note defined as flat to screen;
Attributes	AttributesType	User-defined attributes (typified, binary array, or XML structured).
EntityInternalIds	ArrayReferenceFullType	An array of identifiers of CAD entities associated with this flag note.
EntityExternalIds	ArrayReferenceFullType	An array of identifiers of instances of EntityExternalType associated with this flag note.
Text	xs:string	A text of the flag note (the flag note symbol).
TextHidden	xs:string	A hidden text of the flag note.
URI	xs:anyURI	A Uniform Resource Identifier for the information, which may be, for example, a file or a web site.

Example:

```
<NoteFlag id="731" label="Flag Note_1">
  <EntityInternalIds N="1">
    <Id>1096</Id>
  </EntityInternalIds>
  <Text>1</Text>
</NoteFlag>
```


7.5 Transformations

In order to support all variety of systems and formats QIF allows connecting of transformations to all topological and 3D geometric elements. But it is recommended to limit use of transformations to the high level objects such as: sub-assemblies, parts and bodies. It will help to eliminate any potential interoperability issues.

Transformation defines positioning of a 3D point specifying rotation and translation.

$$T(p) - transformation$$

$$T(p)\colon R_3 \rightarrow R_3$$

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}, \qquad Origin = \begin{bmatrix} Origin_x \\ Origin_y \\ Origin_z \end{bmatrix}, \qquad R = \begin{bmatrix} XDirection_x & XDirection_y & XDirection_z \\ YDirection_x & YDirection_y & YDirection_z \\ ZDirection_x & ZDirection_y & ZDirection_z \end{bmatrix}$$

$$T(p) = pR + Origin$$

Fields:

Field Name	Data Type	Description
Rotation/XDirection	UnitVectorSimpleType	An orthogonal basis of the Cartesian coordinate system. Direction X.
Rotation/YDirection	UnitVectorSimpleType	An orthogonal basis of the Cartesian coordinate system. Direction Y.
Rotation/ZDirection	UnitVectorSimpleType	An orthogonal basis of the Cartesian coordinate system. Direction Z.
Origin	PointSimpleType	An origin of the coordinate system.

Example:

```
<Transform id="348">
  <Rotation>
    <XDirection>0.9405 -0.3396 0.0</XDirection>
    <YDirection>0.3318 0.9191 -0.2123</YDirection>
    <ZDirection>0.0721 0.1996 0.9772</ZDirection>
  </Rotation>
  <Origin>0.0798 -0.2104 -0.0984</Origin>
</Transform>
```

7.6 Auxiliary data

7.6.1 Point

PointAuxiliary describes an auxiliary point, the point which does not directly participate in forming a body, but in a definition of supplemental model entities, e.g. a center of a sphere.

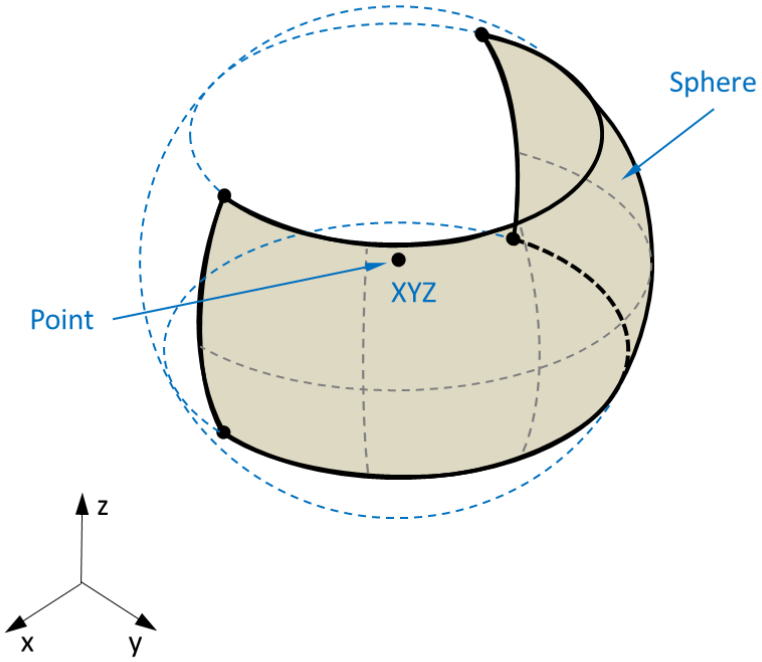


Figure 93 – Point

Fields:

Field Name	Data Type	Description
XYZ	PointType	The Cartesian three-dimensional coordinates of the 3D point.

Example:

```
<PointAuxiliary id="112">
  <XYZ>1.32 9.23 4.22</XYZ>
</PointAuxiliary>
```

7.6.2 Line

LineAuxiliary describes an auxiliary line, the line which does not directly participate in forming a body, but in a definition of supplemental model entities, e.g. an axis of a cylinder.

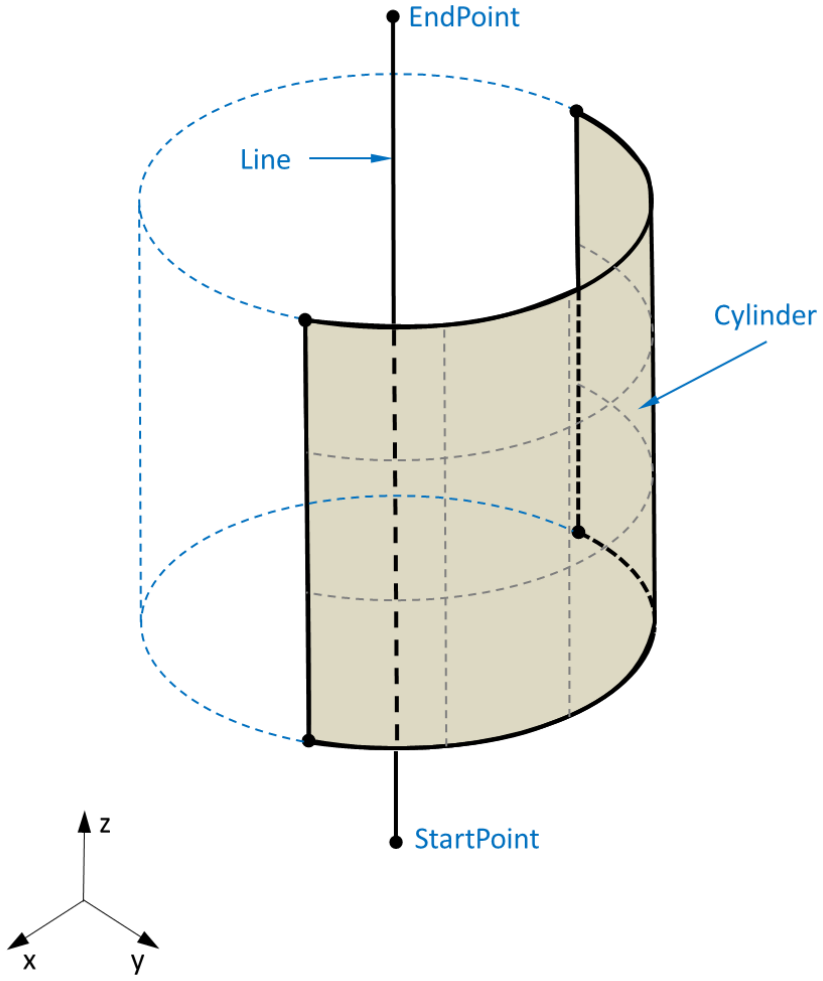


Figure 94 – Line

Fields:

Field Name	Data Type	Description
StartPoint	PointSimpleType	The beginning point of the line segment.
EndPoint	PointSimpleType	The ending point of the line segment.

Example:

```
<LineAuxiliary id="21">
  <StartPoint>1.32 9.23 4.22</StartPoint>
  <EndPoint>3.45 0.33 5.40</EndPoint>
</LineAuxiliary>
```

7.6.3 Reference Plane

PlaneReference describes an auxiliary plane, the plane which does not directly participate in forming of a body, but in a definition of supplemental model entities, e.g. a central plane of a feature.

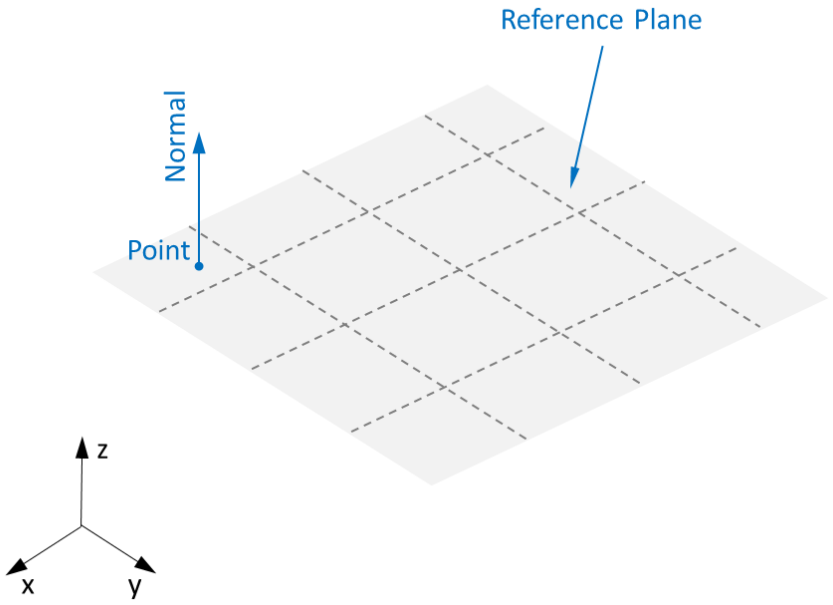


Figure 95 – Reference Plane

Fields:

Field Name	Data Type	Description
Plane/Point	PointType	The plane origin.
Plane/Normal	UnitVectorType	The unit normal vector of the plane.

Example:

```
<PlaneReference id="23">
  <Plane>
    <Point>10.0 12.0 23.1</Point>
    <Normal>1.0 0.0 0.0</Normal>
  </Plane>
</PlaneReference>
```

7.6.4 Clipping Plane

The PlaneClipping defines a model clipping plane. The objects or portions of objects in the scene on the side of the plane in the direction of the plane normal appear to be removed when

the scene is drawn. This enables, for example, producing cutaway views of parts and assemblies.

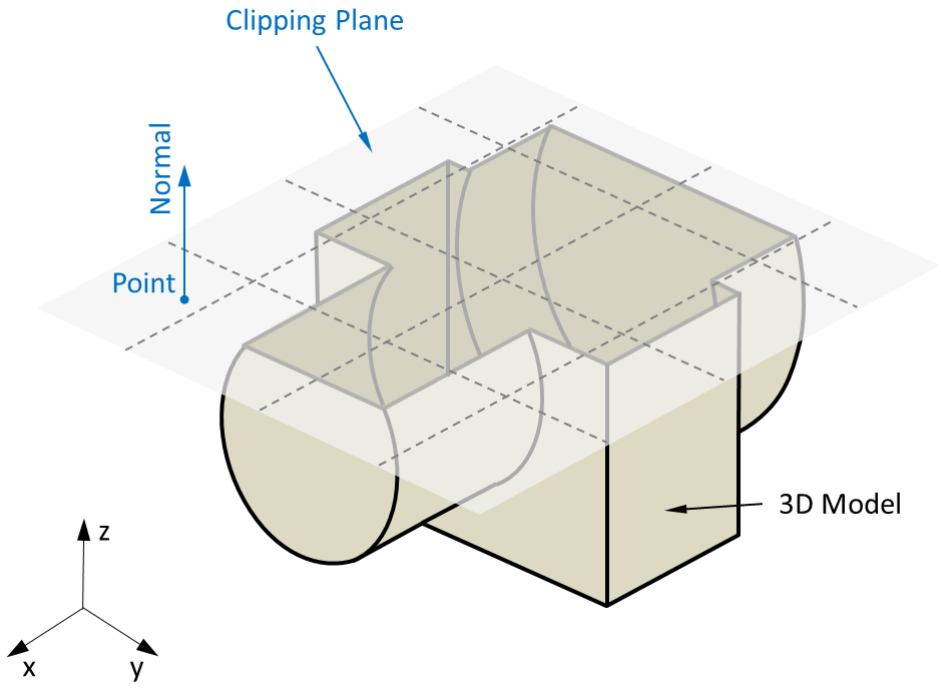


Figure 96 – Clipping Plane

Fields:

Field Name	Data Type	Description
@index	xs:integer	The clipping plane index.
@enable	xs:boolean	The clipping plane state - enabled or disabled.
Plane/Point	PointType	A point on the plane.
Plane/Normal	UnitVectorType	The unit normal vector of the plane.

Example:

```
<PlaneClipping id="23" index="3" enable="1">
  <Plane>
    <Point>10.0 12.0 23.1</Point>
    <Normal>1.0 0.0 0.0</Normal>
  </Plane>
</PlaneClipping>
```

7.6.5 Coordinate System

CADCoordinateSystem describes the Cartesian 3D coordinate system in the model space. The CADCoordinateSystem is defined as a model entity.

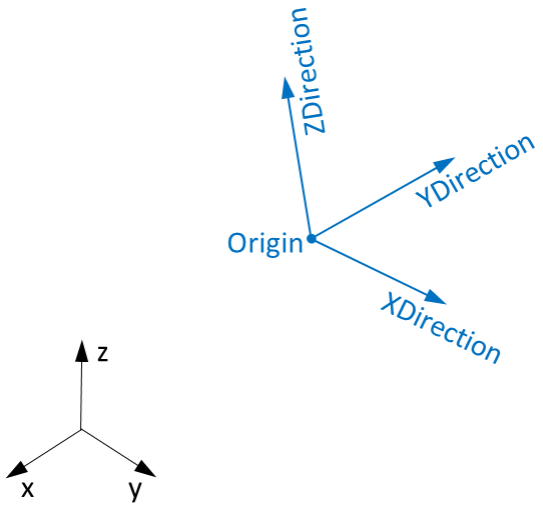


Figure 97 – Coordinate System

$$ZDirection = XDirection \times YDirection$$

$$(XDirection \cdot YDirection) = 0$$

$$(XDirection \cdot ZDirection) = 0$$

$$(YDirection \cdot ZDirection) = 0$$

Fields:

Field Name	Data Type	Description
CoordinateSystemCore/Rotation/XDirection	UnitVectorSimpleType	An orthogonal basis of the Cartesian coordinate system. Direction X.
CoordinateSystemCore/Rotation/YDirection	UnitVectorSimpleType	An orthogonal basis of the Cartesian coordinate system. Direction Y.
CoordinateSystemCore/Rotation/ZDirection	UnitVectorSimpleType	An orthogonal basis of the Cartesian coordinate system. Direction Z.
CoordinateSystemCore/Origin	PointSimpleType	An origin of the coordinate system.

Example:

```
<CoordinateSystem id="33" label="CS1">
  <CoordinateSystemCore>
    <Rotation>
      <XDirection>0.9405 -0.3396 0.0</XDirection>
      <YDirection>0.3318 0.9191 -0.2123</YDirection>
      <ZDirection>0.0721 0.1996 0.9772</ZDirection>
    </Rotation>
    <Origin>0.0798 -0.2104 -0.0984</Origin>
  </CoordinateSystemCore>
</CoordinateSystem>
```

7.7 Visualization data

The VisualizationSet represents a container for storing all visualization data used for displaying PMI entities in a 3D scene.

7.7.1 Fonts

All fonts used for visualization of 3D annotations are collected in the FontSet which is an element of the VisualizationSet.

Font fields:

Field Name	Data Type	Description
@id	xs:unsignedInt	An index identifying a font within the FontSet table.
@bold	xs:boolean	This attribute shows if the text must be bold.
@italic	xs:boolean	This attribute shows if the text must be italicized.
@underline	xs:boolean	This attribute shows if the text must be underlined.
Attributes	AttributesType	User defined attributes (typified, binary array, or XML structured).
Name	xs:string	The font name.
Size	NaturalType	The font size defined in points (typography, 1 pt = 1/72 inch).

Example:

```
<Font id="1" italic="1">
  <Name>Microsoft Sans Serif</Name>
  <Size>8</Size>
</Font>
```

7.7.2 Special symbols

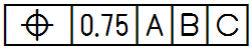
In order to store special symbols in text strings (PMIDisplay/Texts/Text) the following decorated symbol names can be used:

Decorated Symbol Name	Image	Description
{STRAIGHTNESS}		Straightness
{FLATNESS}		Flatness
{CIRCULARITY}		Circularity/Roundness
{CYLINDRICITY}		Cylindricity
{PROFILE_LINE}		Profile of a Line
{PROFILE_SURFACE}		Profile of a Surface
{ANGULARITY}		Angularity

{PERPENDICULARITY}		Perpendicularity
{PARALLELISM}		Parallelism
{POSITION}		Position
{CONCENTRICITY}		Concentricity
{SYMMETRY}		Symmetry
{RUNOUT_CIRCULAR}		Circular Runout
{RUNOUT_TOTAL}		Total Runout
{DIAMETER}	Ø	Diameter
{DIAMETER_SPHERICAL}	SØ	Spherical Diameter
{RADIUS}	R	Radius
{RADIUS_SPHERICAL}	SR	Spherical Radius
{RADIUS_CONTROLLED}	CR	Controlled Radius
{BETWEEN}		Between
{CONICAL_TAPER}		Conical Taper
{SLOPE}		Slope
{COUNTERBORE}		Counterbore
{SPOTFACE}		Spotface
{COUNTERSINK}		Countersink
{DEPTH}		Depth
{SQUARE}		Square
{PLUS_MINUS}	±	Plus Minus
{DEGREE}	°	Degree
{ST}		Statistical Tolerance
{CF}		Continuous Feature
{M}		At Maximum Material Condition
{L}		At Least Material Condition
{P}		Projected Tolerance Zone
{T}		Tangent Plane
{F}		Free State
{U}		Unequally Disposed Profile
{I}		Independency

{TRANSLATION}	▷	Translation
{BRACKET_CURLED_LEFT}	{	Left Curled Bracket
{BRACKET_CURLED_RIGHT}	}	Right Curled Bracket
{BR}		Line Break

Example:




```
<Texts lineHeight="4.5" N="5" fontIndex="1">
  <Text>
    <Data>{POSITION}</Data>
    <XY>0 0</XY>
  </Text>
  <Text>
    <Data>0.75</Data>
    <XY>1.64658 0</XY>
  </Text>
  <Text>
    <Data>A</Data>
    <XY>3.80537 0</XY>
  </Text>
  <Text>
    <Data>B</Data>
    <XY>4.79375 0</XY>
  </Text>
  <Text>
    <Data>C</Data>
    <XY>5.85390 0</XY>
  </Text>
</Texts>
```

7.7.3 PMI display information

The PMIDisplay defines a block of display data used for visualization of a 3D annotation. All PMI display information entities are collected in one set PMIDisplaySet, which is an element of the VisualizationSet.

