INTERNATIONAL STANDARD

ISO 25178-72

First edition
2017-05

Geometrical product specifications (GPS) — Surface texture: Areal —

Part 72: XML file format x3p

Spécification géométrique des produits (GPS) — État de surface: Surfacique —

Partie 72: Format de fichier XML x3p

Reference number
ISO 25178-72:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 213, *Dimensional and geometrical product specifications and verification*.

A list of all parts in the ISO 25178 series can be found on the ISO website.

Introduction

This document is a geometrical product specification (GPS) standard and is to be regarded as a general GPS standard (see ISO 14638). It influences the chain link F of the chains of standards on profile and areal surface texture.

The ISO/GPS matrix model given in ISO 14638 gives an overview of the ISO/GPS system of which this document is a part. The fundamental rules of ISO/GPS given in ISO 8015 apply to this document and the default decision rules given in ISO 14253-1 apply to the specifications made in accordance with this document, unless otherwise indicated.

For more detailed information of the relation of this document to other standards and the GPS matrix model, see Annex C.

The x3p format was in use in industry and academia before the creation of this document. The x3p file format as defined in this document has been developed based on the definitions in ISO 5436-2. The openGPS®[1)] consortium provides a free open source software implementation of this file format to avoid the inevitable inconsistency of multiple proprietary implementations.

1) openGPS® is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of this product.

Geometrical product specifications (GPS) — Surface texture: Areal —

Part 72:
XML file format x3p

1 Scope

This document defines the XML file format x3p for storage and exchange of topography and profile data.

2 Normative references

The following document is referred to in the text in such a way that some or all of its content constitutes requirements 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.

ISO 25178-600[2], *Geometrical product specifications (GPS) — Surface texture: Areal — Part 600: Metrological characteristics for areal-topography measuring methods*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 25178-600 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at http://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
zip-container
file format that can be used as a container for multiple files and folders that does also support a compression of the stored content

Note 1 to entry: The file format description is in the public domain[1].

3.2
md5
method to calculate a unique 16-byte binary checksum used to check the integrity of files

Note 1 to entry: The binary value is typically represented by 32 hexadecimal digits.

Note 2 to entry: See Reference [2].

3.3
int16
2-byte representation of a signed integer

Note 1 to entry: The int16 type has a minimum value of –32 768 and a maximum value of 32 767.

2) Under preparation. Stage at the time of publication: ISO/DIS 25178-600.

Note 2 to entry: The less significant bytes are stored in memory addresses lower than those in which are stored the more significant bytes.

3.4
int32
4-byte representation of a signed integer

Note 1 to entry: The int32 type has a minimum value of -2 147 483 648 and a maximum value of 2 147 483 647.

Note 2 to entry: The less significant bytes are stored in memory addresses lower than those in which are stored the more significant bytes.

3.5
float32
4-byte representation of a floating point number according to IEEE 754

Note 1 to entry: The float32 type has a minimum value of -2^{128} and a maximum value of 2^{128}. The smallest positive number representable is 2^{-126}.

Note 2 to entry: The ASCII representation is a signed floating point number with 8 digits and a signed two-digit exponent in the range [-38.. +38].

Note 3 to entry: The less significant bytes are stored in memory addresses lower than those in which are stored the more significant bytes.

3.6
float64
8-byte representation of a floating point number according to IEEE 754

Note 1 to entry: The float64 type has a minimum value of $-2^{1\,024}$ and a maximum value of $2^{1\,024}$. The smallest positive number representable is $2^{-1\,022}$.

Note 2 to entry: The ASCII representation is a floating point number with 16 digits and a signed three-digit exponent in the range [- 308.. + 308].

Note 3 to entry: The less significant bytes are stored in memory addresses lower than those in which are stored the more significant bytes.

3.7
not a number
NaN
special floating point value defined in IEEE 754 specifying a number that is not computable

Note 1 to entry: Some floating point implementations define more than one value for NaN to distinguish between Quiet NaNs and Signaling NaNs. In this case the Quiet NaN is preferred.

Note 2 to entry: All mathematical operations incorporating a NaN value yield NaN as result. As a consequence, all comparisons with a NaN value yield “unequal”. This is especially true for the equality comparison of two NaN values.

3.8
element
start tag followed by a data value followed by an end tag

EXAMPLE 1 An element with the name “example” comprising a start and an end tag would be implemented as

<example>contents of element</example>

EXAMPLE 2 An empty element with the name “example” would be implemented as

<example/>

Note 1 to entry: An element begins with a start tag and ends with an end tag. Alternatively, an element may consist of an empty tag solely. The content of the element is between the start and end tag and may contain further elements.

3.9
extensible markup language
XML
language for encoding documents electronically

Note 1 to entry: XML is a subset of SGML (see Reference [7]).

