SIEMENS

JT File Format Reference

Version 10.5

Rev-A

Contents

Page

Figures

Tables

1 Siemens JT Data Format Reference Intellectual Property License Terms

The general idea of using an interchange format for electronic documents is in the public domain. Anyone is free to devise a set of unique data structures and operators that define an interchange format for electronic documents. However, Siemens Product Lifecycle Management Software Inc. owns the copyright for the particular data structures and operators, the JT™ Data Format Reference and the written specification constituting the interchange format called the JT Data Format. Thus, these elements of the JT Data Format may not be copied without Siemens's permission.

Siemens will enforce its copyright. Siemens's intention is to maintain the integrity of the JT Data Format standard, enabling the public to distinguish between the JT Data Format and other interchange formats for electronic documents. However, Siemens desires to promote the use of the JT Data Format for information interchange among diverse products and applications. Accordingly, Siemens gives anyone copyright permission, subject to the conditions stated below, to:

Prepare and distribute files whose content conforms solely to the JT Data Format.

Write and distribute software applications that produce discreet output represented in the JT Data Format. Write and distribute software applications that accept input in the form of the JT Data Format and display, print, or otherwise interpret the contents

Copy Siemens's copyrighted list of data structures and operators in the written specification to the extent necessary to use the JT Data Format for the purposes above.

For avoidance of doubt, the permissions granted in the preceding sentences do not include the reading, writing or distribution of files whose content contains output in the JT Data Format and any other data in any other format and do not include the right to incorporate, integrate, or combine the JT Data Format, structure, or schema into any other data format, structure, or schema.

The conditions of such copyright permission are:

Anyone who uses the copyrighted list of data structures and operators, as stated above, must include an appropriate copyright notice.

This limited right to use the copyrighted list of data structures and operators does not include the right to copy this document, other copyrighted material from Siemens, or the software in any of Siemens's products that use the JT Data Format, in whole or in part, nor does it include the right to use any Siemens patents, except as may be permitted by an official Siemens JT Data Format Reference Patent Clarification Notice.

Nothing in this book is intended to grant you any right or license to use the Marks for any purpose.

2 Scope

This reference defines the syntax and semantics of the JT Version 10.5 file format.

The JT format is an industry focused, high-performance, lightweight, flexible file format for capturing and repurposing 3D Product Definition data that enables collaboration, validation and visualization throughout the extended enterprise. JT format is the de-facto standard 3D Visualization format in the automotive industry, and the single most dominant 3D visualization format in Aerospace, Heavy Equipment and other mechanical CAD domains.

The JT format is both robust, and streamable, and contains best-in-class compression for compact and efficient representation. The JT format was designed to be easily integrated into enterprise translation solutions, producing a single set of 3D digital assets that support a full range of downstream processes from lightweight web-based viewing to full product digital mockups.

At its core the JT format is a scene graph with CAD specific node and attributes support. Facet information (triangles), is stored with sophisticated geometry compression techniques. Visual attributes such as lights, textures, materials are supported. Product and Manufacturing Information (PMI), Precise Part definitions (B-Rep) and Metadata as well as a variety of representation configurations are supported by the format. The JT format is also structured to enable support for various delivery methods including asynchronous streaming of content.

Some of the highlights of the JT format include:

- Built-in support for assemblies, sub-assemblies and part constructs
- Flexible partitioning scheme, supporting single or multiple files
- B-Rep, including integrated support for industry standard Parasolid® (XT) format
- Product Manufacturing Information in support of paperless manufacturing initiatives
- Precise and imprecise wireframe
- Discrete purpose-built Levels of Detail
- Triangle sets, Polygon sets, Point sets, Line sets and Implicit Primitive sets (cylinder, cone, sphere, etc…)
- Full array of visual attributes: Materials, Textures, Lights
- Hierarchical Bounding Box and Bounding Spheres
- Advanced data compression that allows producers of JT files to fine tune the tradeoff between compression ratio and fidelity of the data.

Beyond the data contents description of the JT Format, the overall physical structure/organization of the format is also designed to support operations such as:

Offline optimizations of the data contents

- File granularity and flexibility optimized to meet the needs of Enterprise Data Translation Solutions

Asynchronous streaming of content

- Viewing optimizations such as view frustum and occlusion culling and fixed-framerate display modes.

Layers, and Layer Filters.

Along with the pure syntactical definition of the JT Format, there is also series of conventions which although not required to have a reference compliant JT file, have become commonplace within JT format translators. These conventions have been documented in the “Best Practices” section of this JT format reference.

This JT format reference does not specifically address implementation of, nor define, a run-time architecture for viewing and/or processing JT data. This is because although the JT format is closely aligned with a run-time data representation for fast and efficient loading/unloading of data, no interaction behavior is defined within the format itself, either in the form of specific viewer controls, viewport information, animation behavior or other event-based interactivity. This exclusion of interaction behavior from the JT format makes the format more easily reusable for dissimilar application interoperation and also facilitates incremental update, without losing downstream authored data, as the original CAD asset revises.

3 What’s New in This Revision

Revision A

The technical description for MbStrings in Table 4 Composite Data Types has been updated to include UTF 16

The UChar: Version string description from Figure 11 File Header data collection has been updated to reflect the JT version being 10.5

New segment types have been added to Table 6 “Segment Types”. The new types include; Smart Topolgy Table (STT), Info Segment and AEC Shape. The complete descriptions for STT is provided in Annex J. The complete descrption for Info Segment is added in Chapter 10. The AEC Shape data description is under development.

STEP B-rep now has the type number 33.