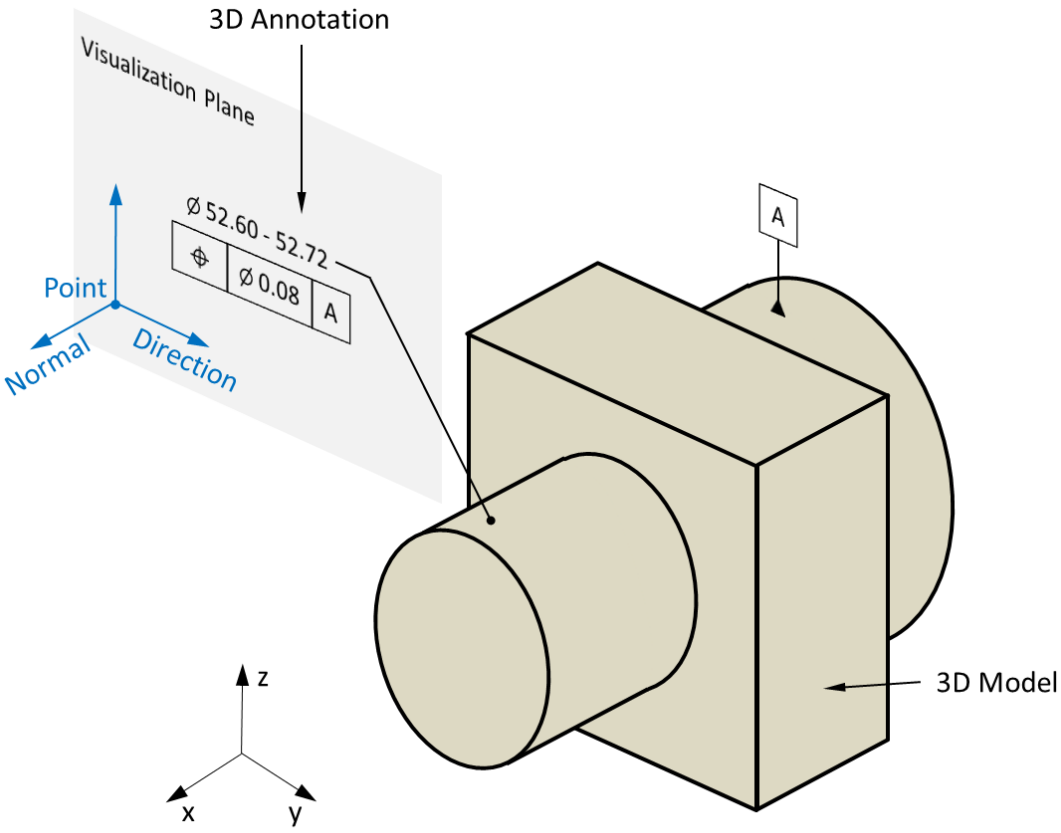


Figure 98 – Display information

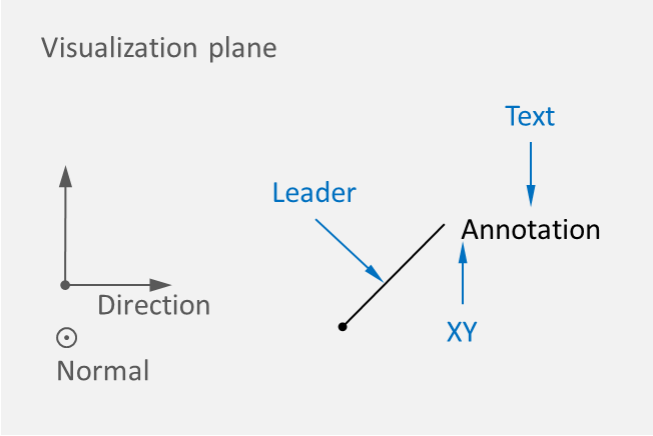


Figure 99 – Text

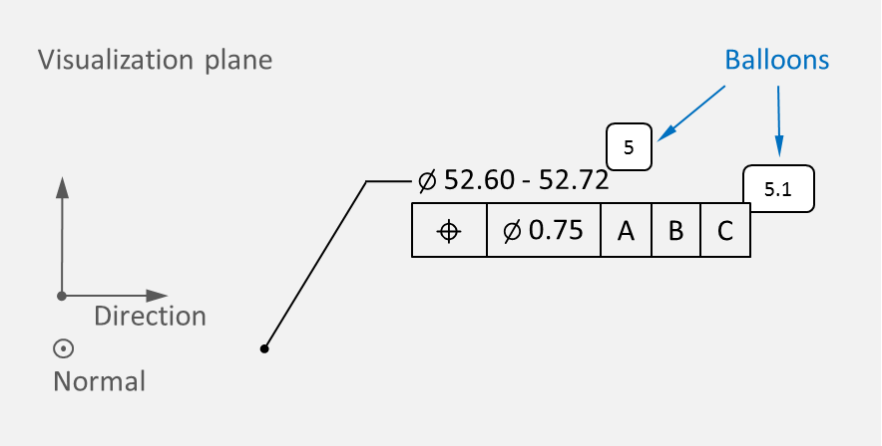


Figure 100 – Balloons

Fields:

Field Name	Data Type	Description
Attributes	AttributesType	User defined attributes (typified, binary array, or XML structured).
Color	ColorType	The RGB color type is a triplet of integer numbers: red-component, the green-component and the blue-component.
Plane	PlaneXType	A visualization plane with a predefined x-direction, which must be used as the text direction.
Plane/Point	PointType	A point on the visualization plane.
Plane/Normal	UnitVectorType	The unit normal vector of the visualization plane.
Plane/Direction	UnitVectorType	The direction of the positive X axis on visualization plane.
Texts	TextsType	A set of 3D annotation text lines.
Texts/@fontId	xs:unsignedInt	An index identifying a font within the FontSet table.
Texts/@lineHeight	xs:double	A height of the text line defined in the model units.
Texts/Text	TextType	A string.
Leader or LeaderExtend or LeaderCircular or LeaderDoubleHead or	LeaderType or LeaderExtendType or LeaderCircularType or LeaderDoubleHeadType or	Information about a leader type.

LeaderDoubleHeadCircular or LeaderDoubleHeadExtend	LeaderDoubleHeadCircularType or LeaderDoubleHeadExtendType	
WitnessLines	WitnessLinesType	Witness lines.
Frames	FramesType	The visualization frames.
Balloon	xs:unsignedInt	The visualization balloon. Every visualization balloon has a unique number.
Balloon/@sub	NaturalType	The visualization balloon. Every visualization balloon has a unique number.
Reference	ElementReferenceFullType	Reference to an annotation entity.

Example:

```
<PMIDisplay>
  <Plane>
    <Point>494.734069824219 136.153625488281 52.4019432067871</Point>
    <Normal>0 -1 0</Normal>
    <Direction>1 0 0</Direction>
  </Plane>
  <Texts lineHeight="4.5" N="3" fontId="1">
    <Text>
      <Data>{PERPENDICULARITY}</Data>
      <XY>0 0</XY>
    </Text>
    <Text>
      <Data>1.5</Data>
      <XY>1.64658203125 0</XY>
    </Text>
    <Text>
      <Data>A</Data>
      <XY>3.37275390625 0</XY>
    </Text>
  </Texts>
  <LeaderExtend>
    <StartPoint>-94.734069824219 -102.512253206787</StartPoint>
    <EndPoint>-10.8 2.7</EndPoint>
    <HeadForm>DOT_FILLED</HeadForm>
    <HeadHeight>3.6</HeadHeight>
    <PointExtension>-1.8 2.7</PointExtension>
  </LeaderExtend>
  <Frames N="3">
    <FrameRectangular>
      <XY>-0.2 -0.2</XY>
```

```
      <Width>1.64658203125</Width>
      <Height>1</Height>
    </FrameRectangular>
    <FrameRectangular>
      <XY>1.44658203125 -0.2</XY>
      <Width>1.726171875</Width>
      <Height>1</Height>
    </FrameRectangular>
    <FrameRectangular>
      <XY>3.17275390625 -0.2</XY>
      <Width>0.98837890625</Width>
      <Height>1</Height>
    </FrameRectangular>
  </Frames>
  <Reference>
    <Id>2320</Id>
  </Reference>
</PMIDisplay>
```

7.7.3.1 Leader

Leader describes a leader of a 3D annotation.

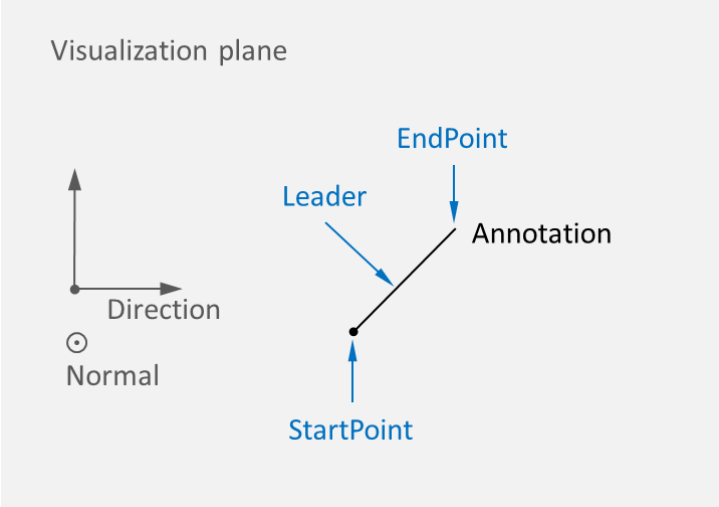


Figure 101 – Leader

Fields:

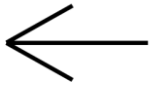
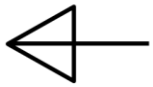
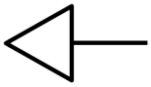
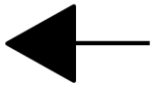
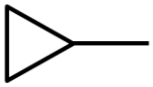
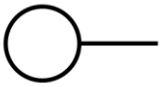
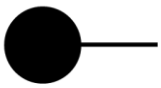
Field Name	Data Type	Description
StartPoint	Point2dSimpleType	The beginning point of the 2D line segment.
EndPoint	Point2dSimpleType	The ending point of the 2D line segment.
HeadForm	LeaderTypeEnumType	A form of the leader head.
HeadHeight	xs:double	A size of the leader head.

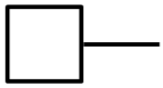
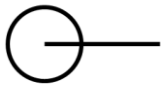
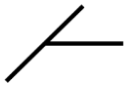
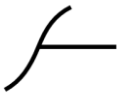
Example:

```
<Leader>
  <StartPoint>2.647 -36.261</StartPoint>
  <EndPoint>2.647 -1.800</EndPoint>
  <HeadForm>TRIANGLE_FILLED</HeadForm>
  <HeadHeight>3.600</HeadHeight>
</Leader>
```

7.7.3.2 Leader head types

LeaderHeadFormEnumType describes forms of the leader head.

Leader type	Image
NONE	
ARROW_OPEN	
ARROW_UNFILLED	
ARROW_BLANKED	
ARROW_FILLED	
TRIANGLE_BLANKED	
TRIANGLE _FILLED	
DOT_BLANKED	
DOT_FILLED	

BOX_BLANKED	
BOX_FILLED	
DIMENSION_ORIGIN	
SYMBOL_SLASH	
SYMBOL_INTEGRAL	
SYMBOL_CROSS	

7.7.3.3 Double head leader

LeaderDoubleHead describes a double head leader.

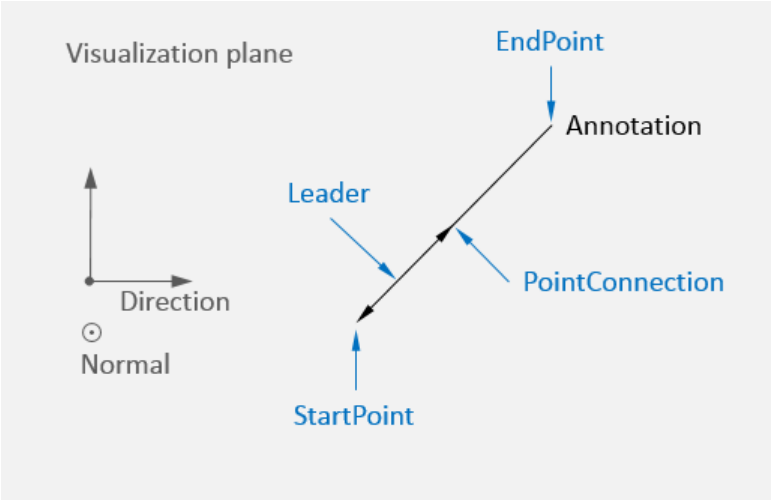


Figure 102 – Double head leader

Fields:

Field Name	Data Type	Description
StartPoint	Point2dSimpleType	The beginning point of the 2D line segment.
EndPoint	Point2dSimpleType	The ending point of the 2D line segment.
HeadForm	LeaderTypeEnumType	The first head form.
HeadHeight	xs:double	A size of the leader head.
HeadForm2	LeaderTypeEnumType	The second head form.
PointConnection	Point2dSimpleType	This field specifies 2D coordinates of the connection point.

Example:

```
<LeaderDoubleHead>
  <StartPoint>-73.225 2.066</StartPoint>
  <EndPoint>0 2.066</EndPoint>
  <HeadForm>ARROW_FILLED</HeadForm>
  <HeadHeight>4.5</HeadHeight>
  <HeadForm2>ARROW_FILLED</HeadForm2>
  <PointConnection>-38.225 2.066</PointConnection>
</LeaderDoubleHead>
```

7.7.3.4 Extended leader

LeaderExtend describes an extended leader, which has a break point and consists of two segments.

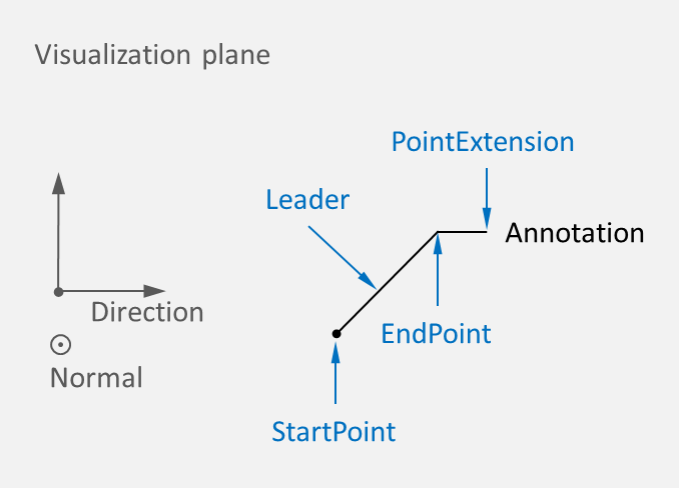


Figure 103 – Extended leader

Fields:

Field Name	Data Type	Description

StartPoint	Point2dSimpleType	The beginning point of the 2D line segment.
EndPoint	Point2dSimpleType	The ending point of the 2D line segment.
HeadForm	LeaderTypeEnumType	A form of the leader head.
HeadHeight	xs:double	A size of the leader head.
PointExtension	Point2dSimpleType	This field specifies 2D coordinates of the extension point.

Example:

```
<LeaderExtend>
  <StartPoint>-94.73 -102.51</StartPoint>
  <EndPoint>-10.80 2.70</EndPoint>
  <HeadForm>DOT_FILLED</HeadForm>
  <HeadHeight>3.6</HeadHeight>
  <PointExtension>-1.80 2.70</PointExtension>
</LeaderExtend>
```

7.7.3.5 Double head extended leader

LeaderDoubleHeadExtend describes a double head extended leader; which has a break point and consists of two segments.

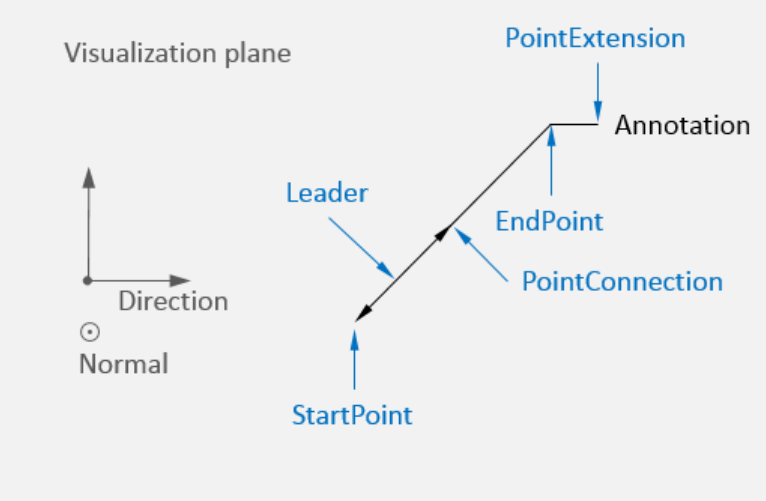


Figure 104 – Double head extended leader

Fields:

Field Name	Data Type	Description
StartPoint	Point2dSimpleType	The beginning point of the 2D line segment.
EndPoint	Point2dSimpleType	The ending point of the 2D line segment.
HeadForm	LeaderTypeEnumType	The first head form.

HeadHeight	xs:double	A size of the leader head.
HeadForm2	LeaderTypeEnumType	The second head form.
PointConnection	Point2dSimpleType	This field specifies 2D coordinates of the connection point.
PointExtension	Point2dSimpleType	This field specifies 2D coordinates of the extension point.

Example:

```
<LeaderDoubleHeadExtend>
  <StartPoint>-73.225 2.066</StartPoint>
  <EndPoint>0 2.066</EndPoint>
  <HeadForm>ARROW_FILLED</HeadForm>
  <HeadHeight>4.5</HeadHeight>
  <HeadForm2>ARROW_FILLED</HeadForm2>
  <PointConnection>-38.225 2.066</PointConnection>
  <PointExtension>15.000 10.000</PointExtension>
</LeaderDoubleHeadExtend>
```

7.7.3.6 Circular leader

LeaderCircular describes a circular leader that is less than a semicircle.

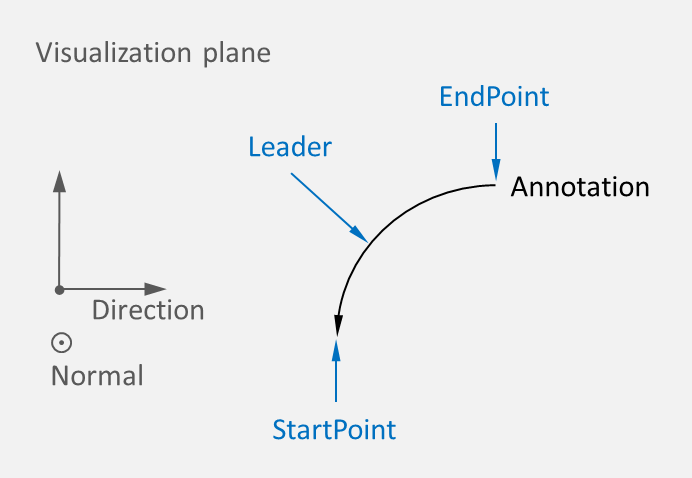


Figure 105 – Circular leader

Fields:

Field Name	Data Type	Description
StartPoint	Point2dSimpleType	The beginning point of the 2D line segment.
EndPoint	Point2dSimpleType	The ending point of the 2D line segment.
HeadForm	LeaderTypeEnumType	The first head form.

HeadHeight	xs:double	A size of the leader head.
Center	Point2dSimpleType	This field specifies 2D coordinates of the leader central point.

Example:

```
<LeaderCircular>
  <StartPoint>0.0 0.0</StartPoint>
  <EndPoint>10.0 10.0</EndPoint>
  <HeadForm>ARROW_FILLED</HeadForm>
  <HeadHeight>1.0</HeadHeight>
  <Center>10.0 0.0</Center>
</LeaderCircular>
```

7.7.3.7 Double head circular leader

LeaderDoubleHeadCircular describes a double head circular leader.

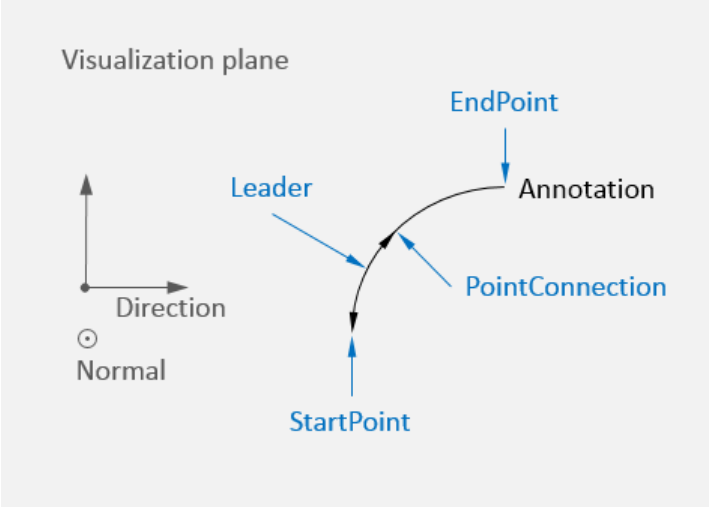


Figure 106 – Double head circular leader

Fields:

Field Name	Data Type	Description
StartPoint	Point2dSimpleType	The beginning point of the 2D line segment.
EndPoint	Point2dSimpleType	The ending point of the 2D line segment.
HeadForm	LeaderTypeEnumType	The first head form.
HeadHeight	xs:double	A size of the leader head.
HeadForm2	LeaderTypeEnumType	The second head form.
PointConnection	Point2dSimpleType	This field specifies 2D coordinates of the connection point.

Center	Point2dSimpleType	This field specifies 2D coordinates of the leader central point.

Example:

```
<LeaderDoubleHeadCircular>
  <StartPoint>0.0 0.0</StartPoint>
  <EndPoint>10.0 10.0</EndPoint>
  <HeadForm>ARROW_FILLED</HeadForm>
  <HeadHeight>1.0</HeadHeight>
  <HeadForm2>ARROW_FILLED</HeadForm2>
  <PointConnection>2.93 7.07</PointConnection>
  <Center>10.0 0.0</Center>
</LeaderDoubleHeadCircular>
```

7.7.3.8 Witness Lines

WitnessLines describes witness lines.

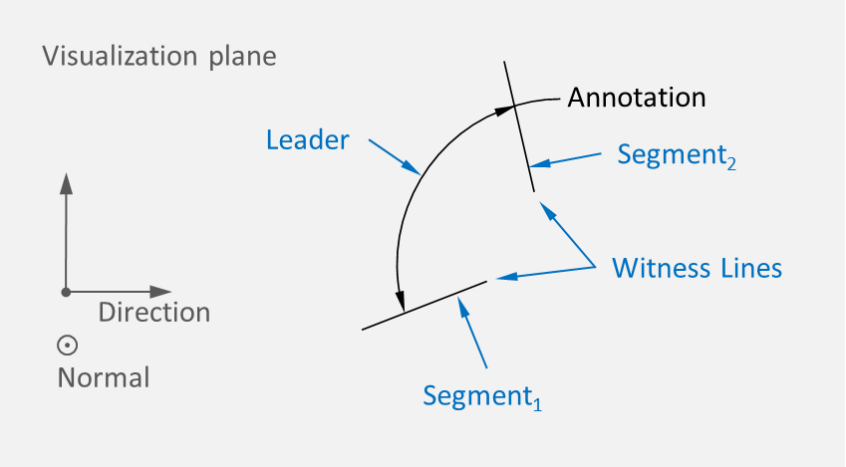


Figure 107 – Witness Lines

Fields:

Field Name	Data Type	Description
@width	xs:double	The width of the witness lines in points.
Segment1/StartPoint	Point2dSimpleType	The beginning point of the first 2D line segment.
Segment1/EndPoint	Point2dSimpleType	The ending point of the first 2D line segment.
Segment2/StartPoint	Point2dSimpleType	The beginning point of the second 2D line segment.
Segment2/EndPoint	Point2dSimpleType	The ending point of the second 2D line segment.

Example:

```
<WitnessLines width="1">
  <Segment1>
    <StartPoint>-73.225 87.343</StartPoint>
    <EndPoint>-73.225 -0.358</EndPoint>
  </Segment1>
  <Segment2>
    <StartPoint>-38.225 87.343</StartPoint>
    <EndPoint>-38.225 -0.358</EndPoint>
  </Segment2>
</WitnessLines>
```

7.7.3.9 Rectangular frame

FrameRectangular describes a rectangular frame.