3.10
uniform resource locator
URL
character string to locate a resource in a computer network or on a local computer

EXAMPLE A well-known use of a URL is the specification of a web site's address like "http://www.iso.org/".

3.11
uniform resource identifier
URI
character string uniquely identifying a name or resource in a hierarchical style

EXAMPLE A URI for this document could be "www.iso.org/ISO_25178_Part_72".

Note 1 to entry: A URL is the most common form of a URI.

Note 2 to entry: The relation between a URI and a URL is like the relation between a person's name (the URI) and a person's address (the URL).

Note 3 to entry: To create a unique URI, it is good practice to start a URI with a domain name that has been registered on the name of the owner.

3.12
offset
distance of the stored geometric data to the origin of the coordinate system along one axis of the coordinate system

3.13
rotation matrix
3×3 matrix defining the rotation of the data set in 3D space

Note 1 to entry: It defines the orientation of the stored point cloud in 3D space.

3.14
global coordinate system
three-dimensional coordinate system in which the position and orientation of the original point cloud is defined

3.15
view coordinate system
three-dimensional coordinate system in which the 3D points are defined

Note 1 to entry: In the view coordinate system, the represented surface or point cloud typically is projectable along one spatial direction.

3.16
data matrix
one-, two- or three-dimensional array of 3D points with a defined neighbourhood relation

Note 1 to entry: Each 3D point has two neighbours along each matrix dimension. The data matrix contains point coordinates in the view coordinate system.

Note 2 to entry: The index in the data matrix is described by the symbols *u*, *v*, and *w*.

Note 3 to entry: The array dimensions of the data matrix should not be confused with the spatial dimensions of the global coordinate system or view coordinate system.

4 Requirements

4.1 Units

All coordinates shall be specified in metres. Other units shall not be used. SI Prefixes shall not be used.

4.2 Recommended offset value

The offset should be set to a value so that the stored point cloud is centred on the origin of the coordinate system.

5 x3p file format

5.1 General

An x3p file is a zip-container for areal and profile data. It can be flexibly used for point clouds without any topology as well as for projectable 2½D topography data and for multilayer topography representations.

NOTE A general container format is described in Reference [5].

5.2 File name extension

The name of a file stored in x3p data format shall end with the string ".x3p". On case sensitive file systems the string shall be typed in lower case letters.

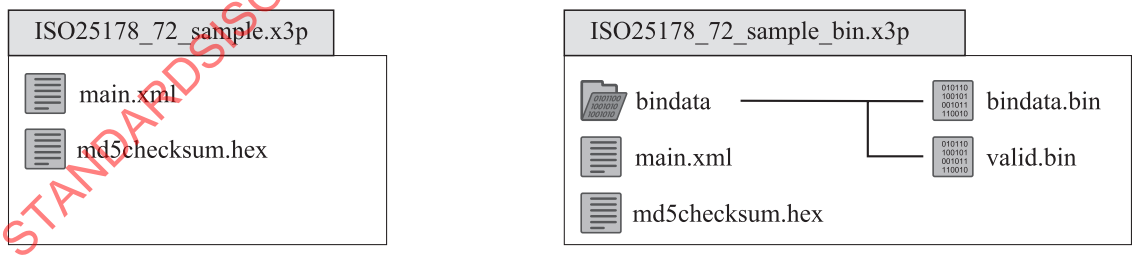
EXAMPLE 1 samplefile.x3p

EXAMPLE 2 longer_filename example123.x3p

5.3 Minimum contents of zip-container

The zip-container representing an x3p file shall contain as a minimum the files "`main.xml`" and "`md5checksum.hex`" in its root directory as displayed in Figure 1 a).

EXAMPLE Figure 1 b) shows a more complex example of the contents of the zip-container.



a) example with minimum contents of an x3p file container, text only format

b) example with binary encoded coordinates "bindata.bin" and binary validity mask "valid.bin"

Figure 1 — x3p container examples

5.4 Optional contents of zip-container

5.4.1 General

The zip-container may contain more files depending on the type and encoding of the stored data.

5.4.2 Binary encoded coordinates

When storing coordinates in a binary encoded file, it should be placed in a subdirectory named "`bindata`" and the file should be named "`bindata.bin`".

NOTE Specifying a different name does not result in a dysfunctional file, because the relative path name to this file is stored in main.xml.

5.4.3 Validity mask

When storing a validity mask in a binary encoded file, it should be placed in a subdirectory named "`bindata`" and the file should be named "`valid.bin`".

NOTE Specifying a different name does not result in a dysfunctional file, because the relative path name to this file is stored in main.xml.

5.4.4 Vendor specific extensions

Vendor specific extensions shall be used to extend x3p-format to a custom file format. Vendor specific extensions can use any file type and any filename except the filenames defined in 5.3.