The loop for Texture Coordinate Generator Attribute Elements has been removed from Figure 20 LSG Segment data collection

The Base Attribute Data description found in 7.2 Attribute Elements has been expanded to include the logical collection Base Attribute Data Fields V2. This logical collection is used to support per face group attributes.

The logical collection Base Attribute Data Fields V2 has been added to the following attribute elements; 7.2.1 Material Attribute Element, Material Attribute Element, 7.2.2 Texture Image Attribute Element, 7.2.3 Draw Style Attribute Element, 7.2.4 Light Set Attribute Element, 7.2.5 Linestyle Attribute Element, 7.2.6 Pointstyle Attribute Element, 7.2.7 Geometric Transform Attribute Element, 7.2.8 Palette Map Attribute Element

A new attribute element has been added to the LSG definition, it is 6.2.8 Palette Map Attribute Element. A Palette Map Attribute is used on a shape such that any face group can be rendered with a chosen entry from the palette.

A new attribute element, “7.2.9 Sabot V104 Attribute Element”, has been added to the LSG definition. Sabot V104 is used to insulate pre ISO Edition 3 readers from attributes with non-fallback pallet Index attributes in order to preserve forward compatibility

In Section 10.2.2 PMI Associations, Table 55 PMI Associations Source Data values has new values added; =20 B-rep body and =23 Group. Table 56 — PMI Associations Reason Code values has two new values =20 Association is to a PMI B-rep entity contained in a virtual group. Similar to reason code 14 but for a virtual group and =78 Association is to a product instance to cut by a PMI section. Only required for partial scene sectioning.

The Table “Common Property Keys and Their Value Encoding formats” has been removed from the PMI Property descriptions in section 10.2.5 PMI Model Views. A complete description of PMI properties is provided in Annex Q

Common Property Keys and Their Value Encoding formats

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Decoding Notes
“PMI_PROP_ANCHOR_POINT"	“Px Py Pz”	Each Px, Py, Pz is a F32 value using “%f” format
“PMI_PROP_NOTE_HAS_URL”	“0” or “1”	0 = = False; 1 = = True
“PMI_PROP_NORMAL_DIR”	“Dx Dy Dz”	Each Dx, Dy, Dz is a F32 value using “%f” format
“PMI_PROP_APPROACH_DIR”	“Dx Dy Dz”	Each Dx, Dy, Dz is a F32 value using “%f” format
“PMI_PROP_CLAMPING_DIR”	“Dx Dy Dz”	Each Dx, Dy, Dz is a F32 value using “%f” format
“PMI_PROP_MEAS_DIR”	“Dx Dy Dz”	Each Dx, Dy, Dz is a F32 value using “%f” format
“PMI_PROP_COORD_DIR”	“Dx Dy Dz”	Each Dx, Dy, Dz is a F32 value using “%f” format
“PMI_PROP_MEAS_LEVEL”	“#”	Integer representing level number

In section 10.2.6 Generic PMI Entities, Table 59 Generic PMI Entity Type values has a new value; 0x0309 Weld Note Type

Annex E Per Face Group Attributes has been added. This provides the description for the applying attributes, such as material and texture image, to a group of faces in the Logical Scene Graph (LSG).

A new section called “Intersection” has been added to Annex F “XT B-Rep data segment. It provides the description for an intersection curve node. An intersection curve is one of the branches of a surface / surface intersection. XT represents these curves exactly; the information held in an intersection curve node is sufficient to identify the particular intersection branch involved, to identify the behaviour of the curve at its ends, and to evaluate precisely at any point in the curve

Annex N, JT PMI Properties, has been updated to include tables of properties for each PMI Type

4 Terms, definitions and abbreviated terms

4.1 Terms and definitions

For the purposes of this document, the following terms apply.

3.1.1

Assembly

A related collection of model parts, represented in a JT format logical scene graph as a logical graph branch.

3.1.2

Attribute

Objects associated with nodes in a logical scene graph and specifying one of several appearances, positioning, or visual characteristics of a shape

3.1.3

Boundary representation

Solid model representation where the solid volume is specified by its surface boundary (both its geometric and topological boundaries)

3.1.4

Code text

Collection of data in encoded form

3.1.5

Coordinate system

A system which uses one or more numbers, or coordinates, to uniquely determine the position of a point or other geometric element

NOTE 1 If not otherwise specified in a data field's description, it is assumed that the data is defined in Local Coordinate System.

3.1.6

Directed acyclic graph

Graph that consists of a set of nodes and a set of edges that connect the nodes in a tree like structure

NOTE 1 A directed graph is one in which every edge has a direction such that edge (u,v), connecting node-u with node-v, is different from edge (v,u).

NOTE 2 A directed acyclic graph is a directed graph with no cycles, where a cycle is a path (sequence of edges) from a node to itself.

NOTE 3 With a directed acyclic graph, there is no path that can be followed within the graph such that the first node in the path is the same as the last node in the path.

3.1.7

JT enabled application

Application which supports reading and/or writing reference compliant JT format files

3.1.8

Level of detail

LOD

Alternative graphical representation for some model component such as part

3.1.9

Local coordinate system

LCS

Coordinate system that is used to specify the raw data of the shape geometry with no transforms applied

3.1.10

Logical scene graph

LSG

Scene graph representing the logical organization of a model

NOTE A scene graph contains shapes and attributes representing the model's physical components, properties identifying arbitrary metadata (e.g. names, semantic roles) of those components, and a hierarchical structure expressing the component relationships.

3.1.11

Mipmap

Reduced resolution version of a texture map

NOTE Mipmaps are used to texture a geometric primitive whose screen resolution differs from the resolution of the source texture map originally applied to the primitive.

