XML

GOM Inspection Exchange Format

Version 2.3

Last modified at: 10.05.16

GOM mbH
Mittelweg 7-8
D-38106 Braunschweig
Germany
Tel.: +49 (0) 531 390 29 0

E-Mail: info@gom.com
http://www.gom.com
Fax: +49 (0) 531 390 29 15

Contents

1 Introduction

The file format described by this document is intended to be used as exchange format internally and for third party software.

The format supports primitives, dimension, inspected elements, GD&T tolerances and datum systems. Also supported are meshes, deviations, point clouds, curves and sections.

Please note that this file format does not support all elements which are available in the GOM software.

Some elements may be exported only as primitive geometry and only if they are calculated. The data, which is being exported, contains nominal (e.g. tolerances) and actual measured data, essential 3D drawing information (e.g. position of an angle) and user data (e.g. name of an element). Again, not all properties (such as visualization color or label position) are stored in this file format.

Please note that meshes, point clouds, curves or sections may create huge file sizes, even though the point data is packed as binary data and stored as a Base64 string in the XML. All character strings inside the file format use the XML quoting and a UTF-8 encoding.

The documentation for XML can be found at http://www.xml.org .

To export data from the GOM software in this format, you have to choose the menu

"File → Export → Elements → Elements XML".

This file format was introduced with GOM software version V8.

Earlier XML versions that were introduced in V6 of the GOM software are still supported, but imported elements may differ from previous versions, because the used element types are no longer supported.

There was another XML based format already in v6.0.0 of the GOM software, but that file format was specialized for measurement plans. So these two file formats have nothing in common except using XML.

2 Binary Data Types

Data of some elements is stored in a binary format that is encoded in a base64 string. Binary data is always stored in big-endian format. Please note that these elements may create very big XML files on export. The data types that occur in a binary block are listed in the following two tables.

Table 1: Basic Data Types

Type	Description
UINT8	Unsigned 8-bit integer value
UINT32	Unsigned 32-bit integer value
F32	32-bit floating point number
F64	64-bit double precision floating point number

Table 2: Composite Data Types

Type	Description
VEC3D32	The VEC3D32 type defines X, Y, Z coordinate values. So a VEC3D32 is made up of three F32 base types.
VEC3D64	The VEC3D64 type defines X, Y, Z coordinate values. So a VEC3D64 is made up of three F64 base types.
RGBA	The RGBA type defines a color composed of red, green, blue and alpha components, of which each is a UINT8. The red, green, blue color values typically range from 0 to 255. The alpha value ranges from 0 to 255 where 255 indicates completely opaque.

3 General File Structure

The main structure of this format is

```
<?xml version="1.0" encoding="UTF-8"?>
<gom>
	<header>
		<version></version>
		<length_unit></length_unit>
		<angle_unit></angle_unit>
	</header>
	<nominal>
		…
	</nominal>
	<measured>
		…
	</measured>
</gom>
```

The processing instruction in the first line has to reflect the encoding of the file. If the encoding is wrong, some names and string values may be imported incorrectly. Exported files always have an UTF-8 encoding. The root tag is <gom>. There is nothing outside this section except the processing instruction in the first line. The section includes four sub-tags:

- `<header>` includes the version of the file and the used units for length and angle.
- `<nominal>` includes all data of the exported nominal elements. Each element has its own sub tag (such as `<point>`) and keeps its data inside that sub tag. Also the checks done on actual elements are listed here.
- `<measured>` includes all data of the exported actual elements. Each element has its own sub tag (same tags as for nominal) and keeps its data inside that sub tag.

Some general information:

- All normals and directions, which occur in the element data, are normalized, because otherwise these vectors may become numerically unstable for very short vectors.
- There are some XML comments inside the example files which give a hint if the XML tag name needs additional explanations and references where the chapters can be found.
- Every element in the XML file has to have a unique ID. Using these ID elements can reference other elements. Referencing can be done in different ways:
  - If the referenced element is in the XML file, a reference is given by the element ID. On import the reference is resolved and the elements are linked.
  - The reference can be given by an element name. On import a matching element with the given name is searched. If it exists, it is linked to the imported element.
  - In some cases both link methods can be used. If the given ID is not found in the file, then it will be tried to link the element using the given name.
  - Linking is done for nominal elements to link to the actual elements (and vice versa) and for GD&T elements that are linked to datum systems and the needed geometry.

4 Shared tags for all elements

Elements can be nominal or actual. Both can be inspected or not. If they are inspected, then they can have a reference to its linked actual element if they are nominal.

All types of elements share the following data:

```
<type id="id" name="name">
      <keywords>element keywords</keywords>
      <state>ok or uncomputed</state>
      <comment>comment</comment>
      <geometry>geometry description</geometry>
      <result></result>
</type>
```

If the element is nominal, additional tags are available:

```
<actual>id to actual element</actual>
<result>
      additional information for the element and values for the used
      checks
</result>
```

The attributes contain the following data:

- <id>: This is an ID, that has to be unique throughout the whole XML file. It is generated by the software automatically and is used to reference other elements in the XML. It can be a UUID or any other unique string value.
- <name>: This tag contains the name of the exported element.