EXAMPLE A vendor specific extension could be an image file named "photography_of_sample.jpg".

5.5 Contents and format of main.xml

5.5.1 General

The exact specification of the xml data structures used in main.xml is defined in Annex A. Here, only the content of the elements and their usage are described.

5.5.2 Main records

The file `main.xml` contains a sequence of four main records and a vendor specific extension:

— Record1: header, data types and axes definitions (see 5.5.3)

— Record2: optional record containing the document's meta data (see 5.5.4)

— Record3: the data (see 5.5.5)

— Record4: an md5 checksum of the XML-document (see 5.5.6)

— Vendor specific extensions (see 5.5.7)

5.5.3 Record1: Header, data types, and axes definitions

5.5.3.1 Revision

The `Revision` record shall contain the string "ISO 5436:2000".

NOTE This is not a reference to ISO 5436, it is only an identification string.

5.5.3.2 FeatureType

5.5.3.2.1 General

The `FeatureType` element specifies the class of 3D data stored in the file. The contents of feature type shall be one of the strings "PRF", "SUR", "PCL". These names correspond to profile, surface and point cloud feature types.

5.5.3.2.2 PRF – Profile

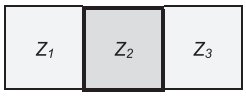
The 3D data in the x3p file represent a profile i.e. a linear sequence of 3D coordinates. Points are stored in a one-dimensional array for single layer profiles or in a two-dimensional array for multilayer profiles. Each point has up to two neighbours for a single layer profile or up to four neighbours in a multilayer profile. See Figure 2.

It shall be assured that the neighbourhood relation of all points in 3D space is the same as in the array.

NOTE 1 A 3D points matrix index *u*, *v*, *w* should not be confused with its 3D coordinates *x*, *y*, *z*.

NOTE 2 The case of a two dimensional matrix is used for multilayer profile representations. The array index *w* represents the index of the layer in this case.

NOTE 3 The 3D coordinates of all points in a profile do not need to be located on a straight line in 3D space. Profile can follow any path in space.



Key

Z_u 3D coordinates of point at matrix location

NOTE Each point has up to two direct neighbours.

Figure 2 — Sample neighbourhood relation of a 3D point in a "PRF" type feature

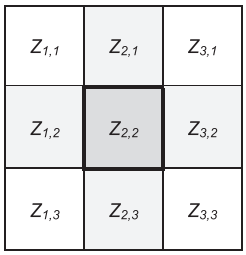
5.5.3.2.3 SUR – Surface

The 3D data in the x3p file represent the topography of a projectable surface with a well-defined topology, i.e. a neighbourhood relation for each 3D point. Points are stored in a two- or three-dimensional array and each array element has a maximum of four or six direct neighbouring elements respectively, see Figure 3.

It shall be assured that the neighbourhood relation of all points in 3D space is the same as in the array.

NOTE 1 A 3D points matrix position *u*, *v*, *w* should not be confused with its 3D coordinates *x*, *y*, *z*.

NOTE 2 The case of a three-dimensional matrix is used for multilayer surface representations. The array index *w* represents the index of the layer in this case.



Key

$Z_{u,v}$ 3D coordinates of point at matrix location *u*, *v*

NOTE Each point has up to four direct neighbours.

Figure 3 — Sample neighbourhood relation of a 3D point for a "SUR" type feature

5.5.3.2.4 PCL – Point cloud

The 3D data in the x3p file represent a cloud of non-related points in 3D space. Points are stored in an unordered list and their neighbourhood relation is unknown.

NOTE The point cloud representation may be useful for 3D data from coordinate measurement machines (CMM) or for data from unknown sensor types with an unknown point topology.

5.5.3.3 Axes

5.5.3.3.1 General

The Axes elements shall be used to store the description of the coordinate system. It shall contain a description for each axis in its three elements named CX, CY, and CZ of type `AxisType`. The structure of `AxisType` elements is described in the following clauses.

5.5.3.3.2 AxisType

5.5.3.3.2.1 General

The `AxisType` element shall be one of the letters "I" for incremental axis or "A" for absolute axis.

5.5.3.3.2.2 Incremental axis type

For x and y axes, an incremental type defines the calculation of x and y coordinates from the matrix indices u and v where

x, y are the spatial coordinates of the point;

u, v are the matrix indices of the point;

O is the offset of the point from the coordinate origin in metres;

I is the increment in metres.

The z axis shall not be incremental.

5.5.3.3.2.3 Absolute axis type

An absolute axis type shall be used for the explicit storage of x, y, and z coordinates. The z axis shall be of absolute type.