3.1.12

Model

Representation, in JT format, of a physical or virtual product, part, assembly; or collections of such objects

3.1.13

Model coordinate system

MCS

Local coordinates transformed by any transforms specified as attributes at or above the node

3.1.14

Product and manufacturing information

PMI

Collection of information created on a 3D/2D CAD model to completely document the product with respect to design, manufacturing, inspection, etc.

NOTE This can include data such as:

— dimensions (tolerances for each dimension);

— geometric tolerances of feature (datums, feature control frames);

— manufacturing information (surface finish, welding notations);

— inspection information (key locations points);

— assembly instructions;

— product information (materials, suppliers, part numbers).

3.1.15

Property

Object associated with a logical scene graph node and identifying arbitrary application or enterprise specific information (meta-data) related to that node

3.1.16

Quantize

Constrain something to a discrete set of values, such as an integer or integral multiplier of a common factor, rather than a continuous set of values, such as a real number

3.1.17

Scene graph

Directed acyclic graph that arranges the logical and often (but not necessarily) spatial representation of a graphical scene

3.1.18

Streaming

Loading from disk based medium only the portions of data that are required by the user to perform the tasks at hand

NOTE 1 The motivation for streaming is to more efficiently manage system memory.

NOTE 2 Transfer of data in a stream of packets, over the internet on an on-demand basis, where the data is interpreted in real-time by the application as the data packets arrive.

NOTE 3 The motivation for streaming is that the user can begin using or interacting with the data almost immediately - no waiting for the entire data file(s) to be transferred before beginning.

NOTE 4 The desired end result of streaming is to deliver only the JT data that the user needs, where the user needs it, when the user needs it.

3.1.19

Shape

Logical scene graph leaf node containing or referencing the geometric shape definition data (such as vertices, polygons and normals) of a model component

3.1.20

Texture channel

Texture unit plus the texture environment.

NOTE The JT format meaning for texture channel is the same as in OpenGL [1].

3.1.21

Texture object

Named cache that stores texture data, such as the image array, associated mipmaps, and associated texture parameter values: width, height, border width, internal format, resolution of components, minification and magnification filters, wrapping modes, border colour, and texture priority

NOTE The JT format meaning for texture object is the same as in OpenGL [1].

3.1.22

Texture unit

A hardware unit used to sample and filter a texture image.

NOTE The JT format meaning for texture unit is the same as in OpenGL [1].

3.1.23

View coordinate system

World coordinates transformed by a view matrix

3.1.24

World coordinate system

WCS

Node coordinates transformed by transforms inherited from a node's parent (i.e. the coordinate system at the root of the graph)

4.2 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

Abs	Absolute Value
Bbox	Bounding Box
B-Rep	Boundary Representation

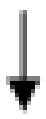
CAD	Computer Aided Design
CODEC	Coder-Decoder
GD&T	Geometric Dimensioning and Tolerancing
GUID	Globally Unique Identifier
HSV	Hue, Saturation, Value
LsbFirst	Least Significant Byte First
Max	Maximum
Min	Minimum
MsbFirst	Most Significant Byte First
N/A	Not Applicable
PCS	Parameter Coordinate Space
PLM	Product Lifecycle Management
RGB	Red, Green, Blue
RGBA	Red, Green, Blue, Alpha
TOC	Table of Contents
VPCS	Viewpoint Coordinate System
URL	Uniform Resource Locator

5 Notational conventions

5.1 Diagrams and field descriptions

Symbolic diagrams are used to describe the structure of the JT file. The symbols used in these diagrams have the following meaning:

Table 1 — Symbols

Symbol	Description
	Rectangles represent a data field of one of the standard data types.
	Folders represent a logical collection of one or more of the standard data types. This information is grouped for clarity and the basic data types that compose the group are detailed in following sections of the document.
	Rectangles with extra lines at left and the right sides corners clipped off represent information logical steps that has been compressed.
	Rectangles with the right side corners clipped off represent information that has been compressed.
	Arrows convey the ordering of the information.

The format used to title the diagram symbols is dependent upon the symbol type as follows:

Diagram "rectangle box" (i.e. standard data types) symbols are titled using a format of "Data_Type : Field_Name." The Data_Type is an abbreviated data type symbol as defined in 3.2 Data Types. In the example below the Data_Type is "I32" (a signed 32 bit integer) and Field_Name is "Count."

Figure 1 — rectangle box diagram

Diagram "folder" (i.e. logical data collections) symbols are simply titled with a collection name. In the example below the collection name is "Graph Elements."

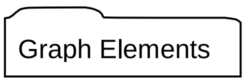


Figure 2 — folder diagram

Diagram "rectangle box with lines at left and right sides" are simply titled with a logic step name. In the example below the logic step name is "Recover First Shell Indices".

Figure 3 — rectangle box with lines at left and right sides diagram

Diagram “rectangle box with clipped right side corners” (i.e. compressed/encoded data fields) are titled using one of the following three formats:

Data Type; followed by open brace “{“, number of bits used to store value, closed brace “}”, and a colon “:”; followed by the Field Name. This format for titling the diagram symbol indicates that the data is compressed but not encoded. The compression is achieved by using only a portion of the total bit range of the data type to store the value (e.g. if a count value can never be larger than the value “63” then only 6 bits are needed to store all possible count values). In the example below the Data Type is “U32”, “6” bits are used to store the value, and Field Name is “Count”

Figure 4 — rectangle box with clipped right side corners

Data Type followed by open brace “{“, compressed data packet type, “,”, Predictor Type, closed brace “}”, and a colon “:”; followed by the field name. This format for titling the diagram indicates that a vector of “Data Type” data (i.e. *primal* values) is ran through “Predictor Type” algorithm and the resulting output array of *residual* values is then compressed and encoded into a series of symbols using one of the two supported compressed data packet types.