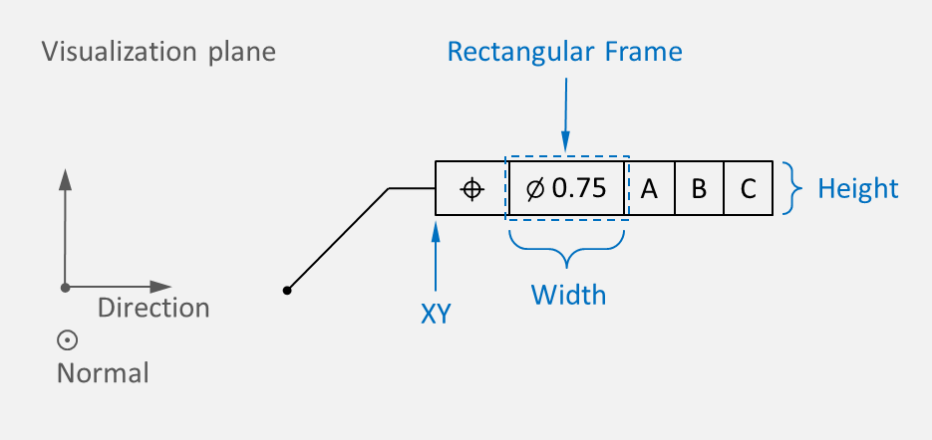


Figure 108 – Rectangular frame

Fields:

Field Name	Data Type	Description
XY	Point2dSimpleType	2D coordinates of the frame left bottom corner point (the anchor point of the rectangular frame).
Width	xs:double	The frame width.
Height	xs:double	The frame height.

Example:

```
<FrameRectangular>
  <XY>-0.2 -0.2</XY>
  <Width>1.6</Width>
  <Height>1.0</Height>
</FrameRectangular>
```


7.7.3.10 Circular frame

FrameCircularType describes a circular frame, which is normally used for visualization of datum targets.

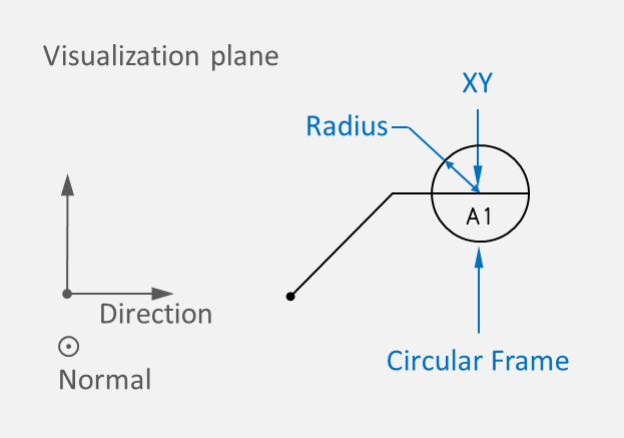


Figure 109 – Circular frame

Fields:

Field Name	Data Type	Description
@crossed	xs:boolean	The optional crossed attribute shows if the frame must be crossed with the middle line which separates the circular frame into halves (top and bottom).
XY	Point2dSimpleType	2D coordinates of the frame center (the anchor point of the circular frame).
Radius	xs:double	The frame radius.

Example:

```
<FrameCircular>
  <XY>-0.2 -0.2</XY>
  <Radius>7.3</Radius>
</FrameCircular>
```

7.7.3.11 Flag frame

FrameFlag describes a flag frame, which is normally used for visualization of flag notes.

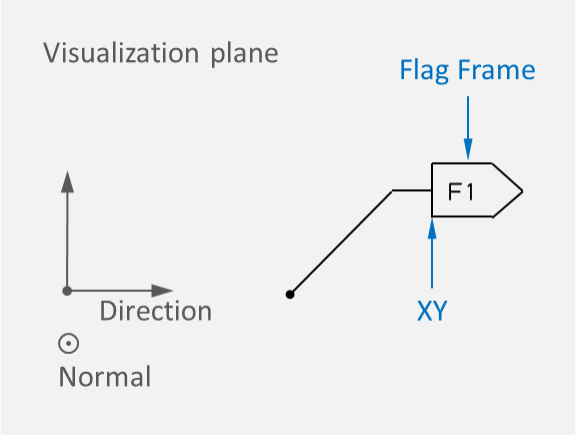


Figure 110 – Flag frame

Fields:

Field Name	Data Type	Description
@right	xs:boolean	This attribute shows if the flag frame has the triangle element at the right side.
XY	Point2dSimpleType	2D coordinates of the frame left bottom corner point (the anchor point of the flag frame).
Width	xs:double	The frame width.
Height	xs:double	The frame height.

Example:

```
<FrameFlag right="1">
  <XY>-0.2 -0.2</XY>
  <Width>19.2</Width>
  <Height>4.1</Height>
</FrameFlag>
```


7.7.3.12 Irregular form frame

FrameIrregularForm defines an irregular form frame.

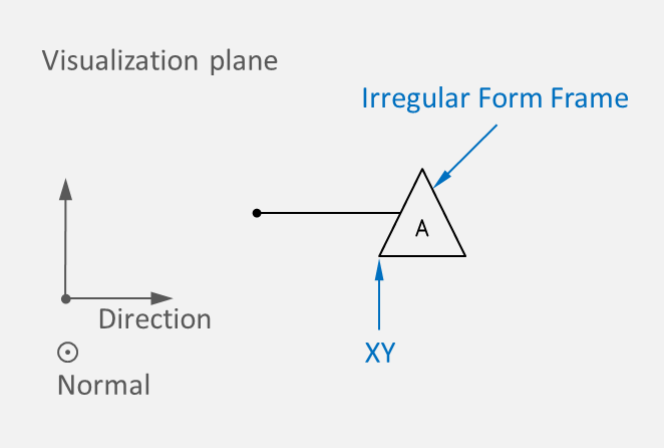


Figure 111 – Irregular form frame

Fields:

Field Name	Data Type	Description
XY	Point2dSimpleType	2D coordinates of the frame the polyline begin point (the anchor point of the irregular form frame).
Points	ArrayPoint2dType	A 2D polyline which describes the frame shape. First point of polyline stored in XY and not included in this array.

Example:

```
<FrameIrregularForm>
  <XY>-0.2 -0.2</XY>
  <Points N="5">
    0.7 -0.2
    0.7 0.8
    -0.2 0.8
    -0.2 -0.2
    0.7 -0.2
  </Points>
</FrameIrregularForm>
```

7.7.4 Camera

Camera describes a camera – projection of the 3D model space to 2D window.

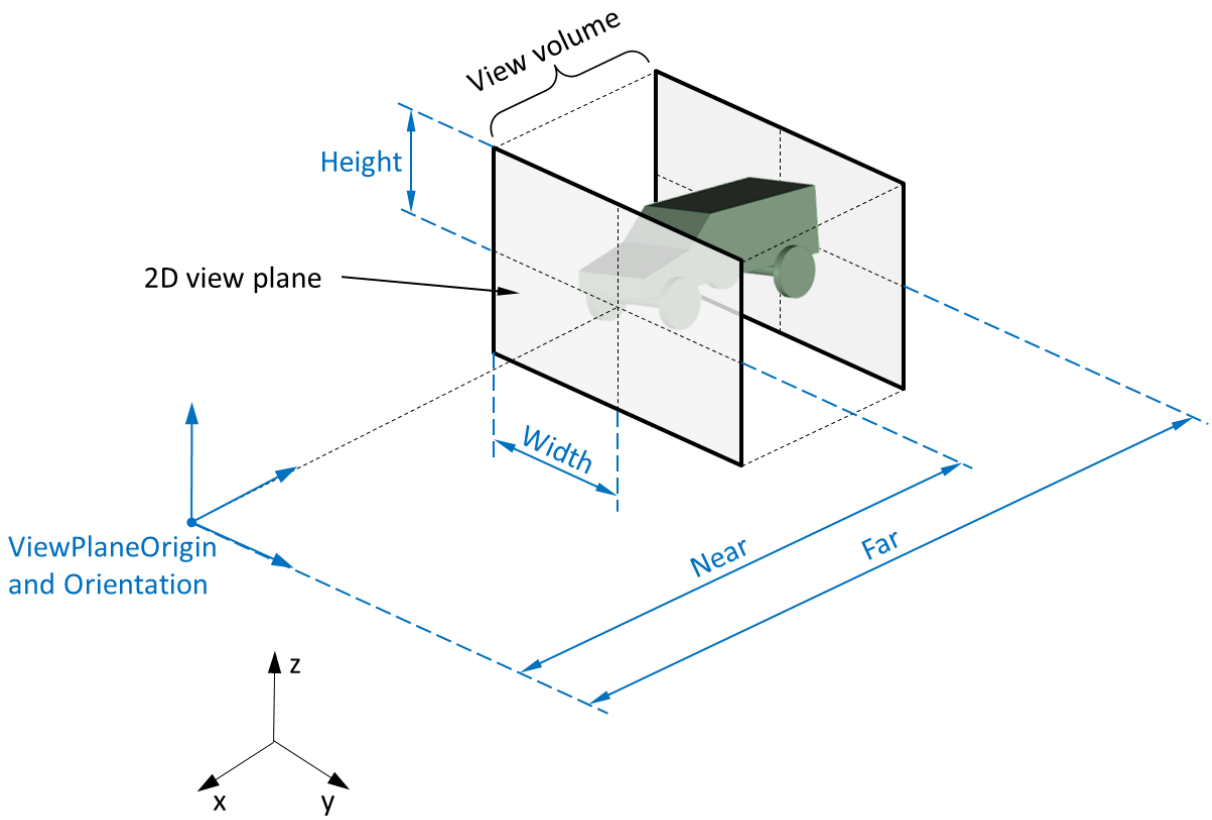


Figure 112 – Orthographic Camera

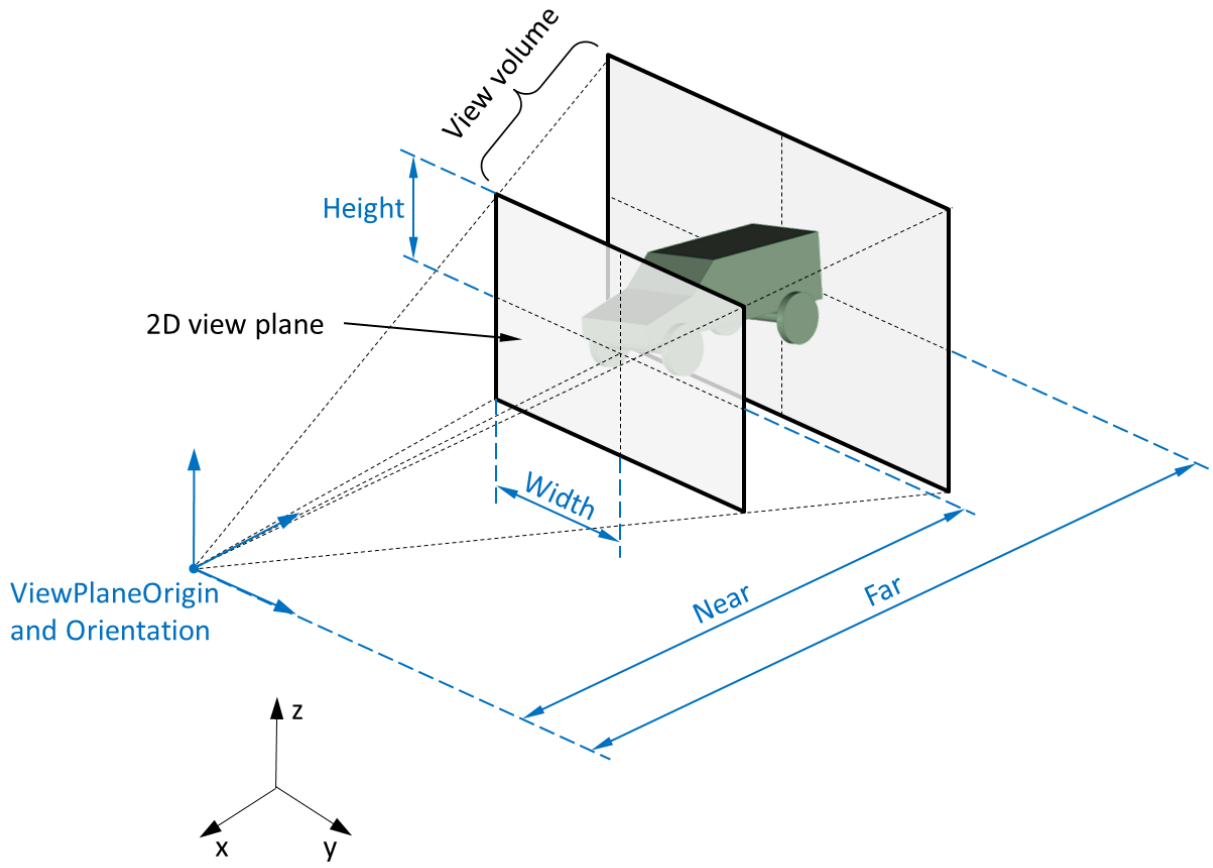


Figure 113 – Perspective Camera

$$Width = Height * Ratio$$

Fields:

Field Name	Data Type	Description
@form	CameraFormEnumType	This attribute specifies the camera type: 'ORTHOGRAPHIC' or 'PERSPECTIVE'.
ViewPlaneOrigin	PointSimpleType	The view plane origin.
Orientation	QuaternionType	The rotation of the view plane around the view plane origin.
Ratio	xs:double	The aspect ratio of the view plane (normally it corresponds to the viewport).
Near	xs:double	The distance from the view plane to the near clipping plane.
Far	xs:double	The distance from the view plane to the far clipping plane.
Height	xs:double	Half of the top to bottom extent of the 2D view plane.

Example:

```
<Camera id="348">
  <ViewPlaneOrigin>0.467 0.587 0.020</ViewPlaneOrigin>
  <Orientation>
    <Value>0.285 0.267 -0.185 -0.901</Value>
  </Orientation>
  <Ratio>1.465</Ratio>
  <Near>-2.449</Near>
  <Far>2.449</Far>
  <Height>1.079</Height>
</Camera>
```

7.7.5 Saved View

SavedView describes a saved view to facilitate representation of the model and its 3D annotations.

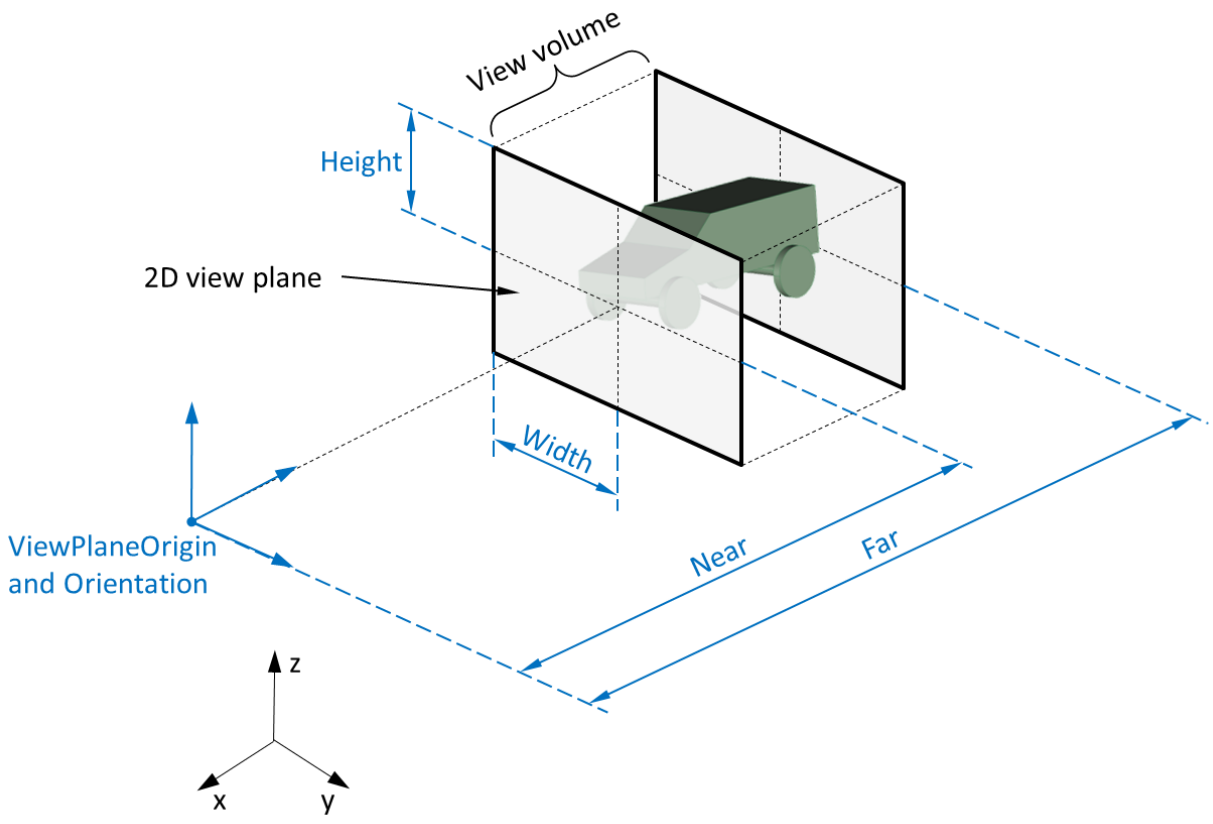


Figure 114 – Saved view orthographic camera

Fields:

Field Name	Data Type	Description
ViewPlaneOrigin	PointSimpleType	The view plane origin.
Orientation	QuaternionType	The rotation of the view plane around the view plane origin.
Ratio	xs:double	The aspect ratio of the view plane (normally it corresponds to the viewport).
Near	xs:double	The distance from the view plane to the near clipping plane.
Far	xs:double	The distance from the view plane to the far clipping plane.
Height	xs:double	Half of the top to bottom extent of the 2D view plane.
CharacteristicNominalVisibleIds	ArrayReferenceFullType	An array of identifiers of model characteristics which must be visible in this saved view.
CharacteristicNominalHiddenIds	ArrayReferenceFullType	An array of identifiers of model characteristics which must be hidden in this saved view.
BodyIds	ArrayReferenceFullType	An array of identifiers of model bodies which must be visible in this saved view. If the BodyIds element is absent, then all model bodies are visible in this saved view.
ComponentIds	ArrayReferenceFullType	An array of identifiers of model components which must be visible in this saved view. If the ComponentIds element is absent, then all model components are visible in this saved view.
PlaneClippingIds	ArrayReferenceFullType	An array of identifiers of model clipping planes which must be activated in this saved view.

Example:

```
<SavedView id="348">
  <ViewPlaneOrigin>0.467 0.587 0.020</ViewPlaneOrigin>
  <Orientation>
    <Value>0.285 0.267 -0.185 -0.901</Value>
  </Orientation>
  <Ratio>1.465</Ratio>
  <Near>-2.449</Near>
  <Far>2.449</Far>
  <Height>1.079</Height>
  <CharacteristicNominalVisibleIds N="1">
    <Id>398</Id>
  </CharacteristicNominalVisibleIds>
  <BodyIds N="2">
    <Id>29</Id>
```

```
    <Id>34</Id>
  </BodyIds>
  <ComponentIds N="1">
    <Id>90</Id>
  </ComponentIds>
</SavedView>
```

7.8 High level description of the product data

The QIF MBD model includes the information items from the Product.xsd schema file and many of the schema files in the QIF Library. The QIF Library files are incorporated into the schema by a chain of "*include*" directives starting in the Product.xsd schema file.

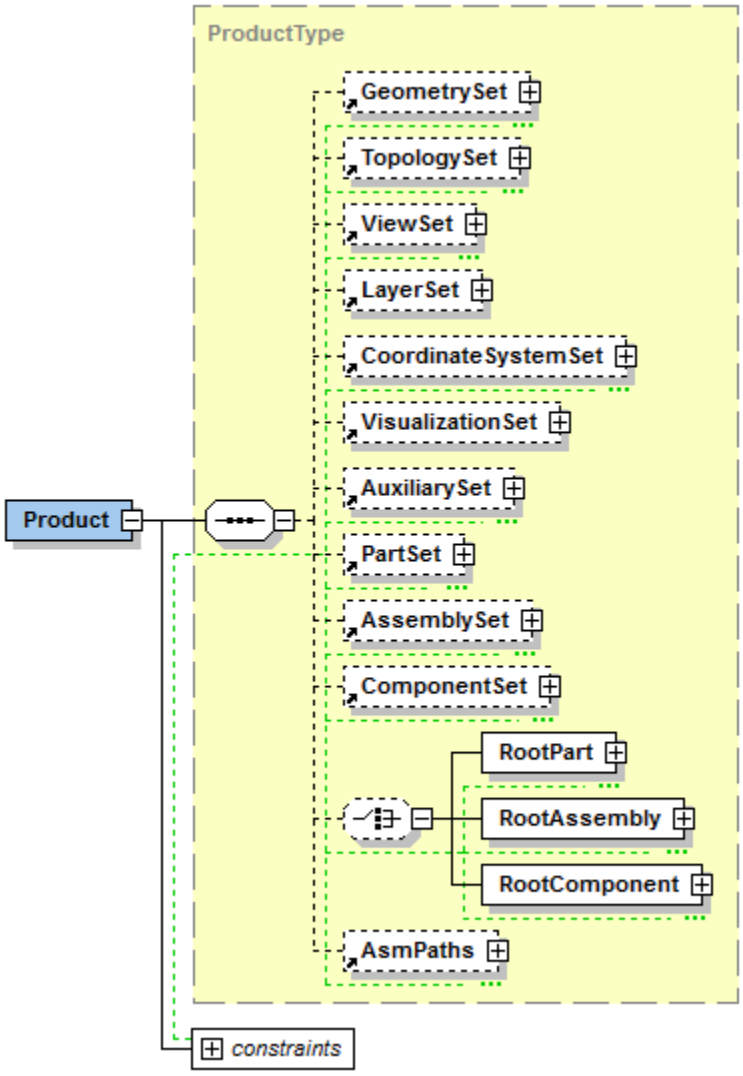


Figure 115 – High level view of QIF MBD highest level elements.