NOTE Compared to an incremental axis type, the absolute axis type causes a higher memory usage for x and y coordinates. The amount of memory used is as large as for the z coordinate because for each 3D point the x and y coordinate has to be stored separately. Therefore, it is recommended to use incremental x and y axes whenever possible, i.e. when point spacing is regular and homogenous.

5.5.3.3.3 DataType

The `DataType` element shall contain one of the letters "I" for int16 data, "L" for int32 data, "F" for float32 data and "D" for float64 data.

5.5.3.3.4 Increment

The `Increment` element shall contain a positive length value in metres specifying the increment of the axis. Increment shall not be zero. The increment values for the x, y and z axes are named with the symbols I_x, I_y, and, I_z.

5.5.3.3.5 Offset

The `Offset` element shall specify the distance to the coordinate origin in metres. The offset may be positive or negative. The offset values for the *x*, *y* and *z* axes are named with the symbols O_x, O_y, and, O_z.

5.5.3.4 Rotation

The Rotation element shall specify a 3×3 elements transformation matrix R with the elements in Formula (1):

$$\mathrm{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad (1)$$

The matrix R shall only contain a rotation transformation. It shall not contain other transformations like mirroring, scaling or shearing.

5.5.3.5 Coordinate transformation

The calculation of the global coordinates from the view coordinates of the stored 3D points is done using Formula (2):

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} O_x \\ O_y \\ O_z \end{pmatrix} \quad (2)$$

where

X, Y, Z global coordinates;

x, y, z view coordinates.

5.5.4 Record2: Meta data

5.5.4.1 General

Record2 contains the data set's meta information. Record2 is optional but it is strongly recommended to specify it as complete as possible to increase the traceability of the contained data set.

5.5.4.2 Date

The Date element shall contain date and time of data set creation in the format "YYYY-MM-DDThh:mm:ss.sTZD" according to Reference [9].

EXAMPLE 2014-07-27T17:45:09.6+02:00

5.5.4.3 Creator

The `Creator` element should contain the name of the person and/or his/her institution or company who created the data set.

National privacy protection or data protection regulations may limit the use of this field in some countries.

EXAMPLE Tom Jones, Universal Metrology Institute

5.5.4.4 Instrument

5.5.4.4.1 General

The `Instrument` element shall contain a description of the measurement instrument or software used to create the data set. The following elements shall be used for the description.

5.5.4.4.2 Manufacturer

The Manufacturer element shall contain the name of the instrument manufacturer.

EXAMPLE Examplebrand Precision Instruments

5.5.4.4.3 Model

The Model element shall contain the model name of the instrument or software used to create the data set.

EXAMPLE ExampleModel Superfluorescence Cosinus Trigonometre

5.5.4.4.4 Serial

The `Serial` element shall contain the serial number of the instrument used to measure the data set. In case of a software created data set, this element may be empty.

EXAMPLE 1 S/N 1013

EXAMPLE 2 314ABC15/D

5.5.4.4.5 Version

The `Version` element shall contain the version number(s) of the instrument and/or software used to create the data set.

5.5.4.5 CalibrationDate

The `CalibrationDate` element should contain the date and time when the last calibration of the instrument was performed. Format of the string is "YYYY-MM-DDThh:mm:ss.sTZD" according to Reference [9]. If the instrument has not been calibrated yet, this element shall be missing.

EXAMPLE 2014-04-30T13:58:02.6+02:00

5.5.4.6 ProbingSystem

5.5.4.6.1 General

The ProbingSystem element shall describe the type and identification of the probing system used. For an optical instrument, this should be the lens specification; for a tactile system, this should be the specification of the stylus.

5.5.4.6.2 Type

The `Type` element shall describe the kind of the probing system. This shall be one of "Contacting" for stylus type instruments, "NonContacting" for optical or other non-contacting instruments or "Software". The last one shall be used for synthetic data sets, filtered data sets or soft-gauges.

EXAMPLE NonContacting

5.5.4.6.3 Identification

The `Identification` element shall provide a description of the probing system as specific as possible. This could include information about the tip specification for a contacting instrument or the objective specification for a non-contacting optical system. If a serial number of the probing system is available, it should be specified here. For "Software" type instruments, it should describe the software used to create the data as specific as possible.

5.5.4.7 Comment

The Comment element shall contain a string describing the data set as precise as possible.

EXAMPLE 1 First measurement from Tom Jones' steel samples #4711, upper left corner

EXAMPLE 2 Areal sinusoidal softgauge with a period length of 100 µm and Sz=1 µm

5.5.5 Record3: 3D point data

5.5.5.1 General

The `Record3` element shall contain specifications for data organization and the actual 3D point coordinates.