The tags contain the following data:

- <comment>: This tag contains the comment of the exported element. This is a string which the user can edit in the GOM software.
- <state>: Elements can be either computed or uncomputed. Setting this tag to "**ok**" or "**uncomputed**" reflects the state of the element in the GOM software when it was exported. Default value is "**uncomputed**" for nominal elements, actual elements are always computed.
- <geometry>: Depending on the element type this tag holds a set of other tags that are describing the geometry data of the element.
- <keywords>: This tag holds a number of sub tags describing the keywords of this element. These keywords can be added by the GOM software automatically (e.g. on import) and contains additional information that is only valid for this element. The definition of these keywords is described in chapter 4.1. This tag is only used if the element is nominal.
- <actual>: Gives the ID to the actual element that is linked with this nominal element. This tag is only valid if the element is nominal or if it is an inspection. If the element has no actual element, this tag is left out.
- <result>: This tag contains the tolerance data tag with floating point numbers for *tolerance limits*, *measured deviation and measured values*. For further details see chapter 4.1.

4.1 Element Keywords

Element keywords which consist of an identifier, a description and a content are arbitrary. Each of these values can be changed by the user in the GOM software. These keywords store information for the elements they are defined on.

These keywords are defined as a sub tag for the element and defined as follows:

```
<point …>
      <keywords>
            <keyword
                  desc="description"
                  name="identifier"
                  editable="0" or "1"
```

```
                    stage="0" or "1">
                    "content"
              </keyword>
        </keywords>
  </point>
```

- *name*: This attribute holds a unique name (e.g. “measuring_point“). It is used to reference the keyword within the GOM software, e.g. in snapshots.
- *desc:* A string that describes the content (e.g. “point to measure“).
- *editable:* Defines the keyword as editable in the software.
- *stage:* Specifies that the keyword can hold separate values for every stage.
- *content*: A string that holds information about this project or session (e.g. “J. Doe“).

For every keyword defined on the element a separate <keyword> tag is generated.

4.2 Results

Nominal elements have a result tag which hold all information about the used (and unused) tolerances, the measured values and the calculated deviations.

```
<type id="id" name="name">
      …
      <result>
            <tolerance category checked="0" or "1">
                  <tolerance lower_limit="floating point number"
                        upper_limit="floating point number" />
                  <nominal_scalar value="floating point number"/>
                  <deviation value="floating point number" or invalid/>
                  <measured value="floating point number" or invalid/>
                  or
                  <measured
                        x="floating point number"
                        y="floating point number"
                        z="floating point number">
                  </measured>
            </tolerance category>
            …
      </result>
</type>
```

The tags contain the following data:

- <*tolerance category*>: This tag contains the type of the tolerance. The usable categories are depending on the element type. The following categories are supported:

tolerance category	**description**
x	• tolerance in x direction
y	• tolerance in y direction
z	• tolerance in z direction
all	• position tolerance
diameter	• diameter tolerance

`width`	• width tolerance
`normal`	• tolerance in normal direction
`trim`	• tolerance in trimming direction
`inplane`	• inplane tolerance
`length`	• length tolerance
`angle`	• angle tolerance
`inner_diameter`	• inner diameter tolerance
`tol_size`	• tolerance zone size for GD&T elements

`direction1`
`direction2`
`direction3`
	• tolerance in appropriate direction for GD&T position tolerance

`linear_size`	• linear size tolerance

- `<tolerance category> - <tolerance>`: This tag gives the upper and lower limits of the tolerance in its attributes `lower_limit` and `upper_limit`. This tag is only available if the tolerance is used.
- `<tolerance category> - <deviation>`: This tag gives the deviation value to this tolerance category. If the value is not calculated, this tag is set to the string value "**invalid**".
- `<tolerance category> - <measured>`: This tag gives the measured value to this tolerance category. Depending on the tolerance category, the value could be a vector that gives the measured values in its `x,y,z` attributes or a single floating point number. If the value is not calculated, this tag is set to the string-value "**invalid**".

5 Geometry Primitives

5.1 Point

A point is written in the form:

```
<point …>
      please have a look at the beginning of chapter 4 for shared tags
      <geometry>
            <pos
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </pos>
      </geometry>
      <result></result>
</point>
```

Specific tags:

- `<geometry> - <pos>`: This tag contains the x, y, z coordinates of the exported 3D point.
- `<geometry> - <normal>`: This tag contains the optional normal vector of the point. If the normal tag is available, the point is a surface point.
- `<geometry> - <trim>`: This tag contains the optional trimming vector of the point. If this tag is available, the point is an edge point.

5.2 Line

A line is written in the form:

```
<line …>
      please have a look at the beginning of chapter 4 for shared tags
      <geometry>
            <pos1
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </pos1>
            <pos2
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </pos2>
            <dir
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </dir>
      </geometry>
</line>
```

Specific tags:

- `<geometry> - <pos1>`: This tag contains the x, y, z coordinates of the first point (start) of the line.
- `<geometry> - <pos2>`: This tag contains the x, y, z coordinates of the second point (end) of the line.
- `<geometry> - <dir>`: This tag contains the normalized vector of the direction of this line.