The two supported compressed data packet types are:

Int32CDP – The Int32CDP (i.e. Int32 Compressed Data Packet) represents a third-generation format used to encode/compress a collection of data into a series of Int32 based symbols. This version of the Int32CDP supersedes the two similarly-named ones from the Version 9 JT Specification, and should not be confused with either of its predecessors. A complete description for Int32 Compressed Data Packet can be found in Int32 Compressed Data Packet.

The Int32 Compressed Data Packet type is used for compressing/encoding both “integer” and “float” (through quantization) data.

In the example below the Data Type is “VecU32”, Int32 Compressed Data Packet type is used, Lag1 Predictor Type is used, and Field Name is “First Shell Index.”

Figure 5 — compressed data packet diagram

As mentioned above (with Predictor Type algorithm), the *primal* input data values are NOT always what is encoded/compressed. This is because the *primal* input data is first run through a Predictor Type algorithm, which produces an output array of residual values (i.e. difference from the predicted value), and this resulting output array of *residual* values is the data which is actually encoded/compressed. The JT format supports several Predictor Type algorithms and each use of Int32CDP specifies, using the above described notation format, what Predictor Type algorithm is being used on the data. The JT format supported Predictor Type algorithms are as follows (note that a sample implementation of decoding the predictor *residual* values back into the primal values can be found in Annex C.

Table 2 — Predictor Type

Predictor Type	Description
Lag1	Predicts as last value
Xor1	Predict as last, but use XOR instead of subtract to compute residual
NULL	No prediction applied

Each predictor type can be combined with additional processing steps, and in such case the predictor type is prefixed with “Combined:”. For example, “Combined:Lag1” means that predictor type “Lag1” is combined with additional preprocessing steps. Additional description about the processing steps is provided whenever such combined predictor is used.

“Data Type : Field Name” . This format for titling the diagram symbol indicates that the data is both compressed and encoded. The Data_Type is an abbreviated data type symbol as defined in Data Types and usually represent a vector/array of data. How the data is compressed and encoded into the Data Type is indicated by a CODEC type and other information stored before the particular data in the file. In the example below the Data_Type is “VecU32” and Field_Name is “CodeText.”

VecU32 : CodeText

Figure 6 — data type : field name diagram

Note that for some JT file Segment Types there is LZMA compression also applied to all bytes of element data stored in the segment. This LZMA compression applied to all the segment’s data is not indicated in the diagrams through the use of “rectangle box with clipped right side corners”. Instead, one shall examine information stored with the first Element in the file segment to determine if LZMA compression is applied to all data in the segment. A complete description of the JT format data compression and encoding can be found in Data Segment and Data Compression and Encoding.

Following each data collection diagram is detailed descriptions for each entry in the data diagram.

For rectangles this detail includes the abbreviated data type symbol, field name, verbal data description, and compression technique/algorithm where appropriate. If the data field is documented as a collection of flags, then the field is to be treated as a bit mask where the bit mask is formed by combining the flags using the binary OR operator. Each bits usage is documented, and bit ON indicates flag value is TRUE and bit OFF indicates flag value is FALSE. All bits fields that are not defined as in use should be set to “0”.

For folders (i.e. data collections), if the collection is not detailed under a sub-section of the particular document section referencing the data collection, then a comment is included following the diagram indicating where in the document the particular data collection is detailed.

If an arrow appears with a branch in its shaft, then there are two or more options for data to be stored in the file. Which data is stored will depend on information previously read from the file. The following example shows data field A followed by (depending on value of A) either data field B, C, or D.

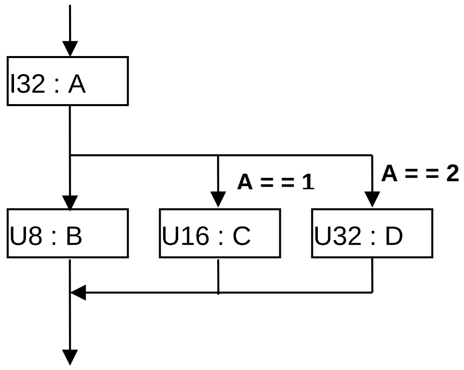


Figure 7 — data filed dependency example

In cases where the same data type repeats, a loop construct is used where the number of iterations appears next to the loop line. There are two forms of this loop construct. The first form is used when the number of iterations is not controlled by some previous read count value. Instead the number of iterations is either a hard coded count (e.g. always 80 characters) or is indicated by some end-of-list marker in the data itself (thus the count is always minimum of 1). This first form of the loop construct looks as follows:

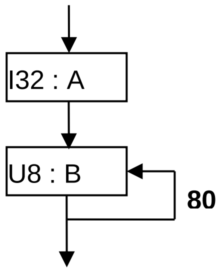


Figure 8 — loop construct example

The second form of this loop construct is used when the number of iterations is based on data (e.g. count) previously read from the file. In this case it is valid for there to be zero data iterations (zero count). This second from of the loop construct looks as follows (data field D is repeated C value times).

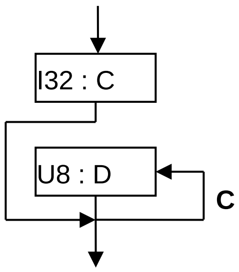


Figure 9 — loop construct with iterations example

5.2 Data Types

The data types that can occur in the JT binary files are listed in the following two tables.