5.5.5.2 Specification of data organization

5.5.5.2.1 General

Depending on the organization of the data set to be stored, either `MatrixDimension` or `ListDimension` element shall be used to specify the size of the data set:

a) Matrix data type shall be used to store PRF or SUR type features that have a well-defined topologic neighbourhood relation of 3D points.

b) List type shall be used to store PCL type features in an unsorted list of 3D points with an undefined neighbourhood relation.

5.5.5.2.2 MatrixDimension

The `MatrixDimension` element shall contain the three elements `SizeX`, `SizeY`, and `SizeZ` defining the size of the data matrix in *u*, *v*, and *w* dimensions.

The names of the elements SizeX, SizeY and SizeZ may be misleading because they do not necessarily define anything directly related to *x*, *y* and *z* dimensions of the 3D coordinates. In datasets with incremental *x* and *y* axes the following relation between *u* and *x*, as well as between *v* and *y* and SizeY, holds:

$x = u - 1, y = \text{SizeY} - v.$

EXAMPLE 1 Definition of a matrix with 4×4 points and one surface layer:

<SizeX>4</SizeX> <SizeY>4</SizeY> <SizeZ>1</SizeZ>

EXAMPLE 2 Definition of a matrix for a profile data set with 10 points and two profile layers:

<SizeX>10</SizeX> <SizeY>1</SizeY> <SizeZ>2</SizeZ>

5.5.5.2.3 ListDimension

The ListDimension element shall define the total number of 3D points in the data set.

EXAMPLE 4711

5.5.5.3 Storage of 3D coordinates

5.5.5.3.1 General

The 3D coordinates of the data set can be stored in two ways: Either directly in the `DataList` element encoded as floating point numbers in ASCII character set or as binary files in several formats. The link to the binary files shall be specified with the `DataLink` element.

The binary representation of 3D coordinates allows for more compact storage and faster reading and writing. It is recommended that a data set with more than 10,000 3D points should be stored in binary representation. It is explicitly allowed to store smaller data sets in binary form too.

5.5.5.3.2 DataList

5.5.5.3.2.1 General

The `DataList` element shall contain a character representation of the 3D coordinates. The 3D points shall be ordered with *u* as the fastest index, *v* as the second index and *w* as the slowest index. An invalid or missing data point in a PRF or SUR type feature shall be identified by an empty element. Missing data points shall not be used in PCL type features.

NOTE Indexes *u* and *v* correspond to the matrix position inside one layer, while index *w* defines the layer number of a data point. For single layer data sets, *w* is always 0.

EXAMPLE For a matrix of *u*, *v*, *w*-dimensions 3,3,2, the order of the 3D points $P_{u,v,w}$ is

P(1,1,1), P(2,1,1), P(3,1,1),

P(1,2,1), P(2,2,1), P(3,2,1),

P(1,3,1), P(2,3,1), P(3,3,1),

P(1,1,2), P(2,1,2), P(3,1,2),

P(1,2,2), P(2,2,2), P(3,2,2),

P(1,3,2), P(2,3,2), P(3,3,2)

5.5.5.3.2.2 Datum

Each `Datum` element shall contain the 3D coordinates in metres of one 3D point. This may be one, two, or three coordinate values depending on the axes types. Multiple coordinates shall be separated by a semicolon ";". For each 3D point one `Datum` element shall be added to the list.

EXAMPLE 1 A 3D Point with a *z* coordinate only: <Datum>-8.08368571682830E-0001</Datum>

EXAMPLE 2 An invalid 3D point: <Datum/>

EXAMPLE 3 A 3D point with three coordinates: <Datum>8.23e-6;-3.5e-6;-1.99423423e-9</Datum>

5.5.5.3.3 DataLink

5.5.5.3.3.1 General

The DataLink element shall be used to establish a link to a file in the zip container with a binary representation of the profile data. The following elements shall be used.

5.5.5.3.3.2 PointDataLink

The `PointDataLink` element shall contain a local URL linking to the external binary file. The link shall not point to an external resource like an internet resource. Software implementing this document shall take measures to avoid an access of network resources by specifying a non-local URL.

EXAMPLE "`bindata/data.bin`"

5.5.5.3.3.3 MD5ChecksumPointData

The `MD5ChecksumPointData` element shall contain the MD5 checksum[2] for the binary file linked to by `PointDataLink`.

EXAMPLE 2e7c6f94ec07e3ef0cd43c0844b31253

5.5.5.3.3.4 ValidPointsLink

The `ValidPointsLink` element shall specify a local URL linking to an optional matrix with the recommended name "`bindata/valid.bin`". This file shall contain a packed array of Boolean values (a bit field). The value true (bit value 1) is associated with a valid 3D point.

NOTE 1 This element is optional and only needed if the binary coordinate file format does not support special values to mark invalid points.

NOTE 2 If no validity file is specified, all 3D points are valid.