5.3 Plane

A plane is written in the form:

```
<plane …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <pos
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </pos>
        <normal
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </normal>
        <distance>floating point number</distance>
        <length>floating point number</length>
        <width>floating point number</width>
        <orientation>floating point number</orientation>
    </geometry>
</plane>
```

Specific tags:

- `<geometry> - <pos>`: This tag contains the x, y, z coordinates of the center point of the visualization of the exported plane.
- `<geometry> - <normal>`: This tag contains the normalized vector of the normal of this plane.
- `<geometry> - <distance>`: This tag contains the distance of this plane.
- `<geometry> - <length>`: This tag contains the visualized length of this plane.
- `<geometry> - <width>`: This tag contains the visualized width of this plane.
- `<geometry> - <orientation>`: This tag contains the visualized orientation of this plane in degree.

5.4 Circle

A circle is written in the form:

```
<circle …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <pos
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </pos>
        <normal
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </normal>
        <radius>floating point number</radius>
    </geometry>
</circle>
```

Specific tags:

- `<geometry> - <pos>`: This tag contains the x, y, z coordinates of the center point of the exported circle.
- `<geometry> - <normal>`: This tag contains the normalized vector of the normal of this circle.
- `<geometry> - <radius>`: This tag contains the radius of this circle.

5.5 Slotted Hole

A slotted hole is written in the form:

```
<slotted_hole …>
```

please have a look at the beginning of chapter 4 for shared tags

```
<geometry>
      <pos
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </pos>
      <normal
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </normal>
      <dir
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </dir>
      <length>floating point number</length>
      <width>floating point number</width>
</geometry>
</slotted_hole>
```

Specific tags:

- `<geometry> - <pos>`: This tag contains the x, y, z coordinates of the center point of the exported slotted hole.
- `<geometry> - <normal>`: This tag contains the normalized vector of the normal of this slotted hole.
- `<geometry> - <dir>`: This tag contains the normalized vector of the direction of this slotted hole.
- `<geometry> - <length>`: This tag contains the length of this slotted hole.
- `<geometry> - <width>`: This tag contains the width of this slotted hole.

5.6 Rectangular Hole

A rectangular hole is written in the form:

```
<rectangular_hole …>
```

please have a look at the beginning of chapter 4 for shared tags

```
<geometry>
      <pos
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </pos>
      <normal
            x="floating point number"
```

```
                    y="floating point number"
                    z="floating point number">
                </normal>
                <dir
                    x="floating point number"
                    y="floating point number"
                    z="floating point number">
                </dir>
                <length>floating point number</length>
                <width>floating point number</width>
                <edge_radius>floating point number</edge_radius>
            </geometry>
        </rectangular_hole>
```

Specific tags:

- <geometry> - <pos>: This tag contains the x, y, z coordinates of the center point of the exported rectangular hole.
- <geometry> - <normal>: This tag contains the normalized vector of the normal of this rectangular hole.
- <geometry> - <dir>: This tag contains the normalized vector of the direction of this rectangular hole.
- <geometry> - <length>: This tag contains the length of this rectangular hole.
- <geometry> - <width>: This tag contains the width of this rectangular hole.
- <geometry> - <edge_radius>: This tag contains the radius of the circle at each corner of this rectangular hole.

5.7 Sphere

A sphere is written in the form:

```
<sphere …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <pos
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </pos>
        <radius>floating point number</radius>
    </geometry>
</sphere>
```

Specific tags:

- <geometry> - <pos>: This tag contains the x, y, z coordinates of the center point of the exported sphere.
- <geometry> - <radius>: This tag contains the radius of this sphere.

5.8 Cylinder

A cylinder is written in the form:

```
<cylinder …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <pos1
```

```
                    x="floating point number"
                    y="floating point number"
                    z="floating point number">
                </pos1>
                <pos2
                    x="floating point number"
                    y="floating point number"
                    z="floating point number">
                </pos2>
                <dir
                    x="floating point number"
                    y="floating point number"
                    z="floating point number">
                </dir>
                <radius>floating point number</radius>
                <length>floating point number</length>
                <location>
                    inner or outer
                </location>
            </geometry>
        </cylinder>
```

Specific tags:

- `<geometry> - <pos1>`: This tag contains the x, y, z coordinates of the base point of the visualized cylinder.
- `<geometry> - <pos2>`: This tag contains the x, y, z coordinates of the top point of the visualized cylinder.
- `<geometry> - <dir>`: This tag contains the normalized vector of the direction of this cylinder.
- `<geometry> - <radius>`: This tag contains the radius of this cylinder.
- `<geometry> - <length>`: This tag contains the length of the visualization of this cylinder.
- `<geometry> - <location>`: This tag contains the position where the cylinder was created.

5.9 Cone

A cone is written in the form:

```
<cone …>
```

please have a look at the beginning of chapter 4 for shared tags

```
<geometry>
    <pos1
        x="floating point number"
        y="floating point number"
        z="floating point number">
    </pos1>
    <pos2
        x="floating point number"
        y="floating point number"
        z="floating point number">
    </pos2>
    <dir
        x="floating point number"
        y="floating point number"
        z="floating point number">
    </dir>
    <radius1>floating point number</radius1>
    <radius2>floating point number</radius2>
    <angle>floating point number</angle>
```

```
            <origin
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </origin>
            <height1>floating point number</height1>
            <height2>floating point number</height2>
            <location>
                  inner or outer
            </location>
      </geometry>
</cone>
```

Specific tags:

- `<geometry> - <pos1>`: This tag contains the first x, y, z coordinates of the exported cone.
- `<geometry> - <radius1>`: This tag contains the first radius of this cone.
  Please note that the cone is not a frustum. The radius describes the element but it does not limit it for computations.
- `<geometry> - <pos2>`: This tag contains the second x, y, z coordinates of this cone.
- `<geometry> - <radius2>`: This tag contains the second radius of this cone.
  Please note that the cone is not a frustum. The radius describes the element but it does not limit it for computations.
- `<geometry> - <dir>`: This tag contains the normalized direction of the cone.
- `<geometry> - <angle>`: This tag contains the angle of the cone.
- `<geometry> - <origin>`: This tag contains the x, y, z coordinates of the apex.
- `<geometry> - <height1>`: This tag contains the distance from the apex to top cutoff.
- `<geometry> - <height2>`: This tag contains the distance from the apex to the bottom.
- `<geometry> - <location>`: This tag contains the position where the cylinder was created.

5.10 Polygon Hole

A polygon hole is written in the form:

```
<polygon_hole …>
```

please have a look at the beginning of chapter 4 for shared tags

```
      <geometry>
            <num_points>integer number</num_points>
            <pos
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </pos>
            <normal
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </normal>
            <dir
                  x="floating point number"
                  y="floating point number"
                  z="floating point number">
            </dir>
            <outer_radius>floating point number</outer_radius>
            or
            <inner_radius>floating point number</inner_radius>
```

```
        </geometry>
    </polygon_hole>
```

Specific tags:

- `<geometry> - <num_points>`: Number of edge points of this polygon hole. All edges have the same length.
- `<geometry> - <pos>`: This tag contains the x, y, z coordinates of the center point of the exported hole.
- `<geometry> - <normal>`: This tag contains the normalized vector of the normal of this hole.
- `<geometry> - <dir>`: This tag contains the normalized vector of the direction of the starting point of this hole.
- `<geometry> - <outer_radius>`: This tag contains the outer radius of this hole.
- `<geometry> - <inner_radius>`: This tag contains the inner radius of this hole. This can be used as an alternative to the `<outer_radius>` tag.

6 Dimensions

6.1 Scalar Dimensions

All scalar dimensions are defined by the tag name “dimension”. The type of the scalar can be derived from a tag entry in the geometry section that holds the value for the inspected dimension.
All types of dimensions share the following data:

```
<dimension …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>floating point number</geometry>
    <result>
        …
    </result>
</dimension>
```

Specific tags:

- <geometry>: This tag is for the dimension type described in chapter 4.1and holds the size of the dimension.
- <result>: This tag contains the result information for dimension described in chapter 4.1.

6.2 Angle

A dimension angle is written in the form:

```
<dimension …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <angle>floating point number</angle>
        <pos
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </pos>
        <pos1
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </pos1>
        <pos2
            x="floating point number"
            y="floating point number"
            z="floating point number">
        </pos2>
    </geometry>
</dimension>
```

Specific tags:

- <geometry> - <angle>: This tag contains the angle of the exported dimension angle.
- <geometry> - <pos>: This tag contains the x, y, z coordinates for the visualization of the dimension angle.
- <geometry> - <pos1>
  <geometry> - <pos2>: These tags contain the x,y,z coordinates of the points on an arm of an angle.

6.3 Distance

A dimension distance is written in the form:

```
<dimension …>
```

please have a look at the beginning of chapter 4 for shared tags

```
<geometry>
      <distance
            x="floating point number"
            y="floating point number"
            z="floating point number">
            floating point number
      </distance>
      <length>
            floating point number
      </length>
      <pos1
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </pos1>
      <pos2
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </pos2>
      <actual_pos1
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </actual_pos1>
      <actual_pos2
            x="floating point number"
            y="floating point number"
            z="floating point number">
      </actual_pos2>
      <restriction>
            floating point number
      </restriction>
</geometry>
</distance>
```

Specific tags:

- `<geometry> - <distance>`: This tag contains the distance vector of the exported distance.
- `<geometry> - <length>`: This tag contains the length of the exported distance.
- `<pos1>`
  `<pos2>`: These tags contain the x, y, z coordinates for visualization of the dimension distance.
- `<actual_pos1>`
  `<actual_pos2>`: These tags contain the x, y, z coordinates for actual part of the dimension and are optional.
- `<restriction>`: These tags contain the restriction of the distance if that is checked, otherwise this tag is left out.

6.4 Value Element

A value element is written in the form:

```
<value_element …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <unit type="unit description"/>
        <value>floating point number</value>
        <offset>floating point number</offset>
    </geometry>
</value_element>
```

Specific tags:

- `<geometry> - <unit>`: This tag contains the unit for the value.
- `<geometry> - <value>`: This tag contains the value itself.
- `<geometry> - <offset>`: This tag contains the offset for the value.

7 Meshes

Data for meshes and deviations are stored in a binary format that is written as a base64 string to XML. The binary data is described in table 3. Please note that these elements may create very big XML files on export.