8 Data dictionary for QIF Product data model

The data dictionary for QIFProduct.xsd is normative to the QIF standard and is found in Annex C.

Annex A – Location of Product.xsd schema file

(normative)

Product.xsd is a single XML schema file that describes the QIF MBD information model. It defines several data types that are unique to the QIF MBD application area, and, in keeping with QIF design guidelines; it reuses definitions from the QIF library whenever possible. All QIF XML schema files are normative and are bundled into a single compressed folder file called "QIF_2.0_XMLSchemaFiles.zip" which is available for download at www.qifstandards.org.

Annex B - Graphical conventions of the data dictionary

(informative)

This section describes the graphical conventions used in the QIF data dictionaries. The data dictionaries describe the structure of the information models and the manufacturing quality semantics of the data types.

The rules of encoding QIF instance files are primarily defined in the XML schema files, but the data dictionaries express many of the same requirements via the pictures and table entries.

Data type definitions are indicated by a box with beveled corners on the left side, as in Figure B.1.

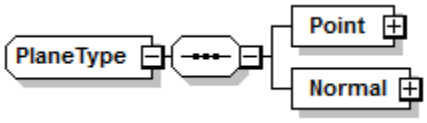


Figure B.1 – Notation for a type definition, *PlaneType*.

Rectangular boxes indicate data *elements*. A solid rectangle indicates a required *element*, whereas a dotted rectangle indicates an optional *element*. If an object is not designated optional, then it is required by default. Small boxes on the right hand end of *element* boxes, containing either "-" or "+" are used to indicate one of the following conditions exist:

- a “+” indicates that the additional structures or *elements* below this node have been hidden in this diagram.
- a “-“ indicates that additional structures or *elements* below this node exist and are visible on the diagram.

The absence of any box at the right hand end of an *element* box indicates that the type of the *element* is a primitive type without any substructure, e.g., xs:decimal. In this case, there will also be three bars in the upper left corner of the *element* box. The beveled box with 3 dots on a line represents the XSDL *sequence* operator. It indicates that the object to the left is composed of all of the *elements* to the right, in that specified order.

Type definitions can be reused to generate data *elements*, as shown by a yellow box in dotted lines, with the name of the type definition at the top. Figure B.2 shows that **ZonePlane** is an *element* of type ***PlaneType***.

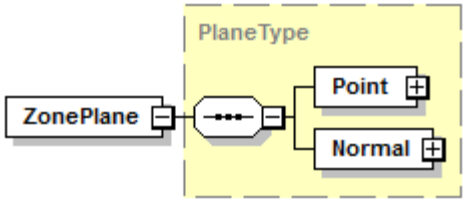


Figure B.2 – Reuse of the type definition *PlaneType* to generate element ZonePlane.

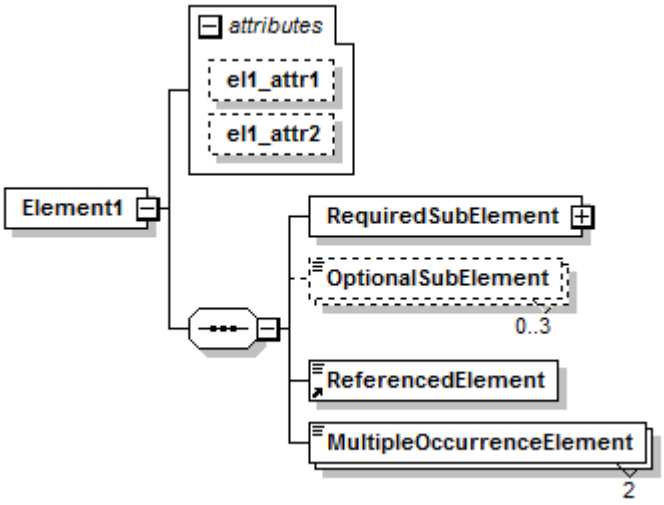


Figure B.3 – Notation for *elements*, sub-*elements*, and *attributes*.

Figure B.3 contains examples of numerous information modeling notations. *Element* definitions in XML schema files can be reused by "reference", indicated by an arrow in the lower left corner of the **ReferencedElement** box. *Elements* may appear in an XML instance document more than once. Figure B.3. shows the **OptionalSubElement** notated with two numerals separated by an ellipsis, "0..3", that indicates the number of occurrences as an inclusive range. The **OptionalSubElement** may occur zero,1, 2, or 3 times as sub-*elements* of **Element1**. Where there is a single cardinality numeral, the *element* must occur exactly that number of times in the instance file. For example, the *element* **MultipleOccurrenceElement** must occur exactly two times as sub-*elements* of **Element1**. Information items can be instantiated in XSDL as *elements* or *attributes*. An *element*'s *attributes* are shown in the data dictionaries as solid-lined boxes that are explicitly labeled *attributes*, as shown at the top of the diagram.

Figure B.4 shows an example *element* definition where exactly one of the three sub-*element* choices must be given. The beveled box with three square dots and a "switch" line ( ) indicate the XSDL *choice* structure. When **Element2** is instantiated in an XML instance file, it must have exactly one sub-*element* chosen among the three sub-*elements* shown.

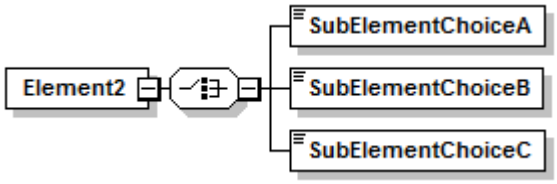


Figure B.4 – The *choice of* notation.

The data dictionaries are grouped by XML schema file. It is characteristic of QIF definitions to use types declared in other XML schema files. The sharing of definitions specified in other files is indicated by the XML schema file directive *include*.

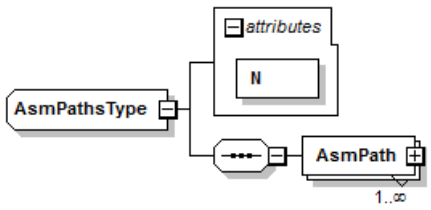
Annex C – Data dictionary for QIFProduct.xsd
(normative)

schema location: **..\QIFApplications\QIFProduct.xsd**
attributeFormDefault: **unqualified**
elementFormDefault: **qualified**
targetNamespace: **http://qifstandards.org/xsd/qif2**

Complex types
AsmPathsType
AsmPathType
AssemblySetType
AssemblyType
ComponentSetType
ComponentType
DefinitionExternalType
DigitalDrawingType
DigitalModelType
FileInternalType
InternalComponentType
InternalHeaderType
InternalPartAssemblyType
LayerSetType
LayerType
NoteFlagSetType
NoteFlagType
NoteSetType
NoteType
PartFamilyType
PartNoteSetType
PartNoteType
PartSetType
PartType
PhysicalModelType
PrintedDrawingType
ProductDefinitionBaseType
ProductType

Simple types
GDTEnumType
NoteFormEnumType
TopologyEnumType

complexType **AsmPathsType**

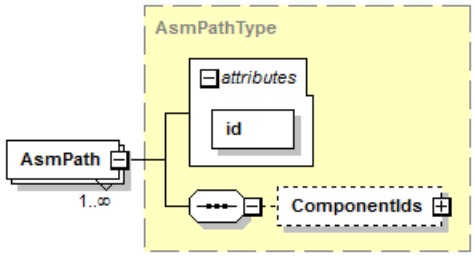
diagram	
children	**AsmPath**
used by	element **AsmPaths**
attributes	Name Type Use Default Fixed Annotation

	N	**NaturalType**	required	documentation The required N attribute is the number of objects in this set.
annotation	documentation The AsmPathsType defines a list of one or more assembly paths.			

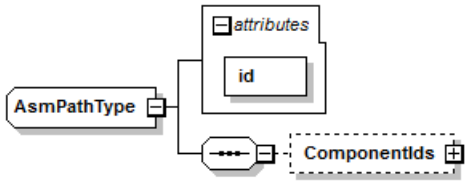
attribute **AsmPathsType/@N**

type	**NaturalType**
properties	use required
facets	Kind: minInclusive; Value: 1; Annotation
annotation	documentation The required N attribute is the number of objects in this set.

element **AsmPathsType/AsmPath**

diagram	
type	**AsmPathType**
properties	minOcc 1 maxOcc unbounded content complex
children	**ComponentIds**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default; Fixed; Annotation: documentation The required id attribute is the QIF id of the assembly path, used for referencing.
annotation	documentation Each AsmPath element is an assembly path in the list.

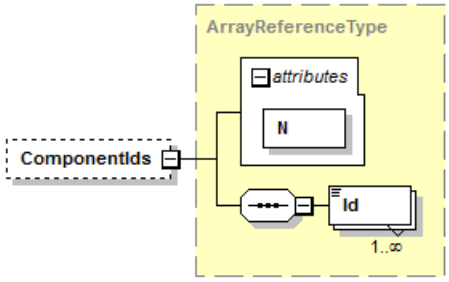
complexType **AsmPathType**

diagram	
children	**ComponentIds**
used by	element **AsmPathsType/AsmPath**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default; Fixed; Annotation: documentation The required id attribute is the QIF id of the assembly path, used for referencing.
annotation	documentation The AsmPathType defines information about assembly path, this path is used for identification of an object inside an assembly.

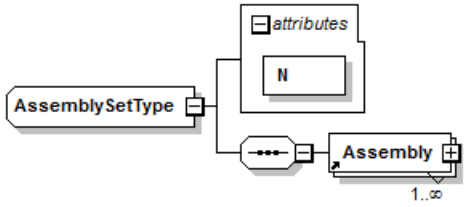
attribute **AsmPathType/@id**

type	**QIFIdType**
properties	use required
annotation	documentation The required id attribute is the QIF id of the assembly path, used for referencing.

element **AsmPathType/ComponentIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The ComponentIds element is an array of identifiers of the scene components which contain this entity. This array shows the "path" from the root to the parent component. The id of the parent component must be the last element of this array.

complexType **AssemblySetType**

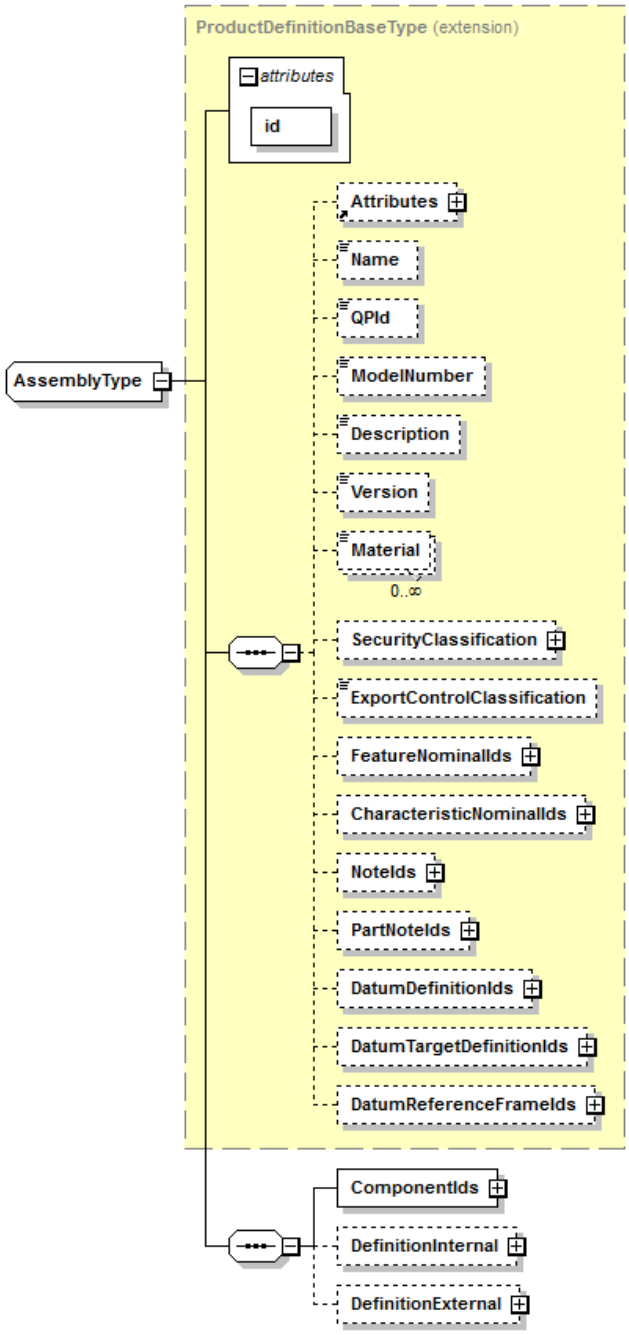
diagram	
children	**Assembly**
used by	element **AssemblySet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute is the number of objects in this set.
annotation	documentation The AssemblySetType represents a container for storing all assemblies present in the CAD scene.

attribute **AssemblySetType/@N**

type	**NaturalType**
properties	use required
facets	Kind Value Annotation minInclusive 1
annotation	documentation The required N attribute is the number of objects in this set.

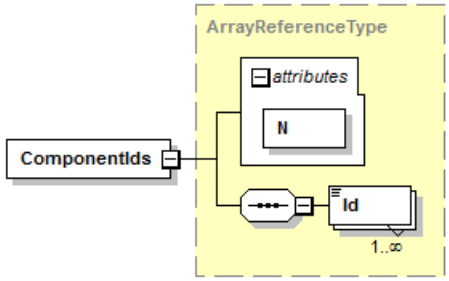
complexType **AssemblyType**

diagram



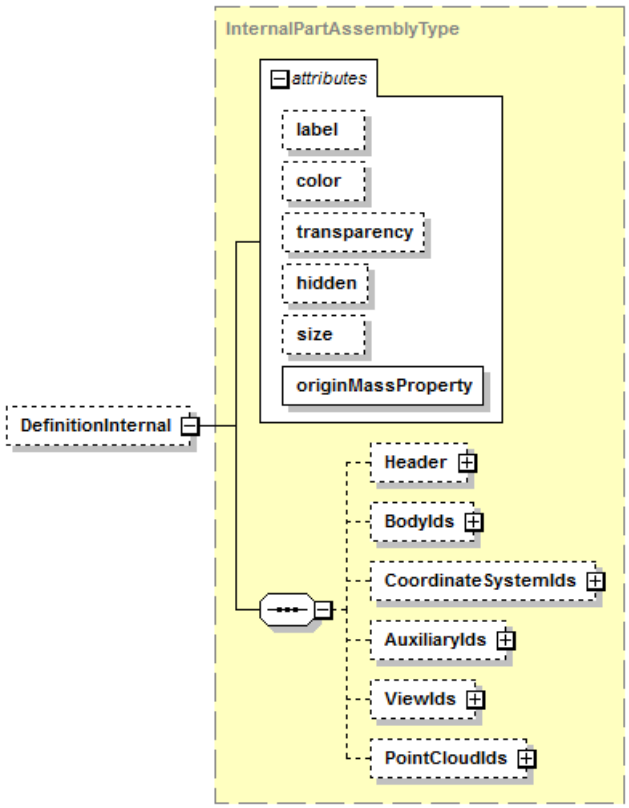
type	extension of **ProductDefinitionBaseType**
properties	base ProductDefinitionBaseType
children	**Attributes** **Name** **QPId** **ModelNumber** **Description** **Version** **Material** **SecurityClassification** **ExportControlClassification** **FeatureNominalIds** **CharacteristicNominalIds** **NoteIds** **PartNoteIds** **DatumDefinitionIds** **DatumTargetDefinitionIds** **DatumReferenceFrameIds** **ComponentIds** **DefinitionInternal** **DefinitionExternal**
used by	element **Assembly**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the product definition, used for referencing.
annotation	documentation: The AssemblyType is the assembly type. It contains a set of entities, defining a feature (body and sub-components), which can be instantiated multiple times in the CAD scene. Use of assemblies simplifies maintenance of identical features and reduces the total amount of scene data.

element **AssemblyType/ComponentIds**

diagram	
type	**ArrayReferenceType**
properties	content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation: The ComponentIds element is an array of component identifiers present in this product definition.

element **AssemblyType/DefinitionInternal**

diagram



type **InternalPartAssemblyType**

properties

minOcc	0
maxOcc	1
content	complex

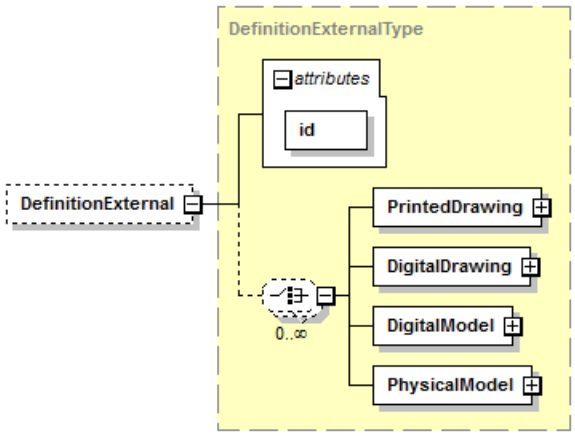
children **Header** **BodyIds** **CoordinateSystemIds** **AuxiliaryIds** **ViewIds** **PointCloudIds**

attributes

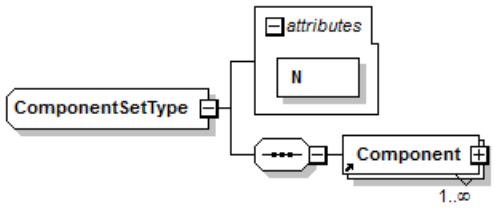
Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.
size	**DoublePositiveType**				documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system).

	originMassProperty	**PointSimpleType**	required	documentation The required originMassProperty attribute is the origin point for calculating of objects mass properties contained in this block.
annotation	documentation The optional DefinitionInternal element is an assembly definition in QIF format.			

element **AssemblyType/DefinitionExternal**

diagram	
type	**DefinitionExternalType**
properties	minOcc 0 maxOcc 1 content complex
children	**PrintedDrawing DigitalDrawing DigitalModel PhysicalModel**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the product geometry definitions, used for referencing.
annotation	documentation The optional DefinitionExternal element is an assembly definition in non-QIF format.

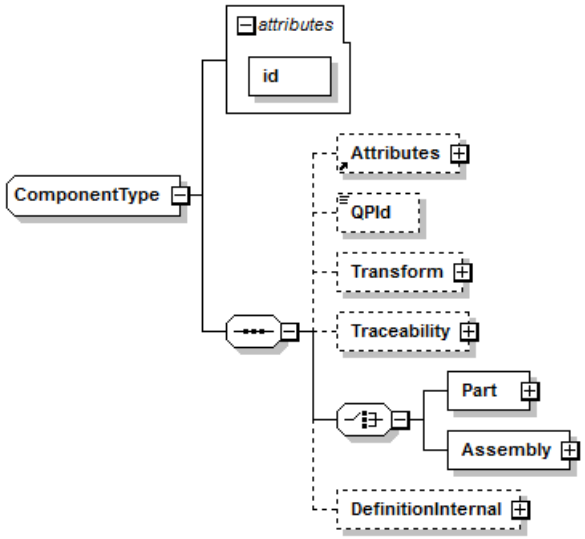
complexType **ComponentSetType**

diagram	
children	**Component**
used by	element **ComponentSet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute is the number of objects in this set.
annotation	documentation The ComponentSetType represents a container for storing all components present in the CAD scene.

attribute **ComponentSetType/@N**

type	**NaturalType**
properties	use required
facets	Kind Value Annotation minInclusive 1
annotation	documentation The required N attribute is the number of objects in this set.

complexType **ComponentType**

diagram	
children	**Attributes** **QPId** **Transform** **Traceability** **Part** **Assembly** **DefinitionInternal**
used by	element **Component**
attributes	Name Type Use Default Fixed Annotation id **QIFIdType** required documentation The required id attribute is the QIF id of the component, used for referencing.
annotation	documentation The ComponentType defines a single instance of a part or an assembly.

attribute **ComponentType/@id**

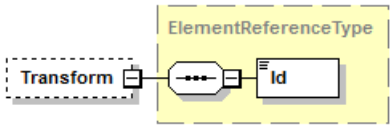
type	**QIFIdType**
properties	use required
annotation	documentation The required id attribute is the QIF id of the component, used for referencing.

element **ComponentType/QPId**

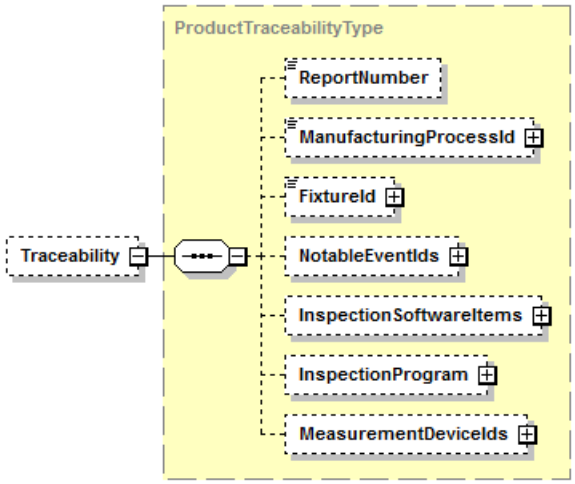
diagram	
type	**QPIdType**
properties	minOcc 0 maxOcc 1 content simple

annotation	documentation The optional QPId element is a persistent identifier for the component. If used, it should be generated using a widely accepted UUID generator.