EXAMPLE ''`bindata/valid.bin`''

5.5.5.3.3.5 MD5ChecksumValidPoints

The element `MD5ChecksumValidPoints` corresponds to element `ValidPointsLink` and shall contain the MD5 checksum for the validity file.

EXAMPLE b8114a9c9a26c5680be3c4d89073bde1

5.5.5.3.4 Binary file format

5.5.5.3.4.1 General

It is good practice to choose binary storage of 3D coordinates for data sets with more than 10,000 3D points, because the binary storage is more efficient in terms of memory usage and allows fast reading and writing of data sets.

5.5.5.3.4.2 Formats of binary files

Data formats for 3D coordinates

The binary 3D coordinate files shall use one of four different binary representations:

a) Int16: 16 bit signed integer (see 3.3)

b) Int32: 32 bit signed integer (see 3.4)

c) Float32: 32 bit floating point (see 3.5)

d) Float64: 64 bit floating point (see 3.6)

The 3D coordinates shall be written in the binary file specified by the `PointDataLink` element. The individual coordinates for each 3D point shall be written with the least significant byte first and the most significant byte last. This order is also called "little-endian". The position index of a data point in a

binary file is defined in 5.5.5.3.2.1. Each 3D point's coordinates shall start with *x* coordinate followed by *y* coordinate and *z* coordinate. There shall be no separating bytes between coordinate values or 3D points.

EXAMPLE 1 The sequence of coordinates in the binary file for an x3p file with absolute *x*, *y*, and *z* axes has the following order, only the first elements of the file are shown:

address offset	1	2	3
0	$x_{1,1,1}$	$y_{1,1,1}$	$z_{1,1,1}$
3	$x_{2,1,1}$	$y_{2,1,1}$	$z_{2,1,1}$
6	$x_{3,1,1}$	$y_{3,1,1}$	$z_{3,1,1}$
…	…	…	…
…	$x_{1,2,1}$	$y_{1,2,1}$	$z_{1,2,1}$
…	$x_{2,2,1}$	$y_{2,2,1}$	$z_{2,2,1}$
…	$x_{3,2,1}$	$y_{3,2,1}$	$z_{3,2,1}$
…	…	…	…

EXAMPLE 2 The sequence of coordinates in the binary file for an x3p file with incremental *x*, *y*, and absolute *z* axes has the following order, only the first elements of the file are shown:

address offset	0
0	$z_{1,1,1}$
1	$z_{2,1,1}$
2	$z_{3,1,1}$
…	…
…	$z_{1,2,1}$
…	$z_{2,2,1}$
…	$z_{3,2,1}$
…	…

5.5.5.3.5 Binary validity file format

The binary validity file shall be written as packed array of bits. The bit index *j* into the packed array shall be calculated in the same way as described for the data list in 5.5.5.3.2.1 from the bit index the byte position j_8 and the bit position j_1 in the packed array shall be calculated using Formulae (3) and (4):

$$j_8 = \left\lfloor \frac{j}{8} \right\rfloor \tag{3}$$

$$j_1 = j - 8 \cdot j_8 \tag{4}$$

The square Gauss brackets in Formula (3) calculate the next smaller integer for a real number.

EXAMPLE See Table 1 for a sample calculation of the indices.

Table 1 — Example calculation of byte and bit index for binary validity file

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	13	15	16
j_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2
j_8	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0

5.5.5.4 Handling of invalid points

5.5.5.4.1 General

Invalid topography points with missing data shall be marked in the case of a data matrix. In the case of a data list, i.e. an unordered point cloud, they shall not be written in the list.

NOTE *x* and *y* coordinates of an invalid point are implicitly defined in case of incremental *x* and or *y* axes. In case of absolute axes, they should be defined if they are known. This allows later interpolation of missing points.

5.5.5.4.2 XML representation

The `Datum` element of an invalid 3D point shall be empty.

5.5.5.4.3 Binary representation in float32 and float64 format

In binary float32 and float64 format, invalid z coordinates shall be set to the special value *Not-a-Number* (NaN).

5.5.5.4.4 Binary representation in int16 and int32 format

For the integer formats int16 and int32 a special value as Not a Number is not defined. All possible binary values represent valid numbers. Therefore, a second binary file shall be created containing a packed array of bits. Each bit shall represent the validity of a single 3D point. A bit with value "1" shall denote a valid 3D point and a bit with value "0" shall denote an invalid one.

5.5.6 Record4: Checksum information

The element `Record4` shall contain the element `ChecksumFile` containing a link or a URL to a file in the zip container containing the md5 checksum of "main.xml". The checksum file shall be stored in the root directory of the container as "md5checksum.hex".

5.5.7 Vendor specific extensions

The `VendorSpecificID` element shall be used to identify extensions of the x3p file format. This tag shall contain a vendor specific ID which is a URI created by the vendor. It shall be worldwide unique. When reading an x3p file an unknown `VendorSpecificID` element can be safely ignored as well as all optional contents of the zip container.