Table 3: Binary Mesh Data

UINT32	Version number		
UINT32	Number of meshes		
For every mesh			
	VEC3D64	Minimum corner of bounding box	
	VEC3D64	Maximum corner of bounding box	
	UINT8	Color data available (0 = false, 1 = true)	
	UINT8	Distance data available (0 = false, 1 = true)	
	UINT32	Number of vertices	
	For every vertex		
		UINT32	Vertex x coordinate normalized from 0 to MAX_UINT regarding the bounding box x dimension.
		UINT32	Vertex y coordinate normalized from 0 to MAX_UINT regarding the bounding box y dimension.
		UINT32	Vertex z coordinate normalized from 0 to MAX_UINT regarding the bounding box z dimension.
		If distance data available	
			F32 Signed distance
			VEC3D32 Distance vector
		If color data available	
			RGBA Vertex color
	UINT32	Number of triangles	
	For every triangle		
		UINT32	Index of first vertex of triangle
		UINT32	Index of second vertex of triangle
		UINT32	Index of third vertex of triangle

7.1 Mesh

The tag name for a mesh is “mesh“. Meshes contain the following data:

```
<mesh …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <mesh chunk="chunk number">
            base64 encoded binary data
        </mesh>
    </geometry>
</mesh>
```

Specific tags:

- `<geometry> - <mesh>`: This tag holds the binary data encoded in a base64 string. The binary data of the mesh is described in detail at the beginning of this chapter. Depending on the mesh size, the data can be split into a number of chunks. Each <mesh> node represents a chunk where the chunk attribute of the node holds the appropriate chunk number starting with 0.

7.2 Colored Mesh

The tag name for a colored mesh is “colored_mesh“. Meshes share the following data:

```
<colored_mesh …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <mesh chunk="chunk number">
            base64 encoded binary data
        </mesh>
    </geometry>
</colored_mesh>
```

Specific tags:

- `<geometry> - <mesh>`: This tag holds the binary data encoded in a base64 string. The binary data of the mesh is described in detail at the beginning of this chapter. Depending on the mesh size, the data can be split into a number of chunks. Each <mesh> node represents a chunk where the chunk attribute of the node holds the appropriate chunk number starting with 0.

7.3 Surface Deviation

The tag name for a surface deviation is “surface_deviation“. They use the following tags:

```
<surface_deviation …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <element>
            name of the projection element
        </element>
        <projection_type>
            "point, plane, line, curve or surface"
        </projection_type>
        <mesh chunk="chunk number">
            base64 encoded binary data
        </mesh>
    </geometry>
</surface_deviation>
```

Specific tags:

- `<geometry> - <element>`: This tag defines the element on which the projection was made.
- `<geometry> - <projection_type>`: This tag defines the projection type that is used for projection. The valid projection types are depending on the referenced projection element.
- `<geometry> - <mesh>`: This tag holds the binary data encoded in a base64 string. The binary data of the mesh is described in detail at the beginning of this chapter. Depending on the mesh size, the data can be split into a number of chunks. Each <mesh> node represents a chunk where the chunk attribute of the node holds the appropriate chunk number starting with 0.

7.4 Deviation to Reference

The tag name for a deviation to a reference is “deviation_to_reference“. They use the following tags:

```
<deviation_to_reference …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <element>
            name of projection element
        </element>
        <projection_type>
            "point, plane, line, curve or surface"
        </projection_type>
        <mesh chunk="chunk number">
```

```
                    base64 encoded binary data
                </mesh>
            </geometry>
        </deviation_to_reference>
```

Specific tags:

- `<geometry> - <element>`: This tag defines the referenced element that is used for projection. If the referenced element is inside the XML file, it is linked by the ID via the *link* attribute. If it is not in the XML file it is linked via the name.
- `<geometry> - <projection_type>`: This tag defines the projection type that is used for projection. The valid projection types are depending on the referenced projection element.
- `<geometry> - <mesh>`: This tag holds the binary data encoded in a base64 string. The binary data of the mesh is described in detail at the beginning of this chapter. Depending on the mesh size, the data can be split into a number of chunks. Each <mesh> node represents a chunk where the chunk attribute of the node holds the appropriate chunk number starting with 0.

8 Sections and Point Clouds

A section can consist of multiple scan lines. Each scan line can have multiple sub scan lines. Data for sections are stored in a binary format which is written as a base64 string to XML. The binary data is described in table 4. Please note that these elements may create very big XML files on export.

Table 4: Binary Section Data

UINT32	Version number (1 or 2)			
	If the version is 2, the deviation type is stored			
		UINT32	Type of deviation 0 – no deviation 1 – xyz-deviation 2 – normal-deviation 3 – trim-deviation	
UINT32	Number of sections			
For every section				
UINT32	Number of scan lines			
For every scan line				
UINT32	Number of sub scan lines			
UINT8	Distance data available (0 = false, 1 = true)			
UINT8	Reserved for future use			
UINT8	Inspection flags available (0 = false, 1 = true)			
UINT8	Spring distance available (0 = false, 1 = true)			
UINT8	Trim distance available (0 = false, 1 = true)			
UINT8	Edge data available (0 = false, 1 = true)			
For every sub scan line				
UINT32	Number of vertices			
For every vertex				
VEC3D64	Vertex position			
For every vertex				
VEC3D32	Vertex normal			
If distance data available, for every vertex				
F32	Signed scalar distance			
VEC3DF32	Vector distance			
If inspection flags available, for every vertex				
UINT8	Inspection flags (bit field)			
1 (bit 0)	Sampled point			
2 (bit 1)	Distance vector allowed			
4 (bit 2)	Distance vector desired			
If spring distance available, for every vertex				
VEC3DF32	Spring distance			
If trim distance available, for every vertex				
VEC3DF32	Trim distance			
If edge data available, for every vertex				

				VEC3DF32	Vertex spring vector
				VEC3DF32	Vertex trim vector
				VEC3DF32	Edge spring vector
				VEC3DF32	Edge trim vector
		UINT8		0 = scan line open, >0 = scan line closed	

8.1 Curve, Section and Edge

The tag name for a curve is “curve“. Curves contain the following data:

```
<curve …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <curve_data>
            base64 encoded binary data
        </curve_data>
    </geometry>
</curve>
```

Specific tags:

- `<geometry> - <curve_data>`: This tag holds the binary data encoded in a base64 string. The binary data of the section is described in detail at the beginning of this chapter.