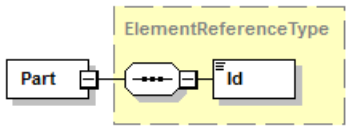
element **ComponentType/Transform**

diagram	
type	**ElementReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
annotation	documentation The optional Transform element is an identifier of the transformation matrix that maps the component into the product.

element **ComponentType/Traceability**

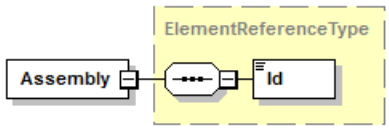
diagram	
type	**ProductTraceabilityType**
properties	minOcc 0 maxOcc 1 content complex
children	**ReportNumber ManufacturingProcessId FixtureId NotableEventIds InspectionSoftwareItems InspectionProgram MeasurementDeviceIds**
annotation	documentation The optional Traceability element gives traceability information for a component.

element **ComponentType/Part**

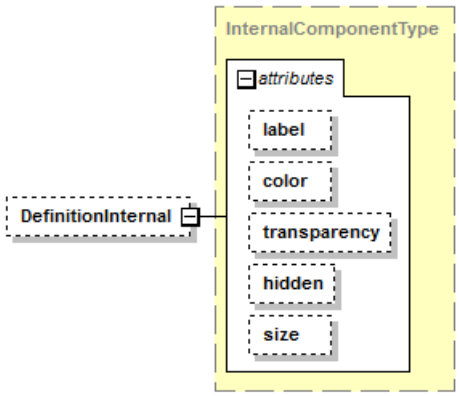
diagram	

type	**ElementReferenceType**
properties	content complex
children	**Id**
used by	complexType **PartSetType**
annotation	documentation The Part element is an identifier of a part to be instantiated.

element **ComponentType/Assembly**

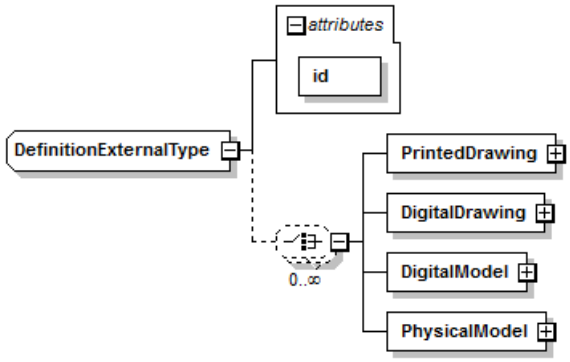
diagram	
type	**ElementReferenceType**
properties	content complex
children	**Id**
used by	complexType **AssemblySetType**
annotation	documentation The Assembly element is an identifier of an assembly to be instantiated.

element **ComponentType/DefinitionInternal**

diagram	
type	**InternalComponentType**
properties	minOcc 0 maxOcc 1 content complex
attributes	Name / Type / Use / Default / Fixed / Annotation label — **xs:string** — documentation: The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree. color — **ColorType** — documentation: The optional color attribute defines the RGB color property of a model entity. transparency — **xs:double** — 0.0 — documentation: The optional transparency attribute defines the transparency property of a

	hidden	**xs:boolean**	0	model entity. documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.
	size	**DoublePositiveType**		documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system).
annotation	documentation The optional DefinitionInternal element is the definition of the design of the component in QIF format.			

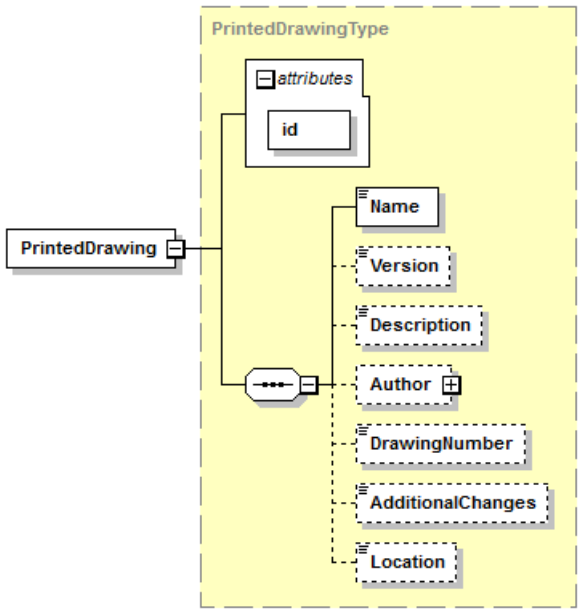
complexType **DefinitionExternalType**

diagram	
children	**PrintedDrawing** **DigitalDrawing** **DigitalModel** **PhysicalModel**
used by	elements **PartType/DefinitionExternal** **AssemblyType/DefinitionExternal**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the product geometry definitions, used for referencing.
annotation	documentation The DefinitionExternalType defines the geometry of a product (a part or an assembly) in various degrees of detail.

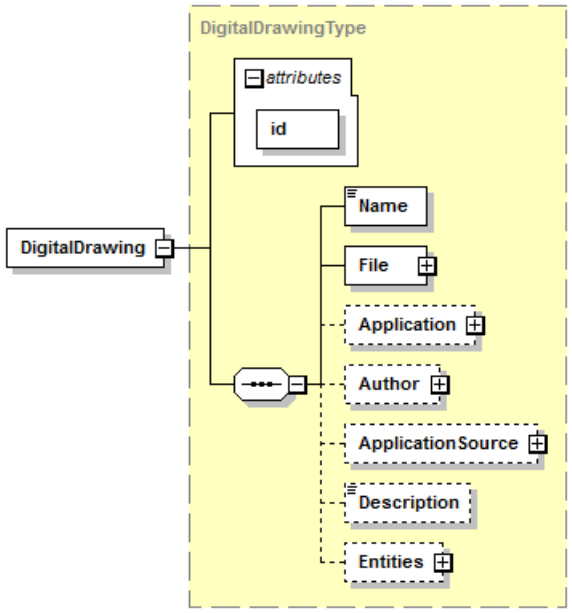
attribute **DefinitionExternalType/@id**

type	**QIFIdType**
properties	use required
annotation	documentation The required id attribute is the QIF id of the product geometry definitions, used for referencing.

element **DefinitionExternalType/PrintedDrawing**

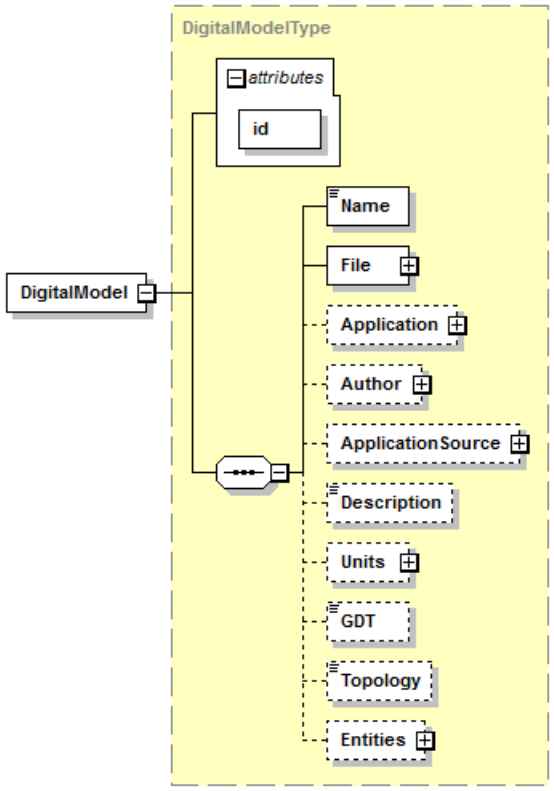
diagram	
type	**PrintedDrawingType**
properties	content complex
children	**Name Version Description Author DrawingNumber AdditionalChanges Location**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the printed drawing.
annotation	documentation: The PrintedDrawing element gives information about a printed part drawing. This element is in an optional choice.

element **DefinitionExternalType/DigitalDrawing**

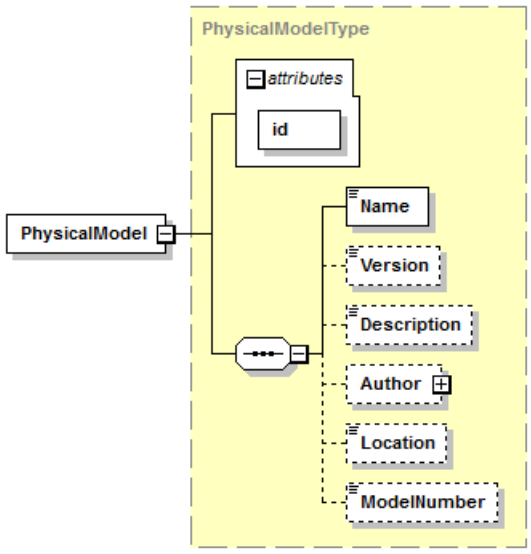
diagram	

type	**DigitalDrawingType**
properties	content complex
children	**Name File Application Author ApplicationSource Description Entities**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the digital drawing.
annotation	documentation: The DigitalDrawing element gives information about a digital part drawing. This element is in an optional choice.

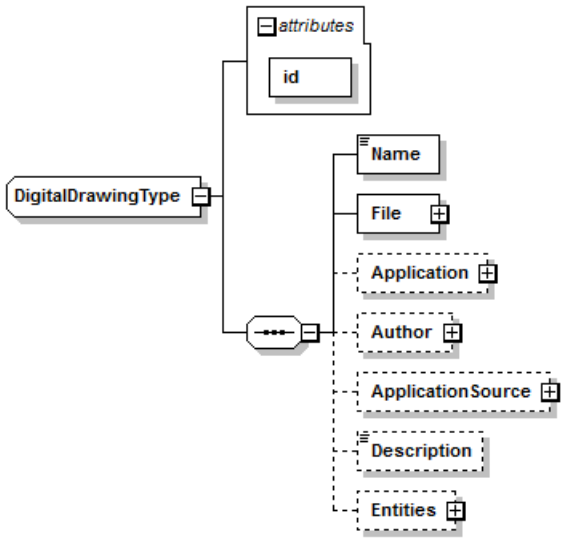
element **DefinitionExternalType/DigitalModel**

diagram	
type	**DigitalModelType**
properties	content complex
children	**Name File Application Author ApplicationSource Description Units GDT Topology Entities**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the digital design model.
annotation	documentation: The DigitalModel element gives information about a digital part design model. This element is in an optional choice.

element **DefinitionExternalType/PhysicalModel**

diagram	
type	**PhysicalModelType**
properties	content complex
children	**Name Version Description Author Location ModelNumber**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation — The required id attribute is the QIF id of the physical model.
annotation	documentation The PhysicalModel element is an actual, hands-on, physical, 3D model or prototype of a part that is used to communicate features, datum reference frames, characteristics, or other information. Examples of physical models include actual part instances, part instances made from other materials, and 3D printing and stereolithography models. This element is in an optional choice.

complexType **DigitalDrawingType**

diagram	
children	**Name File Application Author ApplicationSource Description Entities**

used by	element **DefinitionExternalType/DigitalDrawing**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the digital drawing.
annotation	documentation: The DigitalDrawingType defines an electronic version of a drawing. It has only 2D drawing information. Any geometric dimensioning and tolerancing information is drawn with annotation symbols.

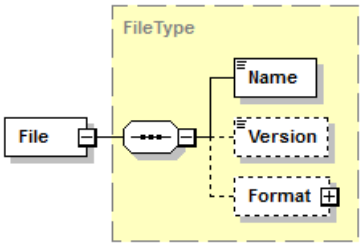
attribute **DigitalDrawingType/@id**

type	**QIFIdType**
properties	use required
annotation	documentation: The required id attribute is the QIF id of the digital drawing.

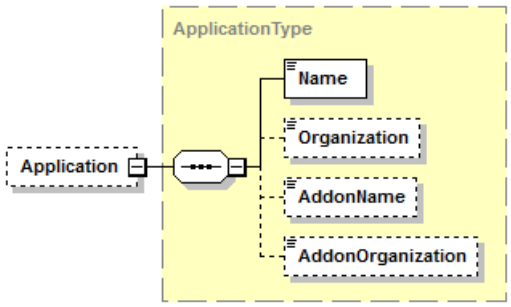
element **DigitalDrawingType/Name**

diagram	
type	**xs:string**
properties	content simple
annotation	documentation: The Name element is the name of the model.

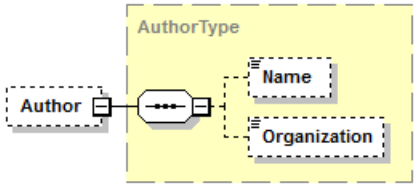
element **DigitalDrawingType/File**

diagram	
type	**FileType**
properties	content complex
children	**Name Version Format**
annotation	documentation: The File element specifies the file used in the model.

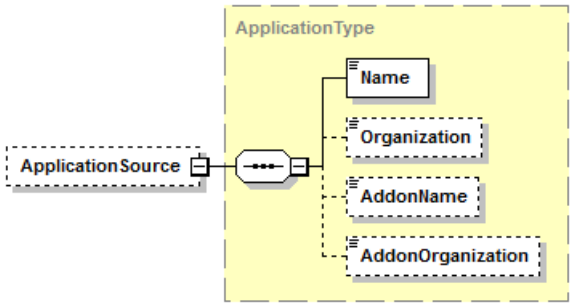
element **DigitalDrawingType/Application**

diagram	
type	**ApplicationType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization AddonName AddonOrganization**
annotation	documentation The optional Application element is information about the software application wherein the model was most recently edited.

element **DigitalDrawingType/Author**

diagram	
type	**AuthorType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization**
annotation	documentation The optional Author element is the author who created this file.

element **DigitalDrawingType/ApplicationSource**

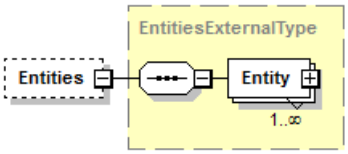
diagram	
type	**ApplicationType**
properties	minOcc 0 maxOcc 1 content complex

children	**Name Organization AddonName AddonOrganization**
annotation	documentation The optional ApplicationSource element is the name of the software application wherein the model was created.

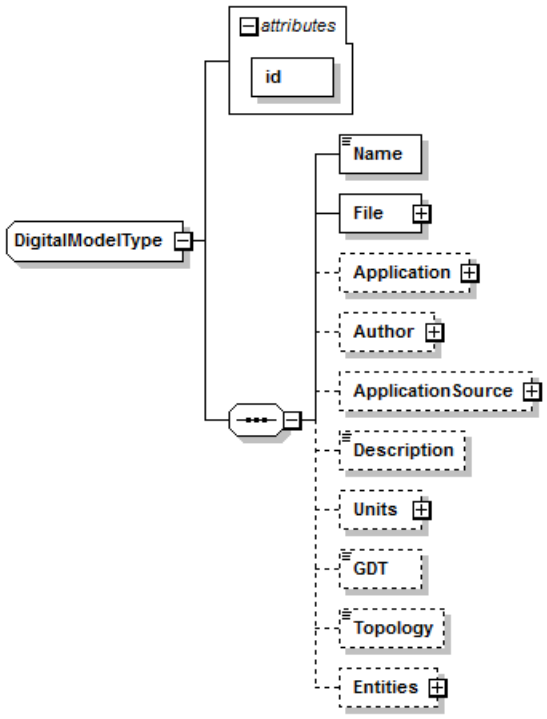
element **DigitalDrawingType/Description**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Description element is a description of the model.

element **DigitalDrawingType/Entities**

diagram	
type	**EntitiesExternalType**
properties	minOcc 0 maxOcc 1 content complex
children	**Entity**
annotation	documentation The optional Entities element is a list of instances of the EntityExternalType associated with the model. Only those entities from the model that need to be referenced should be included in this list.

complexType **DigitalModelType**

diagram	
children	**Name** **File** **Application** **Author** **ApplicationSource** **Description** **Units** **GDT** **Topology** **Entities**
used by	element **DefinitionExternalType/DigitalModel**
attributes	Name: **id** Type: **QIFIdType** Use: required Default: Fixed: Annotation: documentation — The required id attribute is the QIF id of the digital design model.
annotation	documentation The DigitalModelType defines a digital design model that represents information about an assembly or part.

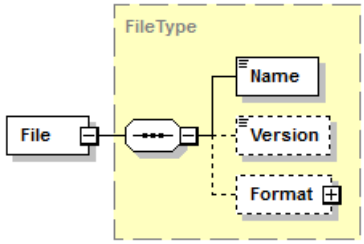
attribute **DigitalModelType/@id**

type	**QIFIdType**
properties	use required
annotation	documentation The required id attribute is the QIF id of the digital design model.

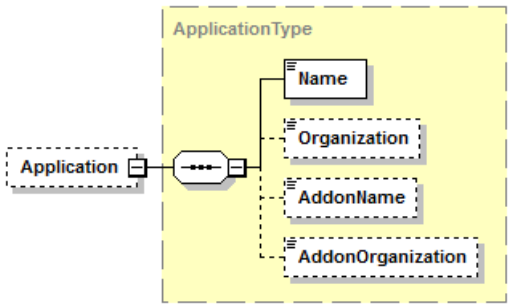
element **DigitalModelType/Name**

diagram	
type	**xs:string**
properties	content simple
annotation	documentation The Name element is the name of the model.

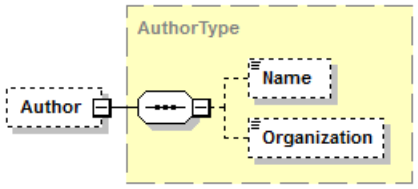
element **DigitalModelType/File**

diagram	
type	**FileType**
properties	content complex
children	**Name Version Format**
annotation	documentation The File element specifies the file used in the model.

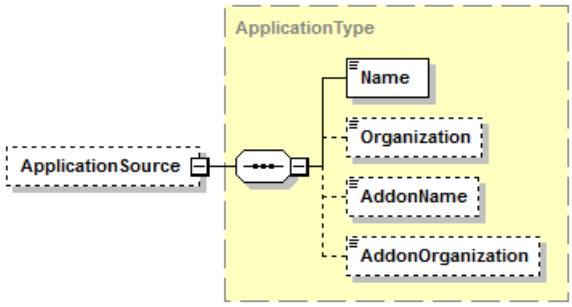
element **DigitalModelType/Application**

diagram	
type	**ApplicationType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization AddonName AddonOrganization**
annotation	documentation The optional Application element is information about the software application wherein the model was most recently edited.

element **DigitalModelType/Author**

diagram	
type	**AuthorType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization**
annotation	documentation The optional Author element is the author who created this file.

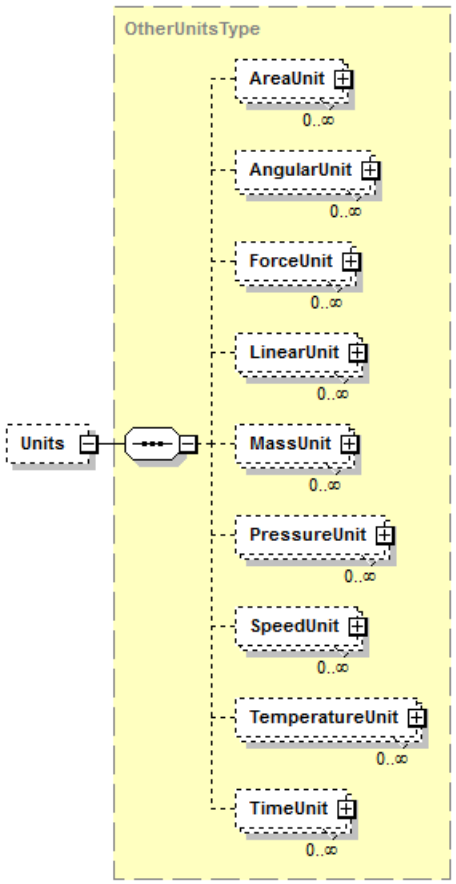
element **DigitalModelType/ApplicationSource**

diagram	
type	**ApplicationType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization AddonName AddonOrganization**
annotation	documentation The optional ApplicationSource element is the name of the software application wherein the model was created.

element **DigitalModelType/Description**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Description element is a description of the model.

element **DigitalModelType/Units**

diagram	
type	**OtherUnitsType**
properties	minOcc 0 maxOcc 1 content complex
children	**AreaUnit AngularUnit ForceUnit LinearUnit MassUnit PressureUnit SpeedUnit TemperatureUnit TimeUnit**
annotation	documentation The optional Units element specifies the units used in the model.

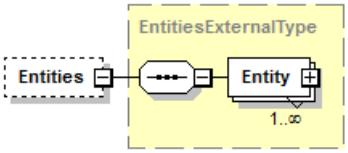
element **DigitalModelType/GDT**

diagram	
type	**GDTEnumType**
properties	minOcc 0 maxOcc 1 content simple default UNKNOWN
facets	Kind / Value / Annotation enumeration UNKNOWN enumeration HUMANREAD enumeration MACHINEREAD enumeration ABSENT
annotation	documentation The optional GDT element specifies the presence of geometric dimensioning and tolerancing information in model.