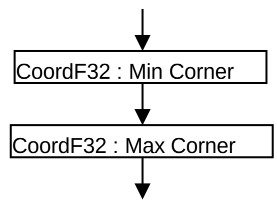
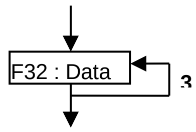
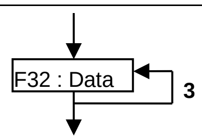
Table 3 — Basic Data Types lists the basic/standard data types which can occur in JT file.

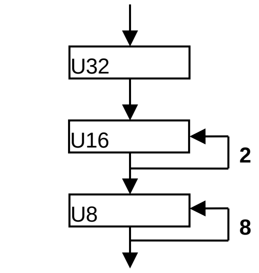
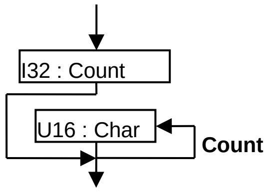
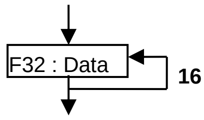
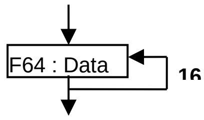
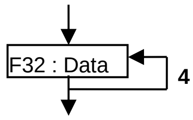
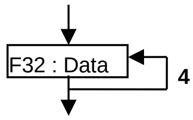
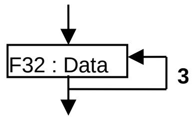
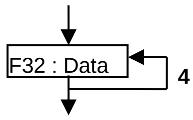
Table 3 — Basic Data Types

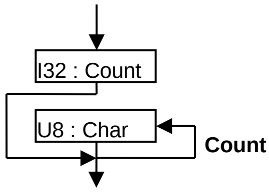
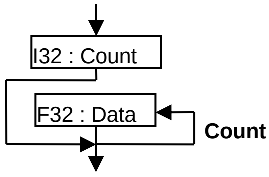
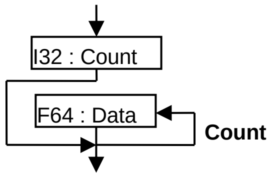
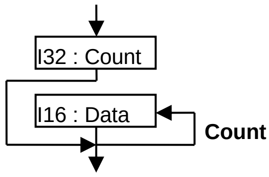
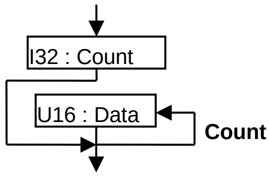
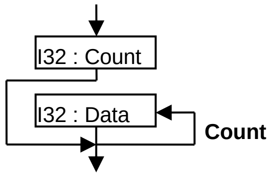
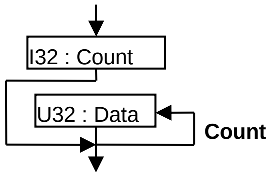
Type	Description
UChar	An unsigned 8-bit byte.
U8	An unsigned 8-bit integer value.
U16	An unsigned 16-bit integer value.
U32	An unsigned 32-bit integer value.
U64	An unsigned 64-bit integer value.
I16	A signed two's complement 16-bit integer value.
I32	A signed two's complement 32-bit integer value.
I64	A signed two's complement 64-bit integer value.
F32	An IEEE 32-bit floating point number.
F64	An IEEE 64-bit double precision floating point number

Table 4 — Composite Data lists some composite data types which are used to represent some frequently occurring groupings of the basic data types (e.g. Vector, RGBA colour). The composite data types are defined in this reference simply for convenience/brevity in describing the JT file contents.

Table 4 — Composite Data Types

Type	Description	Symbolic Diagram
BBoxF32	The BBoxF32 type defines a bounding box using two CoordF32 types to store the XYZ coordinates for the bounding box minimum and maximum corner points.	
CoordF32	The CoordF32 type defines X, Y, Z coordinate values. So a CoordF32 is made up of three F32 base types.	
DirF32	The DirF32 type defines X, Y, Z components of a direction vector. So a DirF32 is made up of three F32 base types.	

GUID	The GUID type is a 16 byte (128-bit) number. GUID is stored/written to the JT file using a four-byte word (U32), 2 two-byte words (U16), and 8 one-byte words (U8) such as: {3F2504E0-4F89-11D3-9A-0C-03-05-E8-2C-33-01} In the JT format GUIDs are used as unique identifiers (e.g. Data Segment ID, Object Type ID, etc.)	
MbString	The MbString type starts with an I32 that defines the number of characters (NumChar) the string contains. The number of bytes of character data is "2 * NumChar" (i.e. the strings are written out as multi-byte characters where each character is UTF16 size).	
Mx4F32	Defines a 4-by-4 matrix of F32 values for a total of 16 F32 values. The values are stored in row major order (right most subscript, column varies fastest), that is, the first 4 elements form the first row of the matrix.	
Mx4F64	Defines a 4-by-4 matrix of F64 values for a total of 16 F64 values. The values are stored in row major order (right most subscript, column varies fastest), that is, the first 4 elements form the first row of the matrix.	
PlaneF32	The PlaneF32 type defines a geometric Plane using the General Form of the plane equation (Ax + By + Cz + D = 0). The PlaneF32 type is made up of four F32 base types where the first three F32 define the plane unit normal vector (A, B, C) and the last F32 defines the negated perpendicular distance (D), along normal vector, from the origin to the plane.	
Quaternion	The Quaternion type defines a 3-dimensional orientation (no translation) in quaternion linear combination form (a + b*i* + c*j* + d*k*) where the four scalar values (a, b, c, d) are associated with the 4 dimensions of a quaternion (1 real dimension, and 3 imaginary dimensions). So the Quaternion type is made up of four F32 base types.	
RGB	The RGB type defines a colour composed of Red, Green, Blue components, each of which is a F32. So a RGB type is made up of three F32 base types. The Red, Green, Blue colour values typically range from 0.0 to 1.0.	
RGBA	The RGBA type defines a colour composed of Red, Green, Blue, Alpha components, each of which is a F32. So a RGBA type is made up of four F32 base types. The Red, Green, Blue colour values typically range from 0.0 to 1.0. The Alpha value ranges from 0.0 to 1.0 where 1.0 indicates completely opaque.	