8.2 Section and Inspection Section (Same as Curve)

The tag name for a section and an inspection section is “section“. If a section is a nominal element it will be imported as an inspection section, otherwise an actual section will be created. Sections have the following additional data.

```
<section …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <plane>
```

please have a look at chapter 5.3 for the plane geometry tags

```
        </plane>
        <curve_data>
            base64 encoded binary data
        </curve_data>
    </geometry>
</section>
```

Specific tags:

- `<geometry> - <plane>`: This tag defines the creation plane of the section.
- `<geometry> - <curve_data>`: This tag holds the binary data encoded in a base64 string. The binary data of the section is described in detail at the beginning of this chapter.

8.3 Edge and Edge Deviation

The tag name for an edge and an edge deviation is “edge“. If an edge is a nominal element, it will be imported as an edge deviation, otherwise an actual edge will be created. Edges have the following additional data.

```
<edge …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <section_data>
```

```
                    base64 encoded binary data
                </section_data>
            </geometry>
        </edge>
```

Specific tags:

- <geometry> - <section_data>: This tag holds the binary data encoded in a base64 string. The binary data used for the edge is the same as for the section which is described in detail at the beginning of this chapter.

8.4 Point Cloud

A point cloud is written in the form:

```
<point_cloud …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <point
            x="floating point number"
            y="floating point number"
            z="floating point number">
            <normal
                x="floating point number"
                y="floating point number"
                z="floating point number">
            </normal>
        </point>
    </geometry>
</point_cloud>
```

Specific tags:

- <geometry> - <point>: This tag defines the point position by the three attributes x,y and z.
- <geometry> - <point>: This optional tag defines the point normal by the three attributes x,y and z.

8.5 Reference Points

Reference points are written in the form:

```
<reference_points …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        For every reference point a point-tag is added
        <point
            id="point id"
            type="uncoded" or "coded"
            x="floating point number"
            y="floating point number"
            z="floating point number">
            <normal
                x="floating point number"
                y="floating point number"
                z="floating point number">
            </normal>
            <properties
                common_point="true" or "false"
                deviation="floating point number"
                feature_point="true" or "false"
```

```
                        radius="floating point number"
                        thickness="floating point number"
                        tritop_point="true" or "false">
                    </properties>
                </point>
            </geometry>
        </reference_points>
```

Specific tags:

- `<geometry> - <point>:` This tag defines the reference point position by:
    - `id:` The point ID
    - `type:` The reference point type
    - The three attributes `x, y` and `z` for the position.
- `<geometry> - <normal>:` This tag defines the normal of the point by the three attributes `x, y` and `z`.
- `<geometry> - <properties>:` This tag defines the additional point parameters:
    - `common_point:` This point exists in more than one measurement series.
    - `deviation:` The measuring residual of the point.
    - `feature_point:` The point is a feature point.
    - `radius:` The measure radius of the point.
    - `tritop_point:` The point was created by a tritop measurement.

8.6 Point Group

A point group is written in the form:

```
<point_group …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <matrix>…</matrix>
        For every point of the point group a point-tag is added
        <point
            id="integer"
            type="uncoded" or "coded"
            x ="floating point number"
            y ="floating point number"
            z ="floating point number">
            <normal
                x ="floating point number"
                y ="floating point number"
                z ="floating point number">
            </normal>
            <properties
                common_point = "true or false"
                feature_point = "true or false"
                tritop_point = "true or false"
                deviation = "floating point number"
                radius = "floating point number"
                thickness = "floating point number">
            </properties>
        </point>
    </geometry>
</point_group>
```

Specific tags:

- <geometry> - <matrix>: This tag contains the transformation matrix of the exported point group (base64 binary encoded) in respect to the reference stage. This data is needed for a complete import into the GOM software but can be ignored by other imports. The data in later tags, e.g. position information, already includes this transformation.
- <geometry> - <point>: This tag contains the description of a point inside the point group. The following attributes are available:
  - id: Identification number of the point. Each point has a unique ID within its component. **Note**: The point IDs are not globally unique! The same IDs for different points can be used in different components.
  - type: Type of point. Value can be "**uncoded**", "**coded**" or "**unknown**".
  - <x>, <y>, <z>: These attributes contain the x, y, z coordinates of the point.
- <geometry> - <points> - <normal>: This tag contains the description of the point normal. The following attributes are available:
  - <x>, <y>, <z>: These attributes contain the x, y, z coordinates of the normal.
- <geometry> - <points> - <properties>: This tag contains additional properties describing the point. The following attributes are available:
  - <deviation>: Identification deviation for this point. **Note**: Not to be mistaken for the displacement!
  - <radius>: Radius of the (reference) point. If the value is < 0.0 the radius is not known.
  - <thickness>: Thickness of the (reference) point.
  - <common_point>: This tag is for internal use and describes how the point was created in the GOM software. If the point is a common point, the value is "**true**" otherwise "**false**".
  - <feature_point>: This tag is for internal use and describes how the point was created in the GOM software. If the point is a feature point, the value is "**true**" otherwise "**false**".
  - <tritop_point>: This tag is for internal use and describes how the point was created in the GOM software. If the point is a TRITOP point, the value is "**true**" otherwise "**false**".