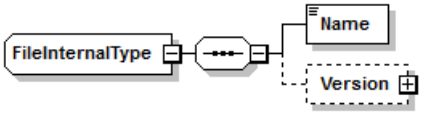
element **DigitalModelType/Topology**

diagram	
type	**TopologyEnumType**
properties	minOcc 0 maxOcc 1 content simple default UNKNOWN
facets	Kind Value Annotation enumeration UNKNOWN enumeration PRESENT enumeration ABSENT
annotation	documentation The optional Topology element specifies the presence of topology information in model.

element **DigitalModelType/Entities**

diagram	
type	**EntitiesExternalType**
properties	minOcc 0 maxOcc 1 content complex
children	**Entity**
annotation	documentation The optional Entities element is a list of instances of the EntityExternalType associated with the model. Only those entities from the model that need to be referenced should be included in this list.

complexType **FileInternalType**

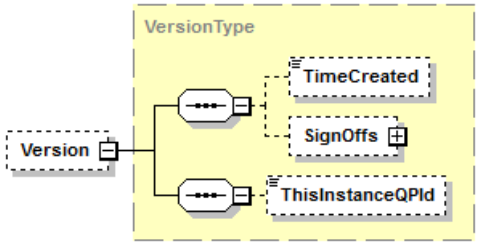
diagram	
children	**Name** **Version**
used by	element **InternalHeaderType/File**
annotation	documentation The FileInternalType defines information identifying a file in QIF format.

element **FileInternalType/Name**

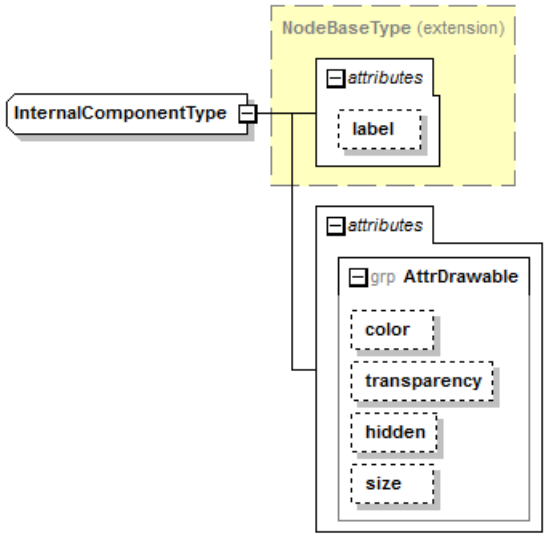
diagram	
type	**xs:string**
properties	content simple

annotation	documentation The Name element is the name of the file.

element **FileInternalType/Version**

diagram	
type	**VersionType**
properties	minOcc 0 maxOcc 1 content complex
children	**TimeCreated SignOffs ThisInstanceQPId**
annotation	documentation The optional Version element gives version information about the file.

complexType **InternalComponentType**

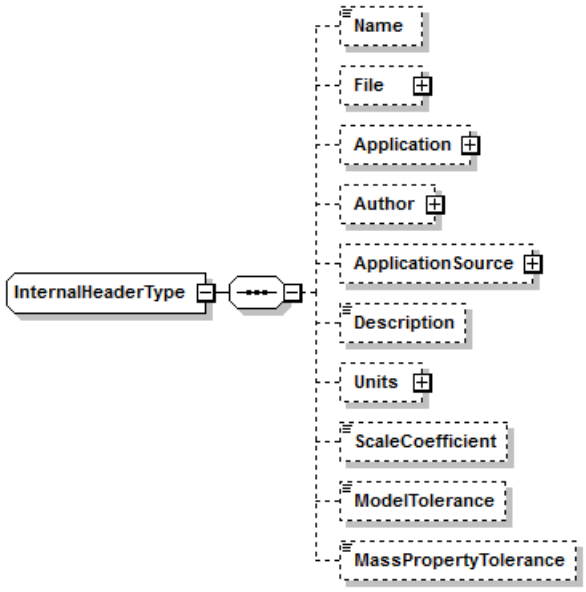
diagram	
type	extension of **NodeBaseType**
properties	base NodeBaseType
used by	element **ComponentType/DefinitionInternal**

attributes

Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.

	transparency	**xs:double**	0.0	documentation The optional transparency attribute defines the transparency property of a model entity.
	hidden	**xs:boolean**	0	documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.
	size	**DoublePositiveType**		documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system).
annotation	documentation The InternalComponentType is an instance of a part.			

complexType **InternalHeaderType**

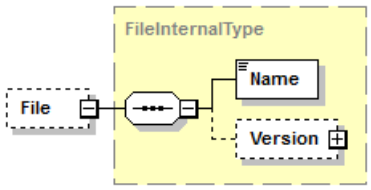
diagram	
children	**Name File Application Author ApplicationSource Description Units ScaleCoefficient ModelTolerance MassPropertyTolerance**
used by	elements **ProductType/Header InternalPartAssemblyType/Header**
annotation	documentation The InternalHeaderType defines information about the creation of the file containing the CAD model and global parameters of the model.

element **InternalHeaderType/Name**

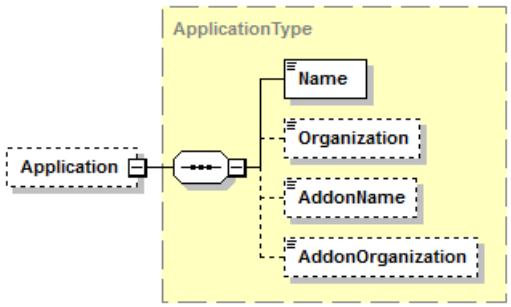
diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple

annotation	documentation The optional Name element is the name of the model.

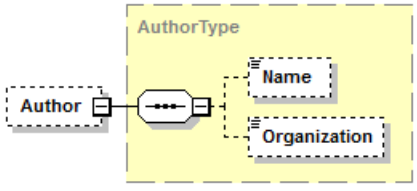
element **InternalHeaderType/File**

diagram	
type	**FileInternalType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Version**
annotation	documentation The optional File element specifies the QIF file used in the model.

element **InternalHeaderType/Application**

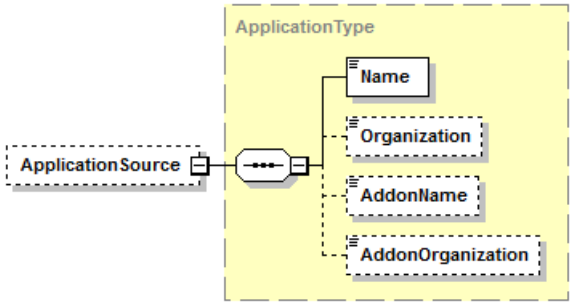
diagram	
type	**ApplicationType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization AddonName AddonOrganization**
annotation	documentation The optional Application element is information about the software application wherein the model was most recently edited.

element **InternalHeaderType/Author**

diagram	
type	**AuthorType**
properties	minOcc 0 maxOcc 1 content complex

children	**Name Organization**
annotation	documentation The optional Author element is the author who created this file.

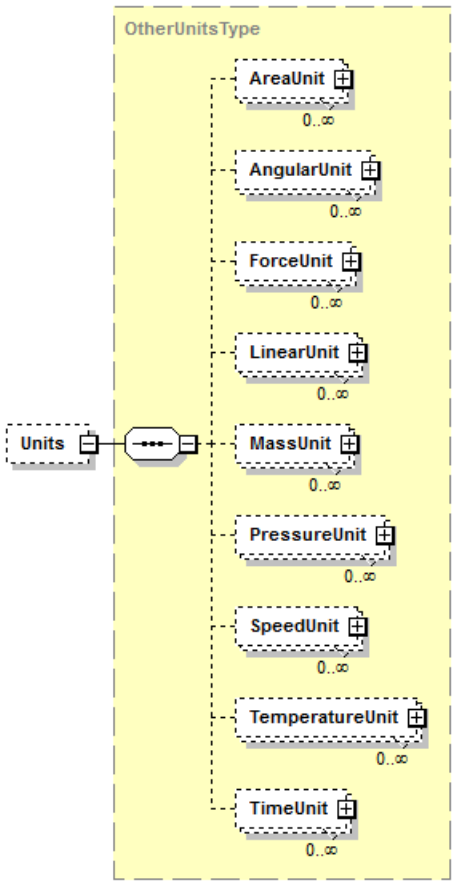
element **InternalHeaderType/ApplicationSource**

diagram	
type	**ApplicationType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization AddonName AddonOrganization**
annotation	documentation The optional ApplicationSource element is the name of the software application wherein the model was created.

element **InternalHeaderType/Description**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Description element is a description of the model.

element **InternalHeaderType/Units**

diagram	
type	**OtherUnitsType**
properties	minOcc 0 maxOcc 1 content complex
children	**AreaUnit AngularUnit ForceUnit LinearUnit MassUnit PressureUnit SpeedUnit TemperatureUnit TimeUnit**
annotation	documentation The optional Units element specifies the units used in the model.

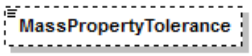
element **InternalHeaderType/ScaleCoefficient**

diagram	
type	**xs:double**
properties	minOcc 0 maxOcc 1 content simple default 1.0
annotation	documentation The optional ScaleCoef element is the global scaling coefficient. The default value is 1.

element **InternalHeaderType/ModelTolerance**

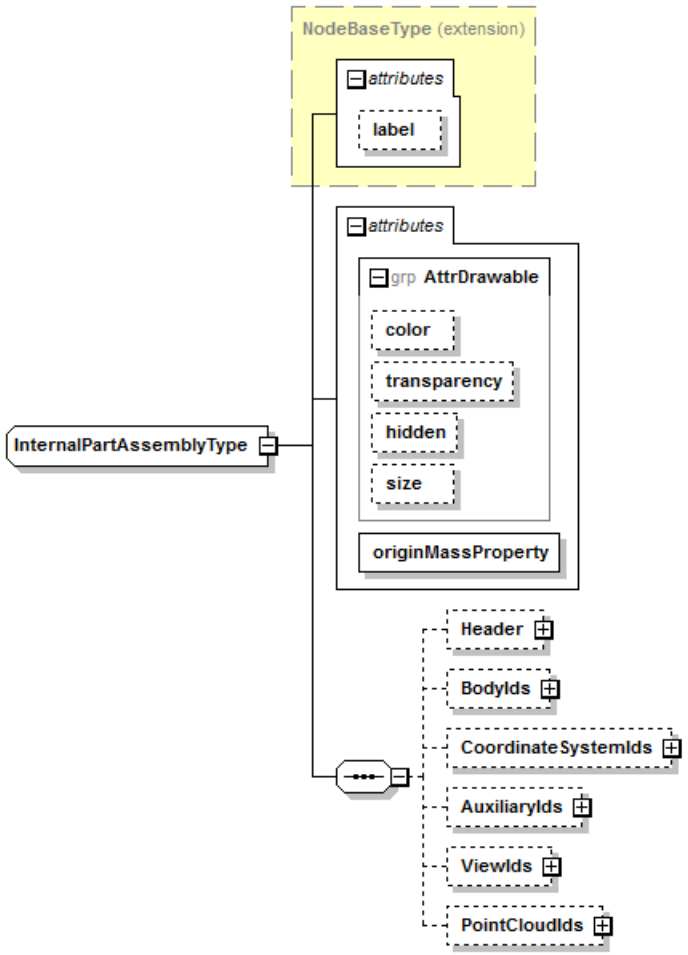
diagram	
type	**xs:double**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional ModelTolerance element is the model tolerance. This value specifies the smallest distance between coordinates, in the global units defined above, that shall be considered as distinct.

element **InternalHeaderType/MassPropertyTolerance**

diagram	
type	**xs:double**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional EpsMassProperty element specifies a tolerance used for calculation of the mass properties of model entities.

complexType **InternalPartAssemblyType**

diagram



type: extension of **NodeBaseType**

properties: base NodeBaseType

children: **Header** **BodyIds** **CoordinateSystemIds** **AuxiliaryIds** **ViewIds** **PointCloudIds**

used by: elements **PartType/DefinitionInternal** **AssemblyType/DefinitionInternal**

attributes

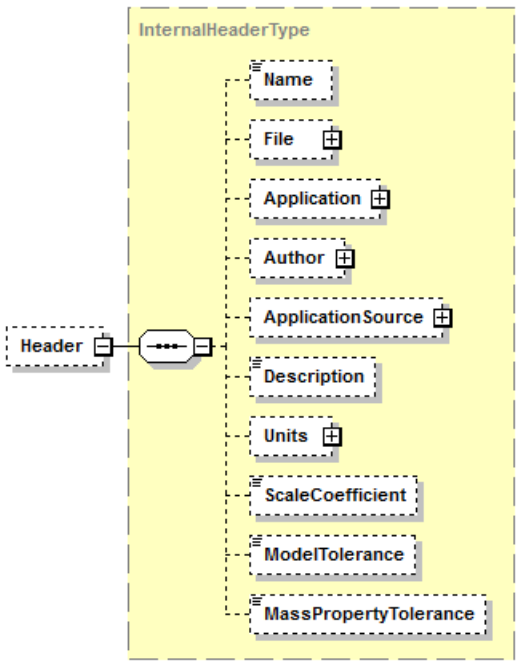
Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.
size	**DoublePositiveType**				documentation

	originMassProperty **PointSimpleType** required	The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system). documentation The required originMassProperty attribute is the origin point for calculating of objects mass properties contained in this block.
annotation	documentation The InternalPartAssemblyType is the type for internal parts and assemblies. It contains a set of entities, defining a feature (body).	

attribute **InternalPartAssemblyType/@originMassProperty**

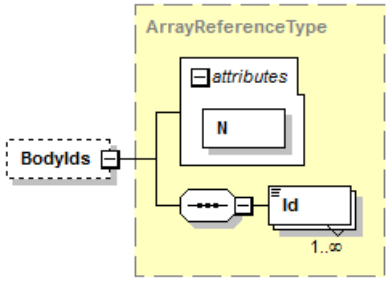
type	**PointSimpleType**
properties	use required
facets	Kind Value Annotation length 3
annotation	documentation The required originMassProperty attribute is the origin point for calculating of objects mass properties contained in this block.

element **InternalPartAssemblyType/Header**

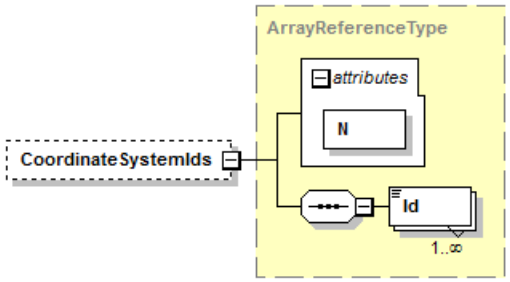
diagram	
type	InternalHeaderType
properties	minOcc 0 maxOcc 1 content complex
children	Name File Application Author ApplicationSource Description Units ScaleCoefficient ModelTolerance

	MassPropertyTolerance
annotation	documentation The optional Header element is a description of the provenance of the file.

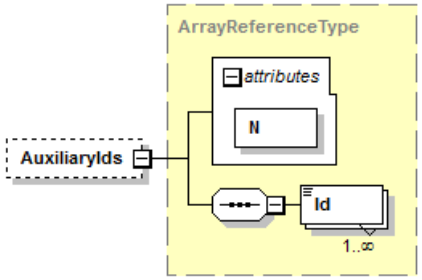
element **InternalPartAssemblyType/BodyIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional BodyIds element is an array of body identifiers present in this block.

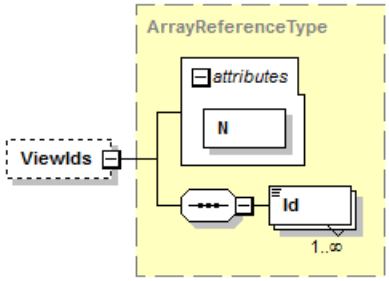
element **InternalPartAssemblyType/CoordinateSystemIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional CoordinateSystemIds element is an array of coordinate system identifiers present in this block.

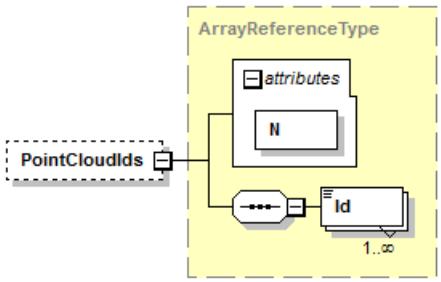
element **InternalPartAssemblyType/AuxiliaryIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional AuxiliaryIds element is an array of auxiliary object identifiers present in this block.

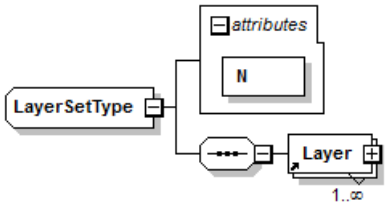
element **InternalPartAssemblyType/ViewIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional ViewIds element is an array of view object identifiers present in this block.

element **InternalPartAssemblyType/PointCloudIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional PointCloudIds element is an array of point cloud identifiers present in this block.

complexType **LayerSetType**

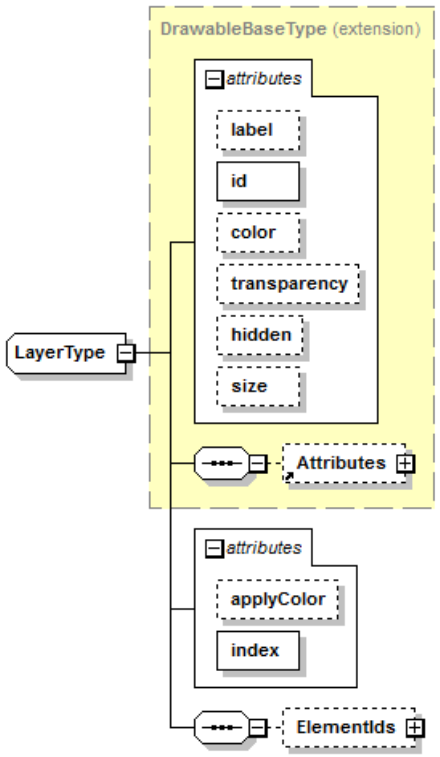
diagram	
children	**Layer**
used by	element **LayerSet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many objects are present in this set.

attribute **LayerSetType/@N**

type	**NaturalType**
properties	use required
facets	Kind: minInclusive; Value: 1; Annotation:
annotation	documentation The required N attribute shows how many objects are present in this set.

complexType **LayerType**

diagram



type	extension of **DrawableBaseType**
properties	base DrawableBaseType
children	**Attributes** **ElementIds**
used by	element **Layer**

attributes

Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
id	**QIFIdType**	required			documentation The required id attribute is the unique model entity identifier.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.
size	**DoublePositiveType**				documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric

					parameters (e.g. coordinate system).
	applyColor	**xs:boolean**		0	documentation The optional applyColor attribute shows if the layer color supersedes colors of the model entities associated with this layer. The default value is FALSE.
	index	**xs:unsignedInt**	required		documentation The required index attribute defines the layer index.
annotation	documentation The LayerType is the type which represents one of the model "grouping" objects. NOTE1: The user can specify arbitrary number of layers for one model. NOTE2: Any model entity can be included in an arbitrary number of layers.				

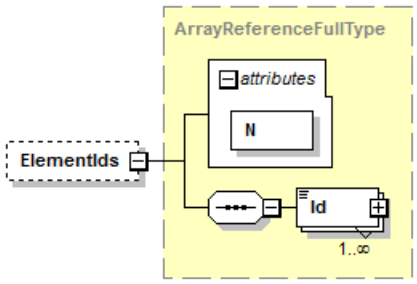
attribute **LayerType/@applyColor**

type	**xs:boolean**
properties	default 0
annotation	documentation The optional applyColor attribute shows if the layer color supersedes colors of the model entities associated with this layer. The default value is FALSE.

attribute **LayerType/@index**

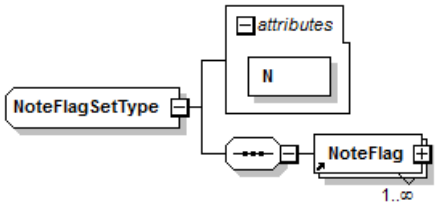
type	**xs:unsignedInt**
properties	use required
annotation	documentation The required index attribute defines the layer index.

element **LayerType/ElementIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N Type: **NaturalType** Use: required Default: Fixed: Annotation: documentation The required N attribute shows how many Id elements are present in this

	array.
annotation	documentation The optional ElementIds element is an array of entity identifiers present in this this layer.

complexType **NoteFlagSetType**

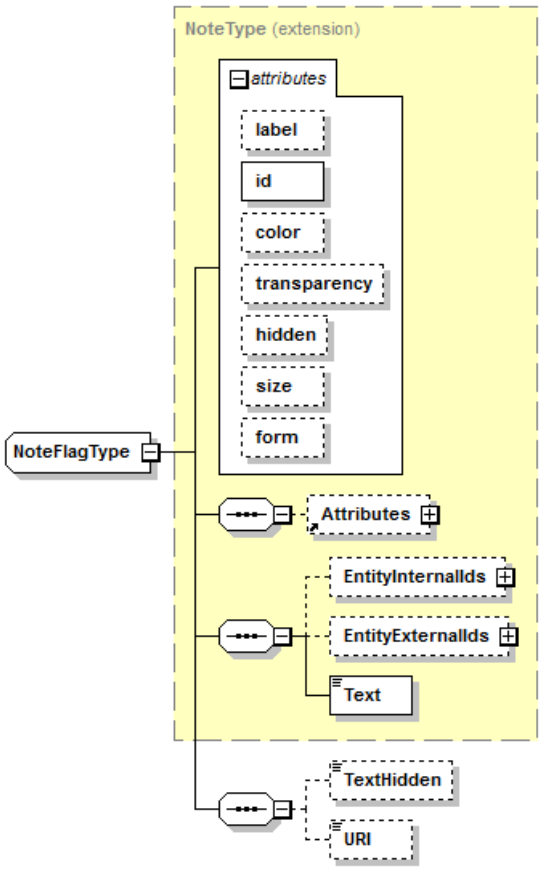
diagram	
children	**NoteFlag**
used by	element **NoteFlagSet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many objects are present in this set.

attribute **NoteFlagSetType/@N**

type	**NaturalType**
properties	use required
facets	Kind: minInclusive; Value: 1; Annotation:
annotation	documentation The required N attribute shows how many objects are present in this set.

complexType **NoteFlagType**

diagram



type: extension of **NoteType**

properties: base NoteType

children: **Attributes** **EntityInternalIds** **EntityExternalIds** **Text** **TextHidden** **URI**

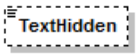
used by: element **NoteFlag**

attributes

Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
id	**QIFIdType**	required			documentation The required id attribute is the unique model entity identifier.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.
size	**DoublePositiveType**				documentation The optional size attribute defines a recommended size for

	form	**NoteFormEnumType**	3D	visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system). documentation The form attribute specifies how the Note is represented.