NOTE 1 A good practice to create a worldwide unique URI is using a domain name owned by the vendor and appending some string to identify the file format extension. This name is guaranteed to be unique at least for the lifetime of the domain.

NOTE 2 An x3p file containing vendor specific extensions keeps full compatibility to all software able to read standard x3p files.

Annex A
(informative)

XML file format

A.1 General

The format of the xml file "main.xml" is defined using the *W3C XML Schema Definition* Language *(XSD)* [7] a human and machine readable description language for XML data files. This approach allows the automatic generation of software programs that are able to read, write and verify the contents of "main.xml". This approach has been preferred to a probably incomplete description in human readable form.

NOTE 1 The string "ISO 5436_2" is used in some portions of the XSD. This has historical reasons and is no reference to ISO 5436-2 but is used as a URI.

NOTE 2 This XSD file has been released as open source by openGPS consortium under the LGPL license 2.0 or newer.

A.2 Schema definition of main XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.opengps.eu/2008/ISO 5436_2" xmlns:xsd="http://www
.w3.org/2001/XMLSchema" xmlns="http://www.opengps.eu/2008/ISO 5436_2"
elementFormDefault="unqualified">
  <xsd:annotation>
      <xsd:documentation>
      XML-implementation for ISO 5436-2 file format.
      04-Apr-2007
      Copyright by Georg Wiora (NanoFocus AG), Jörg Seewig (Uni Hannover),
      Andreas Walther (NanoFocus AG), Mark A. Weber (NanoFocus AG) 2007

      This file is part of the openGPS(R)[TM] software library.
      This program is free software; you can redistribute it and/or modify
      it under the terms of the GNU Lesser General Public License (LGPL)
      as published by the Free Software Foundation; either version 3 of
      the License, or (at your option) any later version.
      for detail see the files "licence_LGPL-2.0.txt" and
      "licence_GPL-2.0.txt".

      openGPS is distributed in the hope that it will be useful,
      but WITHOUT ANY WARRANTY; without even the implied warranty of
      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
      GNU Lesser General Public License for more details.

      You should have received a copy of the GNU General Public License
      along with this program.  If not, see http://www.gnu.org/licenses/.

      openGPS and the openGPS logo is a registered trademark of
      Physikalisch Technische Bundesanstalt (PTB)
      http://www.ptb.de/

      More information about openGPS can be found at
      http://www.opengps.eu/
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="ISO 5436_2" type="ISO 5436_2Type">
  </xsd:element>
  <xsd:complexType name="Record1Type">
      <xsd:sequence>
        <xsd:element name="Revision" type="xsd:token" minOccurs="1" maxOccurs="1">
        </xsd:element>
```

```
        <xsd:element name="FeatureType" maxOccurs="1" minOccurs="1">
             <xsd:simpleType>
               <xsd:restriction base="xsd:token">
                  <xsd:whiteSpace value="collapse">
                  </xsd:whiteSpace>
                  <xsd:enumeration value="PRF">
                  </xsd:enumeration>
                  <xsd:enumeration value="SUR">
                  </xsd:enumeration>
                  <xsd:enumeration value="PCL">
                  </xsd:enumeration>
               </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Axes" type="AxesType" maxOccurs="1" minOccurs="1">
        </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ISO 5436_2Type">
      <xsd:sequence>
        <xsd:element name="Record1" type="Record1Type" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="Record2" type="Record2Type" maxOccurs="1" minOccurs="0">
        </xsd:element>
        <xsd:element name="Record3" type="Record3Type" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="Record4" type="Record4Type" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="VendorSpecificID" type="xsd:anyURI" minOccurs="0" maxOccurs="1">
        </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Record2Type">
      <xsd:sequence>
        <xsd:element name="Date" type="xsd:dateTime" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="Creator" type="xsd:token" maxOccurs="1" minOccurs="0">
        </xsd:element>
        <xsd:element name="Instrument" type="InstrumentType" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="CalibrationDate" type="xsd:dateTime" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="ProbingSystem" type="ProbingSystemType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="Comment" type="xsd:string" maxOccurs="1" minOccurs="0">
        </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Record3Type">
      <xsd:sequence>
        <xsd:choice maxOccurs="1" minOccurs="1">
            <xsd:element name="MatrixDimension" maxOccurs="1" minOccurs="1"
type="MatrixDimensionType">
            </xsd:element>
            <xsd:element name="ListDimension" type="xsd:unsignedLong" maxOccurs="1"
minOccurs="1">
            </xsd:element>
        </xsd:choice>
        <xsd:choice maxOccurs="1" minOccurs="1">
            <xsd:element name="DataLink" type="DataLinkType" maxOccurs="1" minOccurs="1">
            </xsd:element>
            <xsd:element name="DataList" type="DataListType" maxOccurs="1" minOccurs="1">
            </xsd:element>
        </xsd:choice>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Record4Type">
      <xsd:sequence>
        <xsd:element name="ChecksumFile" type="xsd:string" maxOccurs="1" minOccurs="1">
```