String	The String type starts with an I32 that defines the number of characters (NumChar) the string contains. The number of bytes of character data is “NumChar” (i.e. the strings are written out as single-byte characters where each character is U8 size).	
VecF32	The VecF32 type defines a vector/array of F32 base type. The type starts with an I32 that defines the count of following F32 base type data. So a VecF32 is made up of one I32 followed by that number of F32. Note that it is valid for the I32 count number to be equal to “0”, indicating no following F32.	
VecF64	The VecF64 type defines a vector/array of F64 base type. The type starts with an I32 that defines the count of following F64 base type data. So a VecF64 is made up of one I32 followed by that number of F64. Note that it is valid for the I32 count number to be equal to “0”, indicating no following F64.	
VecI16	The VecI16 type defines a vector/array of I16 base type. The type starts with an I32 that defines the count of following I16 base type data. So a VecI16 is made up of one I32 followed by that number of I16. Note that it is valid for the I32 count number to be equal to “0”, indicating no following I16.	
VecU16	The VecU16 type defines a vector/array of U16 base type. The type starts with an I32 that defines the count of following U16 base type data. So a VecU16 is made up of one I32 followed by that number of U16. Note that it is valid for the I32 count number to be equal to “0”, indicating no following U16.	
VecI32	The VecI32 type defines a vector/array of I32 base type. The type starts with an I32 that defines the count of following I32 base type data. So a VecI32 is made up of one I32 followed by that number of I32. Note that it is valid for the I32 count number to be equal to “0”, indicating no following I32.	
VecU32	The VecU32 type defines a vector/array of U32 base type. The type starts with an I32 that defines the count of following U32 base type data. So a VecU32 is made up of one I32 followed by that number of U32. Note that it is valid for the I32 count number to be equal to “0”, indicating no following U32.	

5.3 Empty field

When writing a JT file whose data did not originate from reading a previous JT file, an empty field should be set to a value of “0”.

When writing a JT file whose data originated from reading a previous JT file (i.e. rewriting a JT File), empty fields should be written with the same value that was read from the originating JT file.

Refer to Empty Field guidines in the Common Data Conventions and Constructs section

6 File Format

All objects represented in the JT format are assigned an “object identifier” (e.g. see Base Node Data, or Base Attribute Data Base Attribute Data) and all references from one object to another object are represented in the JT format using the referenced object’s “object identifier”. It is the responsibility of JT format readers/writers to maintain the integrity of these object references by doing appropriate pointer unswizzling/swizzling as JT format data is read into memory or written out to disk. Where “pointer swizzling” refers to the process of converting references based on object identifiers into direct memory pointer references and “pointer unswizzling” is the reverse operation (i.e. replacing references based on memory pointers with object identifier references).

6.1 File Structure

A JT file is structured as a sequence of blocks/segments. The File Header block is always the first block of data in the file. The File Header is followed (in no particular order) by a TOC Segment and a series of other Data Segments. The one Data Segment which shall always exist to have a reference compliant JT file is the LSG Segment. The TOC Segment is located within the file using data stored in the File Header. Within the TOC Segment is information that locates all other Data Segments within the file.

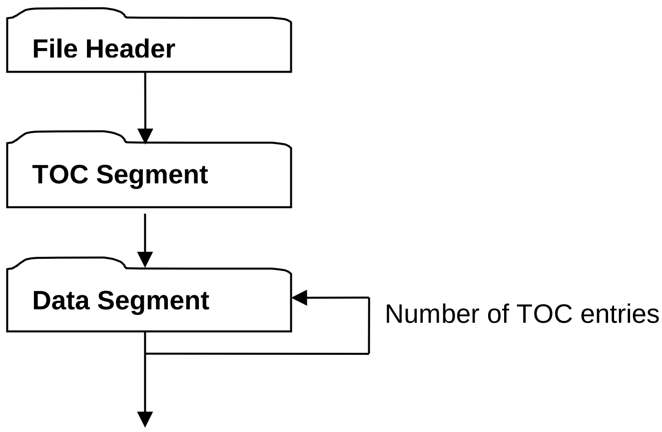


Figure 10 — JT File Structure

6.1.1 File Header

The File Header is always the first block of data in a JT file. The File Header contains information about the JT file version and TOC location, which Loaders use to determine how to read the file. The exact contents of the File Header are as follows:

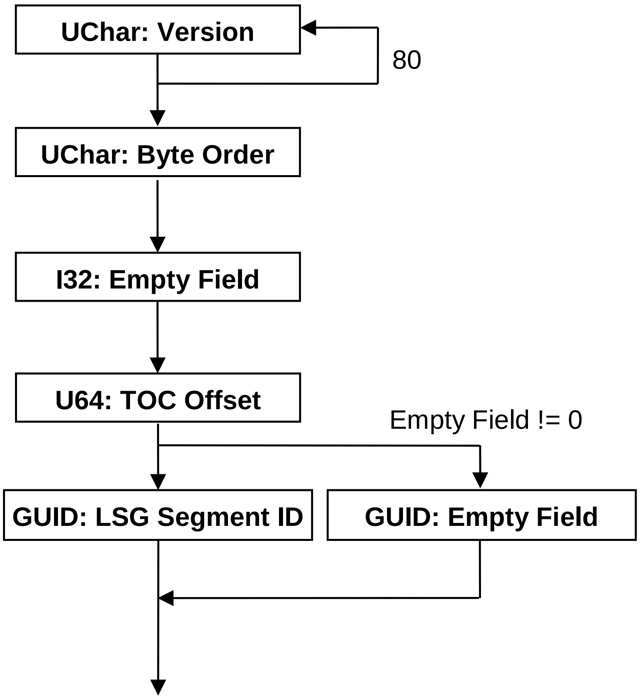


Figure 11 — File Header data collection

UChar: Version

An 80-character version string defining the version of the file format used to write this file. The Version string has the following format:

Version M.n Comment

Where ***M*** is replaced by the major version number, ***n*** is replaced by the minor version number, and ***Comment*** provides other information.

The major.minor version description for ISO JT Edition 3 files is 10.5

The version string is padded with spaces to a length of 75 ASCII characters and then the final five characters shall be filled with the following linefeed and carriage return character combination (shown using c-style syntax):

```
Version[75] = ' '
Version[76] = '\n'
Version[77] = '\r'
Version[78] = '\n'
Version[79] = ' '
```

These final 5 characters (shown above and referred to as ASCII/binary translation detection bytes) can be used by JT file readers to validate that the JT files has not been corrupted by ASCII mode FTP transfers.

As an example, for an ISO edition three file this string will look as follows:

"Version 10.5   \n\r\n "

UChar: Byte Order

Defines the file byte order and thus can be used by the loader to determine if there is a mismatch (thus byte swapping required) between the file byte order and the machine (on which the loader is being run) byte order. Valid values for Byte Order are:

0 – Least Significant byte first (LsbFirst)

1 – Most Significant byte first (MsbFirst)

I32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field description.

U64: TOC Offset

Defines the byte offset from the top of the file to the start of the TOC Segment.

GUID: LSG Segment ID

LSG Segment ID specifies the globally unique identifier for the Logical Scene Graph Data Segment in the file. This ID along with the information in the TOC Segment can be used to locate the start of LSG Data Segment in the file. This ID is needed because without it a loader would have no way of knowing the location of the root LSG Data Segment. All other Data Segments shall be accessible from the root LSG Data Segment.

GUID: Empty Field

Refer to Common Data Conventions and Constructs Empty Field description.

6.1.2 TOC Segment

The TOC Segment contains information identifying and locating all individually addressable Data Segments within the file. A TOC Segment is always required to exist somewhere within a JT file. The actual location of the TOC Segment within the file is specified by the File Header segment's "TOC Offset" field. The TOC Segment contains one TOC Entry for each individually addressable Data Segment in the file.

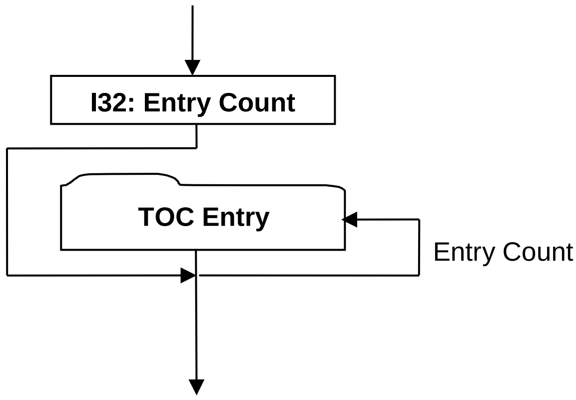


Figure 12 — TOC Segment data collection

I32: Entry Count

Entry Count is the number of entries in the TOC.

TOC Entry

Each TOC Entry represents a Data Segment within the JT File. The essential function of a TOC Entry is to map a Segment ID to an absolute byte offset within the file.

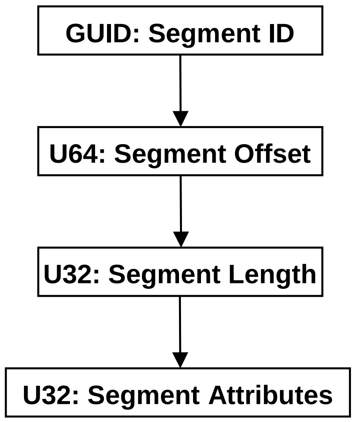


Figure 13 — TOC Entry data collection

GUID: Segment ID

Segment ID is the globally unique identifier for the segment.

U64: Segment Offset

Segment Offset defines the byte offset from the top of the file to start of the segment.

U32: Segment Length

Segment Length is the total size of the segment in bytes.

U32: Segment Attributes

Segment Attributes is a collection of segment information encoded within a single U32 using the following bit allocation.

Table 5 — Segment attributes

Bits 0 - 23	Reserved for future use.
Bits 24 - 31	Segment type. Complete list of Segment types can be found in **Table 6— Segment Types**.

6.1.3 Data Segment

All data stored in a JT file shall be defined within a Data Segment. Data Segments are “typed” based on the general classification of data they contain. See Segment Type field description below for a complete list of the segment types.

Beyond specific data field compression/encoding, some Data Segment types also have compression conditionally applied to all the Data bytes of information persisted within the segment. Whether this compression is conditionally applied to a segment’s Data bytes of information is indicated by information stored with the first “Element” in the segment. Also, **Table 6 Segment Types** has a column indicating whether the Segment Type may have compression applied to its Data bytes.

All Data Segments have the same basic structure.