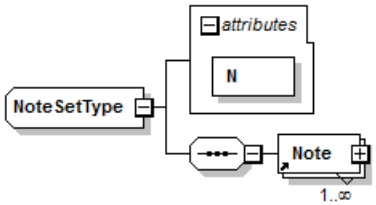
element **NoteFlagType/TextHidden**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional TextHidden element is the hidden text of the flag note.

element **NoteFlagType/URI**

diagram	
type	**xs:anyURI**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional URI element is a Uniform Resource Identifier for the information, which may be, for example, a file or a web site.

complexType **NoteSetType**

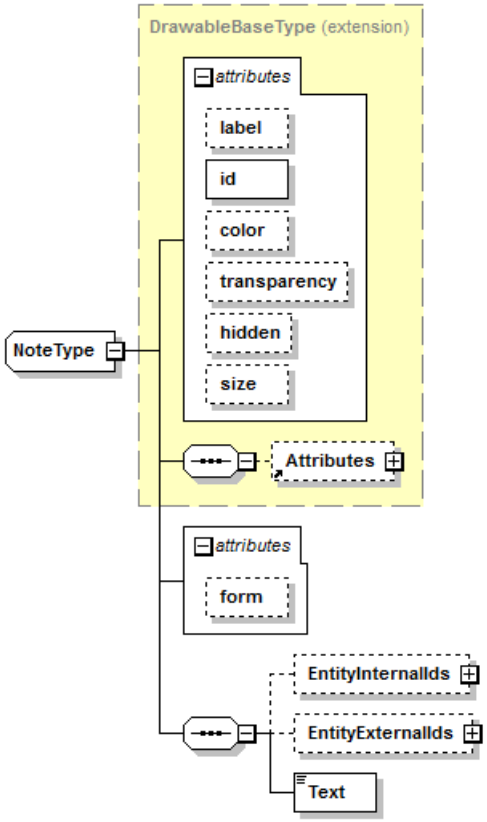
diagram	
children	**Note**
used by	element **NoteSet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation — The required N attribute shows how many objects are present in this set.

attribute **NoteSetType/@N**

type	**NaturalType**
properties	use required
facets	Kind Value Annotation

	minInclusive 1
annotation	documentation The required N attribute shows how many objects are present in this set.

complexType **NoteType**

diagram	
type	extension of **DrawableBaseType**
properties	base DrawableBaseType
children	**Attributes** **EntityInternalIds** **EntityExternalIds** **Text**
used by	element **Note** complexType **NoteFlagType**

attributes

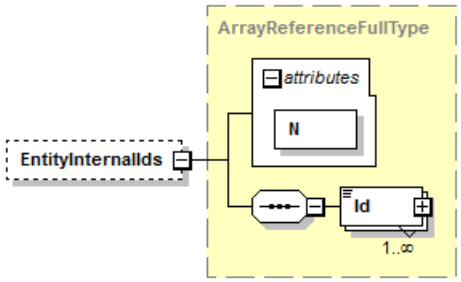
Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
id	**QIFIdType**	required			documentation The required id attribute is the unique model entity identifier.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a

				model entity in the graphical window.
	size	**DoublePositiveType**		documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system).
	form	**NoteFormEnumType**	3D	documentation The form attribute specifies how the Note is represented.

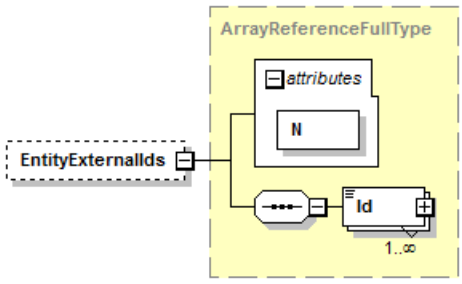
attribute **NoteType/@form**

type	**NoteFormEnumType**
properties	default 3D
facets	Kind Value Annotation enumeration 3D enumeration SCREEN
annotation	documentation The form attribute specifies how the Note is represented.

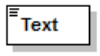
element **NoteType/EntityInternalIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name Type Use Default Fixed Annotation N **NaturalType** required documentation The required N attribute shows how many Id elements are present in this array.
annotation	documentation The EntityInternalIds element is a list of the QIF ids of CAD entities associated with this note.

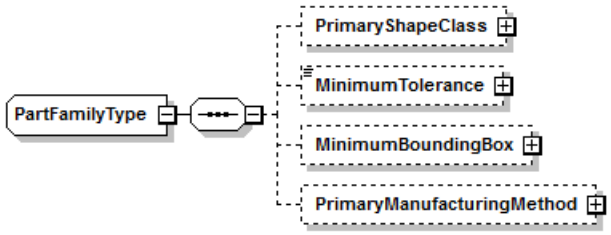
element **NoteType/EntityExternalIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The EntityExternalIds element is a list of the QIF ids of instances of EntityExternalType associated with this note.

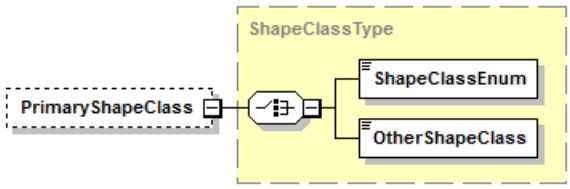
element **NoteType/Text**

diagram	
type	**xs:string**
properties	content simple
used by	complexType **TextsType**
annotation	documentation The Text element is the text of the Note.

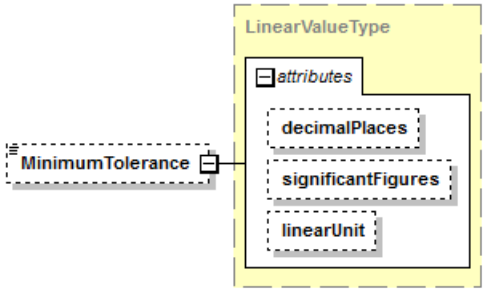
complexType **PartFamilyType**

diagram	
children	**PrimaryShapeClass** **MinimumTolerance** **MinimumBoundingBox** **PrimaryManufacturingMethod**
used by	element **PartType/PartFamily**
annotation	documentation The PartFamilyType provides information useful for grouping parts that have similar process flows and requirements. A part's PartFamilyType can be described by it's primary shape classification, primary manufacturing method, precision needs, and/or size ranges.

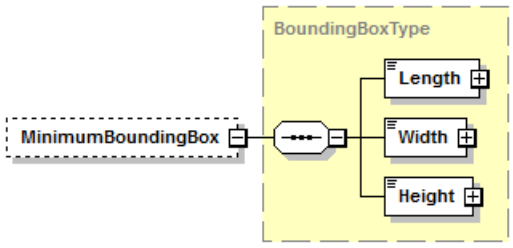
element **PartFamilyType/PrimaryShapeClass**

diagram	
type	**ShapeClassType**
properties	minOcc 0 maxOcc 1 content complex
children	**ShapeClassEnum OtherShapeClass**
annotation	documentation The optional PrimaryShapeClass element provides the primary shape classification of a part.

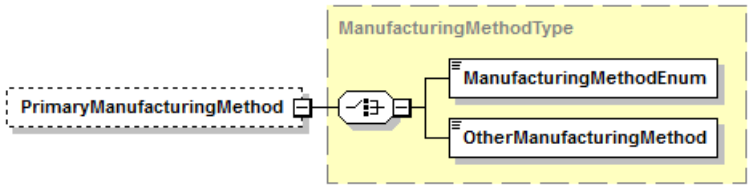
element **PartFamilyType/MinimumTolerance**

diagram	
type	**LinearValueType**
properties	minOcc 0 maxOcc 1 content complex
attributes	Name / Type / Use / Default / Fixed / Annotation decimalPlaces — **xs:nonNegativeInteger** — documentation: See documentation of SpecifiedDecimalType. significantFigures — **xs:nonNegativeInteger** — documentation: See documentation of SpecifiedDecimalType. linearUnit — **xs:token** — documentation: The optional linearUnit attribute defines the UnitName for the LinearValueType.
annotation	documentation The optional MinimumTolerance element identifies the smallest linear tolerance that a part possesses that will need to be verified.

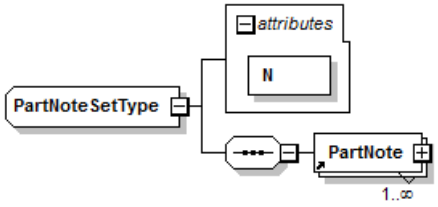
element **PartFamilyType/MinimumBoundingBox**

diagram	
type	**BoundingBoxType**
properties	minOcc 0 maxOcc 1 content complex
children	**Length Width Height**
annotation	documentation The optional MinimumBoundingBox element provides the smallest bounding box into which the part will fit.

element **PartFamilyType/PrimaryManufacturingMethod**

diagram	
type	**ManufacturingMethodType**
properties	minOcc 0 maxOcc 1 content complex
children	**ManufacturingMethodEnum OtherManufacturingMethod**
annotation	documentation The optional PrimaryManufacturingMethod element identifies the manufacturing method most likely used to produce the primary shape of a part.

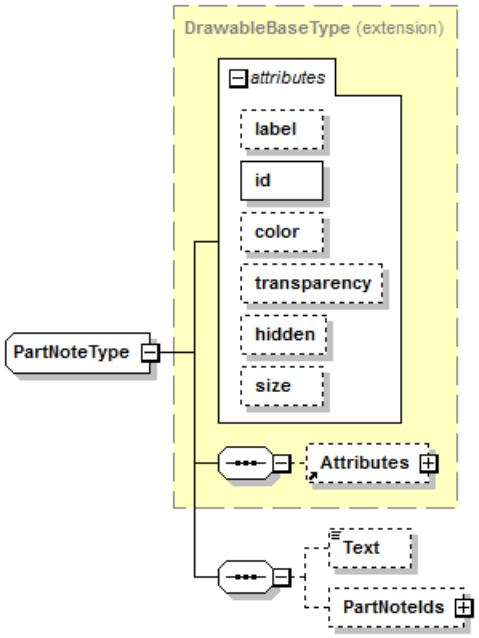
complexType **PartNoteSetType**

diagram	
children	**PartNote**
used by	element **PartNoteSet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many objects are present in this set.

attribute **PartNoteSetType/@N**

type	**NaturalType**
properties	use required
facets	Kind Value Annotation minInclusive 1
annotation	documentation The required N attribute shows how many objects are present in this set.

complexType **PartNoteType**

diagram	
type	extension of **DrawableBaseType**
properties	base DrawableBaseType
children	**Attributes** **Text** **PartNoteIds**
used by	element **PartNote**

attributes

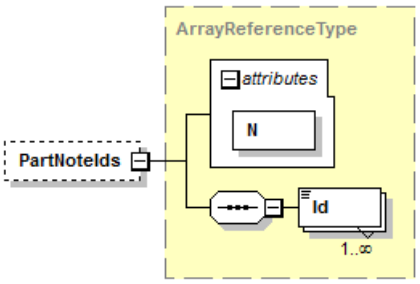
Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
id	**QIFIdType**	required			documentation The required id attribute is the unique model entity identifier.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a model entity in the graphical

	size **DoublePositiveType**	window. documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system).

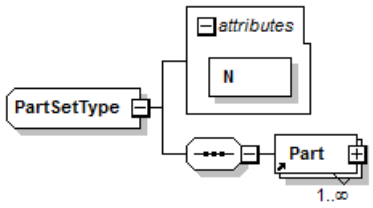
element **PartNoteType/Text**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
used by	complexType **TextsType**
annotation	documentation The optional Text element is the text of the PartNote.

element **PartNoteType/PartNoteIds**

diagram	
type	**ArrayReferenceType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name N; Type **NaturalType**; Use required; Default; Fixed; Annotation documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional PartNoteIds element is an array of identifiers of nested part notes.

complexType **PartSetType**

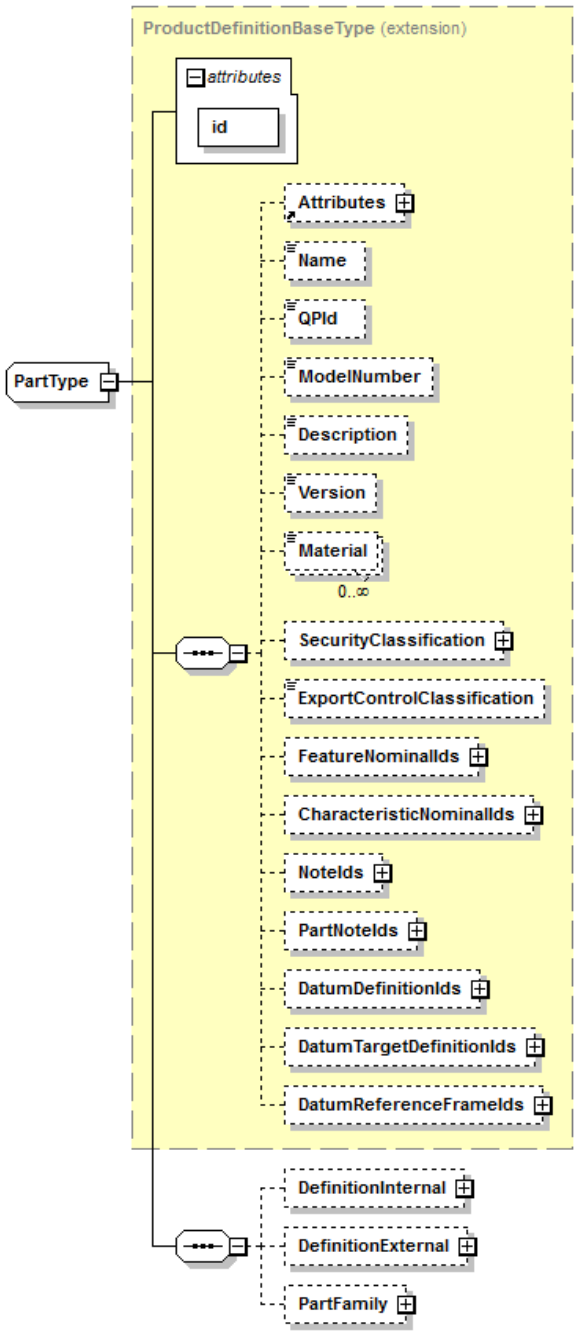
diagram	
children	**Part**
used by	element **PartSet**
attributes	Name: N; Type: **NaturalType**; Use: required; Default; Fixed; Annotation: documentation: The required N attribute is the number of objects in this set.
annotation	documentation The PartSetType represents a container for storing all parts present in the CAD scene.

attribute **PartSetType/@N**

type	**NaturalType**
properties	use required
facets	Kind: minInclusive; Value: 1; Annotation
annotation	documentation The required N attribute is the number of objects in this set.

complexType **PartType**

diagram



type	extension of **ProductDefinitionBaseType**
properties	base ProductDefinitionBaseType
children	**Attributes** **Name** **QPId** **ModelNumber** **Description** **Version** **Material** **SecurityClassification** **ExportControlClassification** **FeatureNominalIds** **CharacteristicNominalIds** **NoteIds** **PartNoteIds** **DatumDefinitionIds** **DatumTargetDefinitionIds** **DatumReferenceFrameIds** **DefinitionInternal** **DefinitionExternal** **PartFamily**
used by	element **Part**

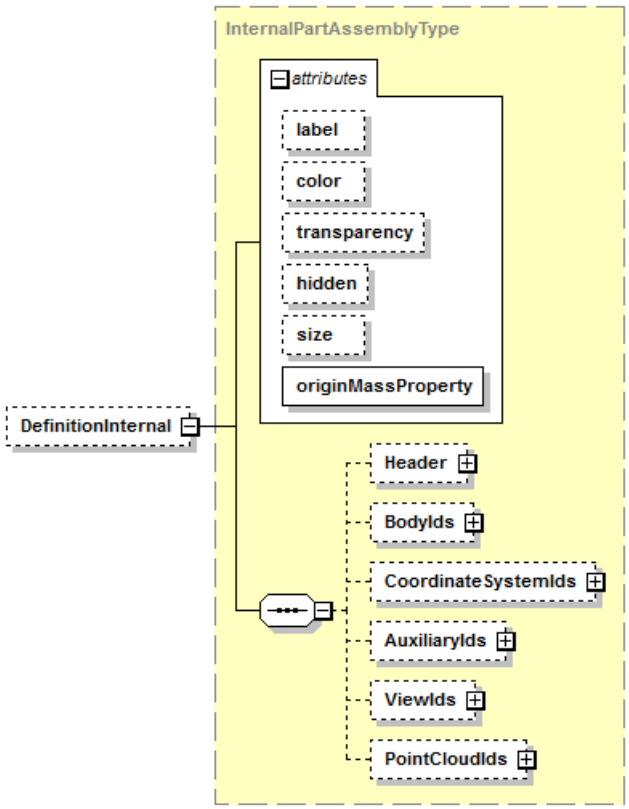
attributes

Name	Type	Use	Default	Fixed	Annotation
id	**QIFIdType**	required			documentation The required id attribute is the QIF id

	of the product definition, used for referencing.
annotation	documentation The PartType is the part type. It contains a set of entities, defining a feature (body), which can be instantiated multiple times in the CAD scene. Use of parts simplifies maintenance of identical features and reduces the total amount of scene data.

element **PartType/DefinitionInternal**

diagram



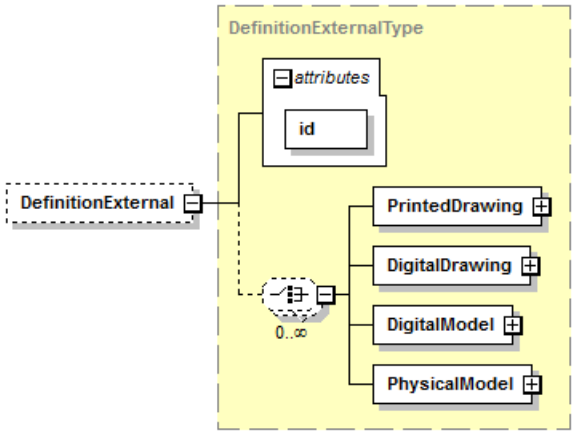
type	**InternalPartAssemblyType**
properties	minOcc 0 maxOcc 1 content complex
children	**Header BodyIds CoordinateSystemIds AuxiliaryIds ViewIds PointCloudIds**

attributes

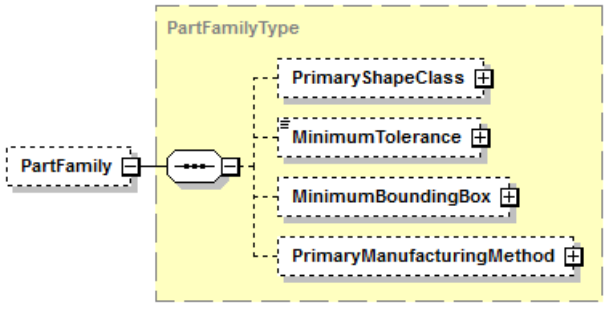
Name	Type	Use	Default	Fixed	Annotation
label	**xs:string**				documentation The optional label attribute is the model entity "nameplate". Normally it can be seen at the entity item in the project tree.
color	**ColorType**				documentation The optional color attribute defines the RGB color property of a model entity.
transparency	**xs:double**		0.0		documentation The optional transparency attribute defines the transparency property of a model entity.
hidden	**xs:boolean**		0		documentation The optional hidden attribute defines the visibility property of a model entity in the graphical window.

	size	**DoublePositiveType**		documentation The optional size attribute defines a recommended size for visualization of an infinite drawable element such as plane, cylinder, axis, etc., or objects without explicit geometric parameters (e.g. coordinate system).
	originMassProperty	**PointSimpleType**	required	documentation The required originMassProperty attribute is the origin point for calculating of objects mass properties contained in this block.
annotation	documentation The optional DefinitionInternal element is a part definition in QIF format.			

element **PartType/DefinitionExternal**

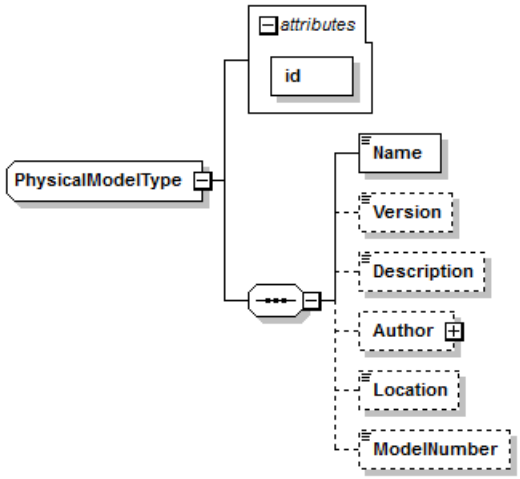
diagram	
type	**DefinitionExternalType**
properties	minOcc 0 maxOcc 1 content complex
children	**PrintedDrawing** **DigitalDrawing** **DigitalModel** **PhysicalModel**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation The required id attribute is the QIF id of the product geometry definitions, used for referencing.
annotation	documentation The optional DefinitionExternal element is a part definition in non-QIF format.

element **PartType/PartFamily**

diagram	

type	**PartFamilyType**
properties	minOcc 0 maxOcc 1 content complex
children	**PrimaryShapeClass MinimumTolerance MinimumBoundingBox PrimaryManufacturingMethod**
annotation	documentation The optional PartFamily element gives information about the part that is useful for inspection planning.

complexType **PhysicalModelType**

diagram	
children	**Name Version Description Author Location ModelNumber**
used by	element **DefinitionExternalType/PhysicalModel**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the physical model.
annotation	documentation The PhysicalModelType defines an actual, hands-on, physical, 3D model or prototype of a product that is used to communicate features, datum reference frames, characteristics, or other information. Examples of physical models include actual product instances, product instances made from other materials, and 3D printing and stereolithography models. If the product is an assembly, a physical part model is not necessarily an assembly.

attribute **PhysicalModelType/@id**

type	**QIFIdType**
properties	use required
annotation	documentation The required id attribute is the QIF id of the physical model.

element **PhysicalModelType/Name**

diagram	
type	**xs:string**
properties	content simple

annotation	documentation The Name element is the name of the model.