8.7 Vector Field

A vector field is written in the form:

```
<vector_field …>
```

please have a look at the beginning of chapter 4 for shared tags

```
<geometry>
</geometry>
<results>
      For every point of the vector field a tolerance category-tag
      is added
      <tolerance category>
            <tolerance
                  lower_limit="floating point number"
                  upper_limit="floating point number"/>
            <deviation value="floating point number"
                  or invalid/>
            <measured value="floating point number"
                  or invalid/>
            or
            <measured>
                  x="floating point number"
```

```
                        y="floating point number"
                        z="floating point number"
                    </measured>
                </tolerance category>
            </results>
        </vector_field>
```

Specific tags:

- <geometry>: This tag contains geometry information. Because the vector field is used for inspection only, this field remains empty.

The vector field has a result tag that holds all information about the used tolerances, the measured values and the calculated deviations.

The result tag contains the following data:

- <*tolerance category*>: This tag contains the type of the tolerance, e.g. **displacement.**
- <*tolerance category*> - <tolerance>: This tag gives the upper and lower limits of the tolerance in its attributes lower_limit and upper_limit. This tag is only available if the tolerance is used.
- <*tolerance category*> - <deviation>: This tag gives the deviation value of this tolerance category. If the value is not calculated, this tag is set to the string value "**invalid**".
- <*tolerance category*> - <measured>: This tag gives the measured value of this tolerance category. Depending on the tolerance category, the value could be a vector that gives the measured values in its x,y,z attributes or a single floating point number. If the value is not calculated, this tag is set to the string value "**invalid**".

9 GD&T Elements

GD&T Elements are based on selections that will not be exported to this format. Due to this limitations an import of these elements is possible for some cases but they won't work as usual and may not recalculate. Importing these elements is not recommended.

9.1 Datum System

A datum system is written in the form:

```
<datum_system …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <datum_1 link="id">name</datum_1>
        <datum_1a link="id">name</datum_1a>
        <datum_2 link="id">name</datum_2>
        <datum_2a link="id">name</datum_2a>
        <datum_3 link="id">name</datum_3>
        <datum_3a link="id">name</datum_3a>
        <inclusion>0 or 1</inclusion>
    </geometry>
</datum_system>
```

Specific tags:

- `<geometry> - <datum_1>`, `<geometry> - <datum_1a>` `<geometry> - <datum_2>` `<geometry> - <datum_2a>` `<geometry> - <datum_3>` `<geometry> - <datum_3a>`: These tags define the datum system and the referenced elements. For some tolerances at least the `<datum_1>` tag has to be defined, the other tags are optional. If the referenced element is inside the XML file, it is linked by the ID via the *link* attribute. If it is not in the XML file, it is linked via the name. Please note: on import the datum system tries to link to the existing objects.
- `<inclusion>`: This tag defines if datum1 is included in datum2 or vice versa.

9.2 GD&T Tolerances

Not all of the following tags are used by all GD&T tolerances. Please take a look at the description for every tag for the limitations.
A GD&T tolerance is written in the form:

```
<type_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
        <standard>
            iso or asme
        </standard>
        <zone_type>
            planar, circular or spherical
        </zone_type>
        <datum_system link="id">
            name of datum system
        </datum_system>
        <datum_direction>
            0,1 or 2
        </datum_direction>
        <element link="id">
            name of fitting element
```

```
            </element>
            <material_requirement>
                  mmr or lmr
            </material_requirement>
            <use_rpr>0 or 1</use_rpr>
      </geometry>
</type_tolerance>
```

Specific tags:

- `<type_tolerance>`: This tag defines the type of the tolerance. The following types are supported: **flatness, cylindricity, roundness, straightness, parallelism, perpendicularity, angularity, position, circular_runout, total_runout** and **concentricity.** The type part of this tag is replaced by the used type of tolerance, e.g. `<flatness_tolerance>`.
- `<geometry> - <standard>`: This tag defines what standard should be used for calculating the tolerance. If this tag is not available, the default value "*iso*" is used.
- `<geometry> - <zone_type>`: This tag defines what zone type is used for the tolerance zone. This tag is only valid if the element is a **straightness, parallelism, perpendicularity, angularity** or **position tolerance**. The default value is "*planar*".
- `<geometry> - <datum_system>`: This tag defines the referenced datum system. If the referenced datum system is inside the XML file, it is linked by the ID via the *link* attribute. If it is not in the XML file, it is linked via the name. Please note: on import the tolerance tries to link to the referenced element. This tag is only valid for a **parallelism, perpendicularity, angularity, position** or **concentricity tolerance.**
- `<geometry> - <datum_direction>`: This tag defines the datum direction to be used. This tag is only valid for a **parallelism, perpendicularity, angularity, position** or **concentricity tolerance**. The default value is "*0*".
- `<geometry> - <element>`: This tag defines the referenced fitting element. If the referenced element is inside the XML file, it is linked by the ID via the *link* attribute. If it is not in the XML file, it is linked via the name. Please note: on import the tolerance tries to link to the referenced element.
- `<geometry> - <material_requirement>`: This optional tag defines the material requirement. The string value "*mmr*" sets a maximum material requirement, "*lmr*" sets a least material requirement. This tag is only valid, if the element is a **straightness, parallelism, perpendicularity, angularity, concentricity** or **position tolerance**.
- `<geometry> - <use_rpr>`: This optional tag defines if the reciprocity requirement is used. This tag is only valid, if the element is a **straightness, parallelism, perpendicularity, angularity, concentricity** or **position tolerance**. In addition, the tolerance has to have a material requirement set.
- `<geometry> - <dimension_element>`: This optional tag defines the scalar dimension element that is used for the calculation of the maximum/least material condition. This tag is only valid, if the element is a **straightness, parallelism, perpendicularity, angularity, concentricity** or **position tolerance**. In addition, the tolerance has to have a material requirement set. If the referenced dimension element is inside the XML file, it is linked by the ID via the *link* attribute. If it is not in the XML file, it is linked via the name. Please note: on import the tolerance tries to link to the referenced element.

9.2.1 Flatness Tolerance

The flatness tolerance is written in the form:

```
<flatness_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <element></element>
    </geometry>
</flatness_tolerance>
```

9.2.2 Position Tolerance

The position tolerance is written in the form:

```
<position_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <standard></standard>
        <zone_type></zone_type>
        <datum_system></datum_system>
        <element></element>
        <datum_direction></datum_direction>
        <true_angle></true_angle>
        <true_position_1></true_position_1>
        <true_position_2></true_position_2>
        <true_position_3></true_position_3>
    </geometry>
</position_tolerance>
```

Specific tags:

- `<true_angle>`: This tag defines the value of the nominal angle between the datum system and the element.
- `<true_position_1>`: This tag defines the value of the nominal distance to the first datum system plane.
- `<true_position_2>`: If there is a second datum system plane, this tag defines the nominal distance to it.
- `<true_position_3>`: If there is a third datum system plane, this tag defines the nominal distance to it.

9.2.3 Straightness Tolerance

The straightness tolerance is written in the form:

```
<straightness_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <zone_type></zone_type>
```

```
            <element></element>
        </geometry>
    </straightness_tolerance>
```

9.2.4 Roundness Tolerance

The roundness tolerance is written in the form:

```
    <roundness_tolerance …>
        please have a look at the beginning of chapter 4 for shared tags
        <geometry>
            please have a look at the beginning of chapter  9.2 for the shared geometry tags
            <element></element>
        </geometry>
    </roundness_tolerance>
```

9.2.5 Cylindricity Tolerance

The cylindricity tolerance is written in the form:

```
    <cylindricity_tolerance …>
        please have a look at the beginning of chapter 4 for shared tags
        <geometry>
            please have a look at the beginning of chapter 9.2 for the shared geometry tags
            <element></element>
        </geometry>
    </cylindricity_tolerance>
```

9.2.6 Angularity Tolerance

The angularity tolerance is written in the form:

```
    <angularity_tolerance …>
        please have a look at the beginning of chapter 4 for shared tags
        <geometry>
            please have a look at the beginning of chapter 9.2 for the shared geometry tags
            <standard></standard>
            <zone_type></zone_type>
            <datum_system></datum_system>
            <element></element>
            <datum_direction></datum_direction>
            <true_angle></true_angle>
        </geometry>
    </angularity_tolerance>
```

Specific tags:

- `<true_angle>`: This tag defines the value of the nominal angle between the referenced fitting element and the datum system.

9.2.7 Perpendicularity Tolerance

The perpendicularity tolerance is written in the form:

```
<perpendicularity_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <standard></standard>
        <zone_type></zone_type>
        <datum_system></datum_system>
        <element></element>
        <datum_direction></datum_direction>
    </geometry>
</perpendicularity_tolerance>
```

9.2.8 Parallelism Tolerance

The parallelism tolerance is written in the form:

```
<parallelism_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <standard></standard>
        <zone_type></zone_type>
        <datum_system></datum_system>
        <element></element>
        <datum_direction></datum_direction>
    </geometry>
</parallelism_tolerance>
```

9.2.9 Concentricity Tolerance

The concentricity tolerance is written in the form:

```
<concentricity_tolerance …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <standard></standard>
        <datum_system></datum_system>
        <element></element>
        <datum_direction></datum_direction>
    </geometry>
</concentricity_tolerance>
```

9.2.10 Circular Runout

The circular runout is written in the form:

```
<circular_runout …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <datum_system></datum_system>
        <element></element>
        <datum_direction></datum_direction>
    </geometry>
</circular_runout>
```

9.2.11 Total Runout

The total runout is written in the form:

```
<total_runout …>
```

please have a look at the beginning of chapter 4 for shared tags

```
    <geometry>
```

please have a look at the beginning of chapter 9.2 for the shared geometry tags

```
        <datum_system></datum_system>
        <element></element>
        <datum_direction></datum_direction>
    </geometry>
</total_runout>
```