```
      </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AxesType">
    <xsd:sequence>
      <xsd:element name="CX" type="AxisDescriptionType" maxOccurs="1" minOccurs="1">
      </xsd:element>
      <xsd:element name="CY" type="AxisDescriptionType" maxOccurs="1" minOccurs="1">
      </xsd:element>
      <xsd:element name="CZ" type="AxisDescriptionType" maxOccurs="1" minOccurs="1">
      </xsd:element>
      <xsd:element name="Rotation" type="RotationType" maxOccurs="1" minOccurs="0">
      </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AxisDescriptionType">
    <xsd:sequence>
      <xsd:element name="AxisType" maxOccurs="1" minOccurs="1">
          <xsd:simpleType>
            <xsd:restriction base="xsd:token">
              <xsd:enumeration value="A">
              </xsd:enumeration>
              <xsd:enumeration value="I">
              </xsd:enumeration>
              </xsd:restriction>
            </xsd:simpleType>
      </xsd:element>
      <xsd:element name="DataType" maxOccurs="1" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:token">
              <xsd:enumeration value="I">
              </xsd:enumeration>
              <xsd:enumeration value="L">
              </xsd:enumeration>
              <xsd:enumeration value="F">
              </xsd:enumeration>
              <xsd:enumeration value="D">
              </xsd:enumeration>
            </xsd:restriction>
          </xsd:simpleType>
      </xsd:element>
      <xsd:element name="Increment" type="xsd:double" maxOccurs="1" minOccurs="0">
      </xsd:element>
      <xsd:element name="Offset" type="xsd:double" maxOccurs="1" minOccurs="0">
      </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="InstrumentType">
    <xsd:sequence>
      <xsd:element name="Manufacturer" type="xsd:token" maxOccurs="1" minOccurs="1">
      </xsd:element>
      <xsd:element name="Model" type="xsd:token">
      </xsd:element>
      <xsd:element name="Serial" type="xsd:token">
      </xsd:element>
      <xsd:element name="Version" type="xsd:token">
      </xsd:element>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ProbingSystemType">
    <xsd:sequence>
      <xsd:element name="Type" maxOccurs="1" minOccurs="1">
        <xsd:simpleType>
          <xsd:restriction base="xsd:token">
            <xsd:enumeration value="Contacting">
            </xsd:enumeration>
            <xsd:enumeration value="NonContacting">
            </xsd:enumeration>
            <xsd:enumeration value="Software">
            </xsd:enumeration>
          </xsd:restriction>
```

```
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Identification" type="xsd:token" maxOccurs="1" minOccurs="1">
        </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="DataListType">
      <xsd:sequence>
        <xsd:element name="Datum" maxOccurs="unbounded" minOccurs="1">
            <xsd:simpleType>
              <xsd:restriction base="xsd:token">
                  <xsd:pattern value=
        "((-|\+)?(\d*\.?\d+)((e|E)(-|\+)?\d{1,4})?)?(;(-|\+)?(\d*\.?\d+)((e|E)
(-|\+)?\d{1,4})?)*">
                </xsd:pattern>
              </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="DataLinkType">
      <xsd:sequence>
        <xsd:element name="PointDataLink" type="xsd:string" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="MD5ChecksumPointData" type="xsd:hexBinary" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:sequence maxOccurs="1" minOccurs="0">
            <xsd:element name="ValidPointsLink" type="xsd:string" maxOccurs="1"
minOccurs="1">
            </xsd:element>
            <xsd:element name="MD5ChecksumValidPoints" type="xsd:hexBinary" maxOccurs="1"
                    minOccurs="1">
            </xsd:element>
        </xsd:sequence>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="MatrixDimensionType">
      <xsd:sequence>
        <xsd:element name="SizeX" type="xsd:unsignedLong" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="SizeY" type="xsd:unsignedLong" maxOccurs="1" minOccurs="1">
        </xsd:element>
        <xsd:element name="SizeZ" type="xsd:unsignedLong" maxOccurs="1" minOccurs="1">
        </xsd:element>
      </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="RotationType">
      <xsd:sequence>
        <xsd:element name="r11" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r12" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r13" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r21" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r22" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r23" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r31" type="RotationMatrixElementType" maxOccurs="1"
minOccurs="1">
        </xsd:element>
        <xsd:element name="r32" type="RotationMatrixElementType" maxOccurs="1"
```