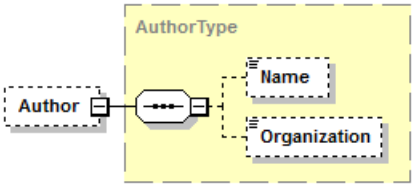
element **PhysicalModelType/Version**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Version element is the version of the model associated with product being inspected.

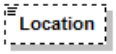
element **PhysicalModelType/Description**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Description element is a description of the model.

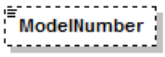
element **PhysicalModelType/Author**

diagram	
type	**AuthorType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization**
annotation	documentation The optional Author element is the author who created this model.

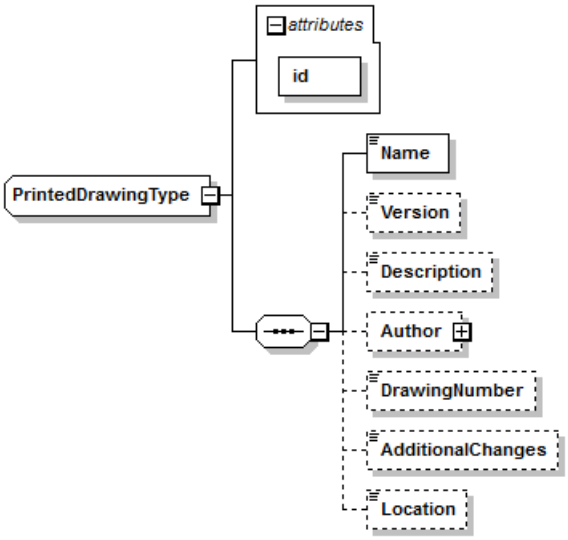
element **PhysicalModelType/Location**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Location element is a description of the physical location of the printed drawing.

element **PhysicalModelType/ModelNumber**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional ModelNumber element is the drawing number of the printed drawing associated with product being inspected. This may be used for the Drawing Number field of an AS9102A form.

complexType **PrintedDrawingType**

diagram	
children	**Name Version Description Author DrawingNumber AdditionalChanges Location**
used by	element **DefinitionExternalType/PrintedDrawing**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the printed drawing.
annotation	documentation The PrintedDrawingType defines information about a printed drawing of a product. This may be on paper, mylar, or some other physical media.

attribute **PrintedDrawingType/@id**

type	**QIFIdType**
properties	use required
annotation	documentation The required id attribute is the QIF id of the printed drawing.

element **PrintedDrawingType/Name**

diagram	
type	**xs:string**
properties	content simple
annotation	documentation The Name element is the name of the model.

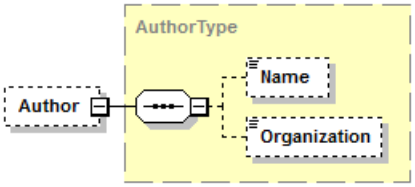
element **PrintedDrawingType/Version**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Version element is the version of the model associated with product being inspected. The information recorded here may be used for the DrawingRevisionLevel field of an AS9102A form.

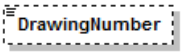
element **PrintedDrawingType/Description**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Description element is a description of the model.

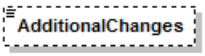
element **PrintedDrawingType/Author**

diagram	
type	**AuthorType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name Organization**
annotation	documentation The optional Author element is the author who created this drawing.

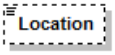
element **PrintedDrawingType/DrawingNumber**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional DrawingNumber element is the drawing number of the printed drawing associated with product being inspected. This may be used for the Drawing Number field of an AS9102A form.

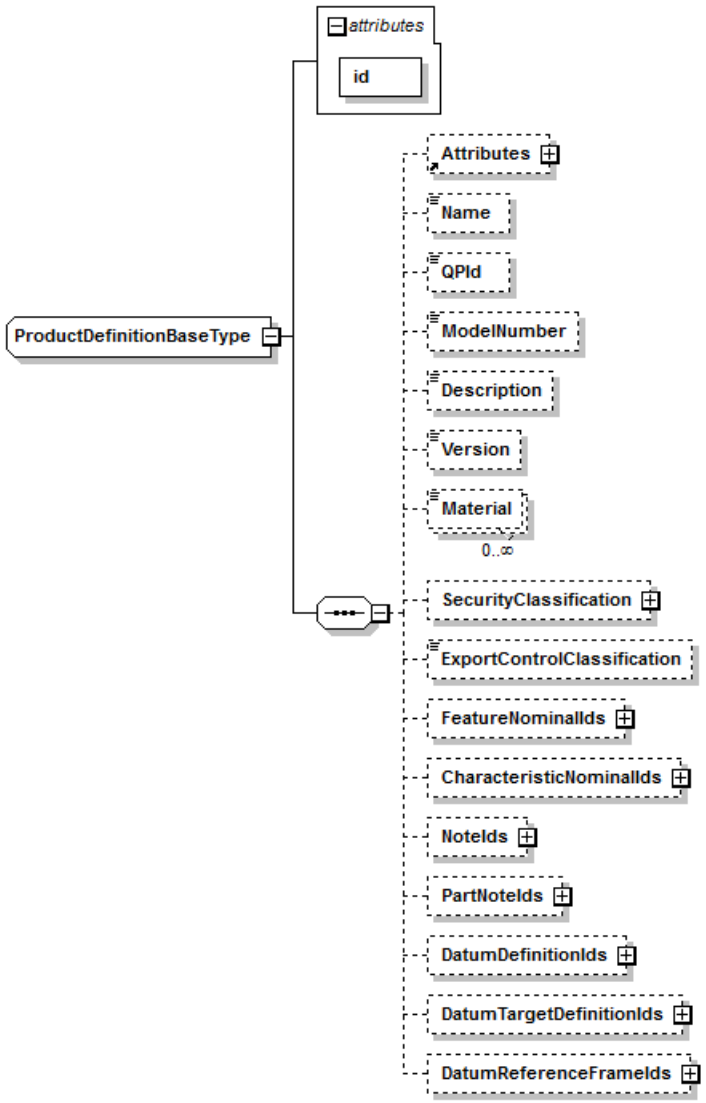
element **PrintedDrawingType/AdditionalChanges**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional AdditionalChanges element is a description or references to descriptions of any additional changes to the drawing beyond what is included at the Version. If there are no additional changes to the drawing, this element should be omitted. The information recorded here may be used for the Additional Changes field of an AS9102A form.

element **PrintedDrawingType/Location**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Location element is a description of the physical location of the printed drawing.

complexType **ProductDefinitionBaseType**

diagram	
properties	abstract true
children	**Attributes** **Name** **QPId** **ModelNumber** **Description** **Version** **Material** **SecurityClassification** **ExportControlClassification** **FeatureNominalIds** **CharacteristicNominalIds** **NoteIds** **PartNoteIds** **DatumDefinitionIds** **DatumTargetDefinitionIds** **DatumReferenceFrameIds**
used by	complexTypes **AssemblyType** **PartType**
attributes	Name: id; Type: **QIFIdType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required id attribute is the QIF id of the product definition, used for referencing.
annotation	documentation: The ProductDefinitionBaseType is the abstract base type that defines information about a product definition that was inspected or will be inspected.

attribute **ProductDefinitionBaseType/@id**

type	**QIFIdType**

properties	use required
annotation	documentation The required id attribute is the QIF id of the product definition, used for referencing.

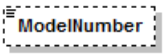
element **ProductDefinitionBaseType/Name**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Name element is a name given to a product definition. There may be several products with the same name but different model numbers.

element **ProductDefinitionBaseType/QPId**

diagram	
type	**QPIdType**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional QPId element is a persistent identifier for the product definition. If used, it should be generated using a widely accepted UUID generator.

element **ProductDefinitionBaseType/ModelNumber**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional ModelNumber element is the identifier of a very specific type of product definition. There may be many product instances with the same ModelNumber, but they should all be identical (or nearly so).

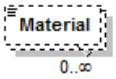
element **ProductDefinitionBaseType/Description**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Description element is a natural language description of the product definition.

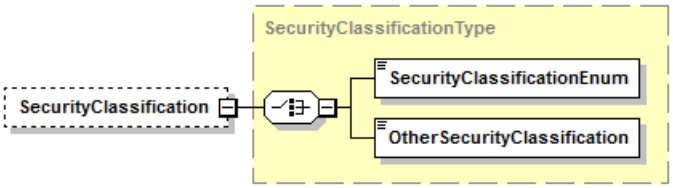
element **ProductDefinitionBaseType/Version**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional Version element is the version of the product definition.

element **ProductDefinitionBaseType/Material**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc unbounded content simple
annotation	documentation Each optional Material element is a material from which the actual product is fabricated. More than one Material element may be used since some products are made from more than one material.

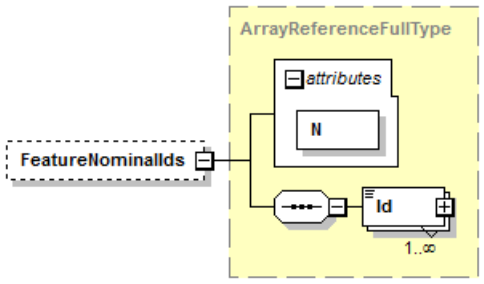
element **ProductDefinitionBaseType/SecurityClassification**

diagram	
type	**SecurityClassificationType**
properties	minOcc 0 maxOcc 1 content complex
children	**SecurityClassificationEnum OtherSecurityClassification**
annotation	documentation The optional SecurityClassification element describes the security classification of the product definition.

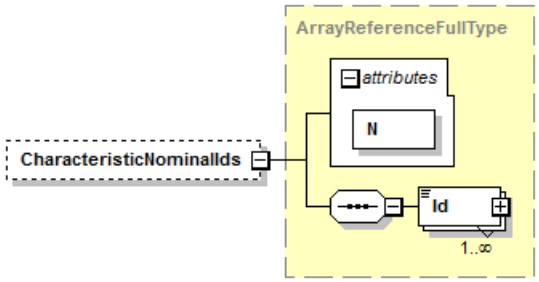
element **ProductDefinitionBaseType/ExportControlClassification**

diagram	
type	**xs:string**
properties	minOcc 0 maxOcc 1 content simple
annotation	documentation The optional ExportControlClassification element describes the export control classification of the product definition.

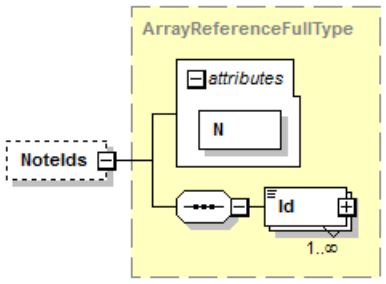
element **ProductDefinitionBaseType/FeatureNominalIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional FeatureNominalIds element contains any feature nominals for the product definition.

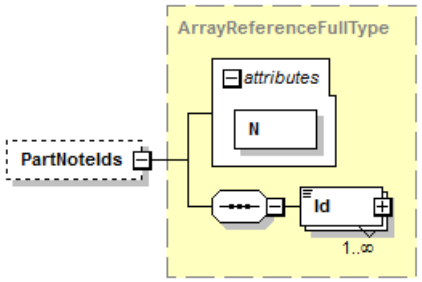
element **ProductDefinitionBaseType/CharacteristicNominalIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional CharacteristicNominalIds element contains any characteristic nominals for the product definition.

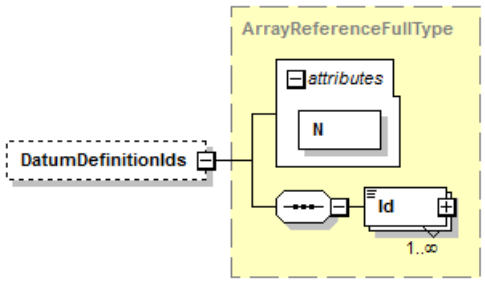
element **ProductDefinitionBaseType/NoteIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional NoteIds element contains any notes for the product definition.

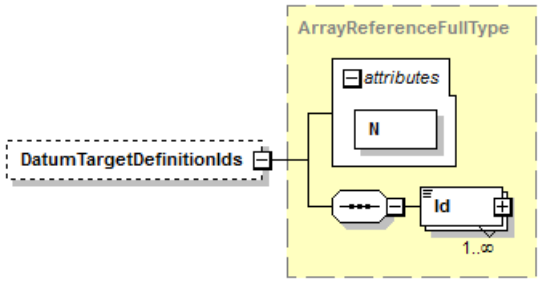
element **ProductDefinitionBaseType/PartNoteIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional PartNoteIds element contains any part notes for the product definition.

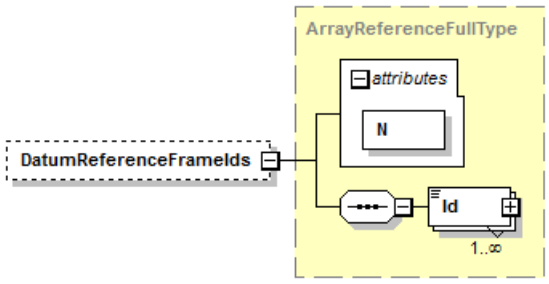
element **ProductDefinitionBaseType/DatumDefinitionIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional DatumDefinitionIds element contains any datum definitions for the product definition.

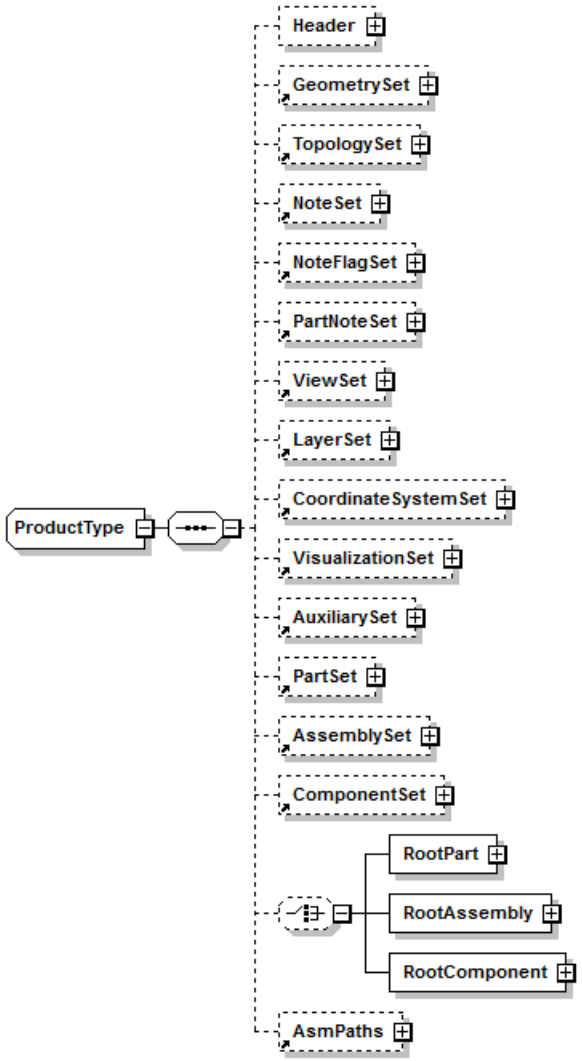
element **ProductDefinitionBaseType/DatumTargetDefinitionIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional DatumTargetDefinitionIds element contains any datum target definitions for the product definition.

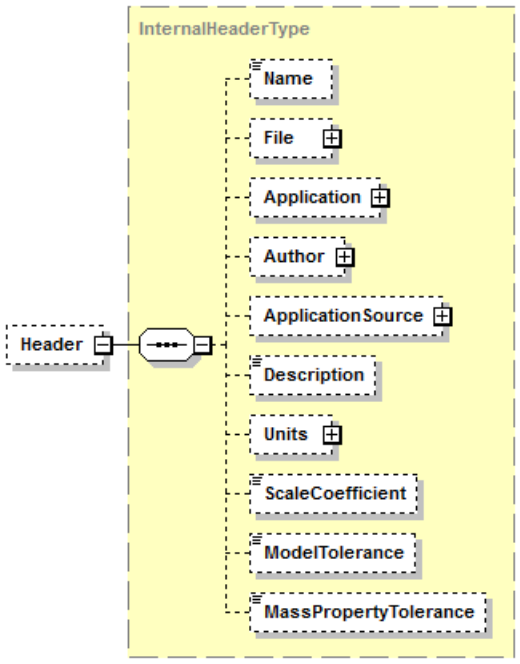
element **ProductDefinitionBaseType/DatumReferenceFrameIds**

diagram	
type	**ArrayReferenceFullType**
properties	minOcc 0 maxOcc 1 content complex
children	**Id**
attributes	Name: N; Type: **NaturalType**; Use: required; Default: ; Fixed: ; Annotation: documentation: The required N attribute shows how many Id elements are present in this array.
annotation	documentation The optional DatumReferenceFrameIds element contains any datum reference frames for the product definition.

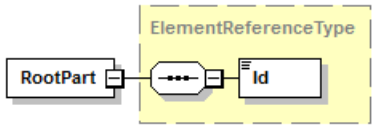
complexType **ProductType**

diagram	
children	Header **GeometrySet TopologySet NoteSet NoteFlagSet PartNoteSet ViewSet LayerSet CoordinateSystemSet VisualizationSet AuxiliarySet PartSet AssemblySet ComponentSet** RootPart RootAssembly RootComponent **AsmPaths**
used by	element **Product**
annotation	documentation The ProductType defines parts and assemblies.

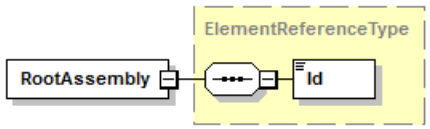
element **ProductType/Header**

diagram	
type	**InternalHeaderType**
properties	minOcc 0 maxOcc 1 content complex
children	**Name File Application Author ApplicationSource Description Units ScaleCoefficient ModelTolerance MassPropertyTolerance**
annotation	documentation The Header element gives information about the provenance of the Product data set.

element **ProductType/RootPart**

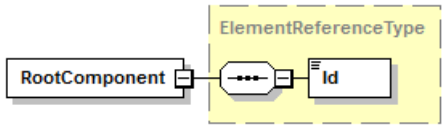
diagram	
type	**ElementReferenceType**
properties	content complex
children	**Id**
annotation	documentation The RootPart element is a root of CAD scene.

element **ProductType/RootAssembly**

diagram	
type	**ElementReferenceType**

properties	content complex
children	**Id**
annotation	documentation The RootAssembly element is a root of CAD scene.

element **ProductType/RootComponent**

diagram	
type	**ElementReferenceType**
properties	content complex
children	**Id**
annotation	documentation The RootComponent element is a root of CAD scene.

simpleType **GDTEnumType**

type	restriction of **xs:string**
properties	base xs:string
used by	element **DigitalModelType/GDT**
facets	Kind Value Annotation enumeration UNKNOWN enumeration HUMANREAD enumeration MACHINEREAD enumeration ABSENT
annotation	documentation The GDTEnumType enumerates values that describe the geometric dimensioning and tolerancing information in model.

simpleType **NoteFormEnumType**

type	restriction of **xs:string**
properties	base xs:string
used by	attribute **NoteType/@form**
facets	Kind Value Annotation enumeration 3D enumeration SCREEN
annotation	documentation The NoteFormType enumerates values that describe the form of the Note and can be the following values: `3D` - Note defined on the 3D annotation plane; `SCREEN` - Note defined as flat to screen;

simpleType **TopologyEnumType**

type	restriction of **xs:string**
properties	base xs:string
used by	element **DigitalModelType/Topology**
facets	Kind Value Annotation enumeration UNKNOWN enumeration PRESENT enumeration ABSENT
annotation	documentation The TopologyEnumType enumerates values that describe the topology information in model.

~~ end of QIFProduct.xsd data dictionary ~~

Bibliography

[1] SAE AS9102a (2004-01), *Aerospace First Article Inspection Requirement*

[2] Walmsley, Priscilla., 2002. *Definitive XML Schema*. Prentice Hall, Upper Saddle River, NJ, USA.