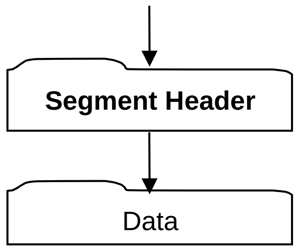


Figure 14 — Data Segment data collection

Segment Header

Segment Header contains information that determines how the remainder of the Segment is interpreted by the loader.

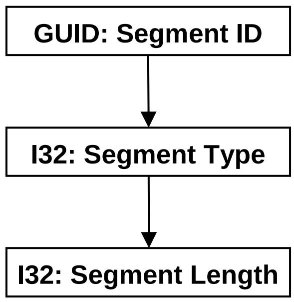


Figure 15 — Segment Header data collection

GUID: Segment ID

Global Unique Identifier for the segment.

I32: Segment Type

Segment Type defines a broad classification of the segment contents. For example, a Segment Type of “1” denotes that the segment contains Logical Scene Graph material; “2” denotes contents of a B-Rep, etc.

The complete list of segment and whether or not they support compression on all Data bytes in the payload is as follows: payload is as follows:

Table 6 — Segment Types

Type	Data Contents	Compression
1	Logical Scene Graph	Yes
2	JT B-Rep	Yes
3	PMI Data	Yes
4	Meta Data	Yes
6	Shape	No
7	Shape LOD0	No
8	Shape LOD1	No
9	Shape LOD2	No
10	Shape LOD3	No
11	Shape LOD4	No

Type	Data Contents	Compression
12	Shape LOD5	No
13	Shape LOD6	No
14	Shape LOD7	No
15	Shape LOD8	No
16	Shape LOD9	No
17	XT B-Rep	Yes
18	Wireframe Representation	Yes
20	ULP	Yes
23	STT	Yes
24	LWPA	Yes
30	MultiXT B-Rep	Yes
31	InfoSegment	Yes
32	AEC Shape	Yes
33	STEP B-rep	Yes

NOTE 1 Segment Types 7-16 all identify the contents as LOD Shape data, where the increasing type number is intended to convey some notion of how high an LOD the specific shape segment represents. The lower the type in this 7-16 range the more detailed the Shape LOD (i.e. Segment Type 7 is the most detailed Shape LOD Segment). For the rare case when there are more than 10 LODs, LOD9 and greater are all assigned Segment Type 16.

NOTE 2 The more generic Shape Segment type (i.e. Segment Type 6) is used when the Shape Segment has one or more of the following characteristics:

— not a descendant of an LOD node;

— is referenced by (i.e. is a child of) more than one LOD node;

— Shape has its own built-in LODs;

— no way to determine what LOD a Shape Segment represents.

I32: Segment Length

Segment Length is the total size of the segment in bytes. This length value includes all segment Data bytes plus the Segment Header bytes (i.e. it is the size of the complete segment) and should be equal to the length value stored with this segment’s TOC Entry.

Data

The interpretation of the Data section depends on the Segment Type. See Data Segments for complete description for all Data Segments that may be contained in a JT file.

Although the Data section is Segment Type dependent there is a common structure which often occurs within the Data section. This structure is a list or multiple lists of Elements where each Element has the same basic structure which consists of some fixed length header information describing the type of object contained in the Element, followed by some variable length object type specific data.

Individual data fields of an Element data collection (and its children data collections) may have advanced compression/encoding applied to them as indicated through compression related data values stored as part of the particular Element’s storage format. In addition, another level of compression (i.e. LZMA compression) may be conditionally applied to all bytes of information stored for all Elements within a particular Segment. Not all Segment types support compression on all

Segment data as indicated Table 8 — Segment Types If a particular file Segment is of the type which supports compression on all the Segment data, whether this compression is applied or not is indicated by data values stored in the Logical Element Header Compressed data collection of the first Element within the Segment. An in-depth description of JT file compression/encoding techniques can be found in this document.

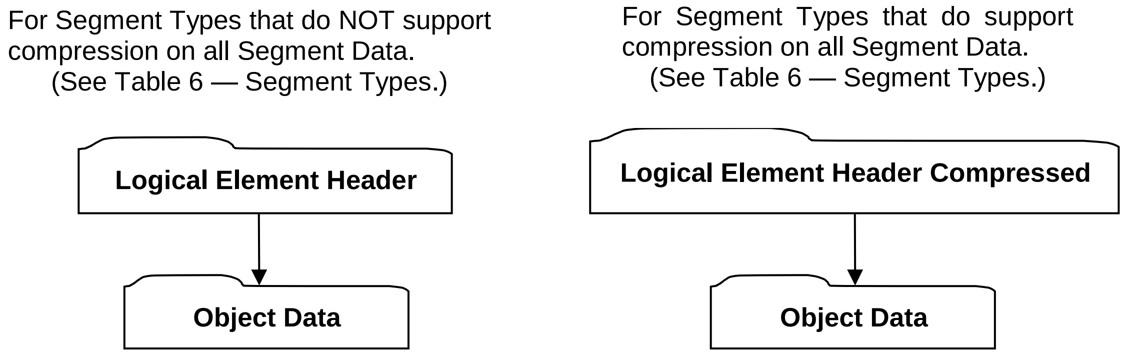


Figure 16 — Data collection

Logical Element Header

Logical Element Header contains data defining the length in bytes of the Element along with the Element Header.

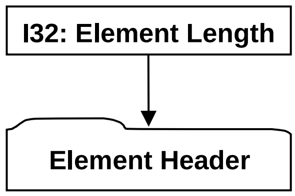


Figure 17 — Logical Element Header data collection

Complete description for Logical Element Header can be found in the Element Header.

I32: Element Length

Element Length is the total length in bytes of the element Object Data.

Element Header

Element Header contains data describing the object type contained in the Element.

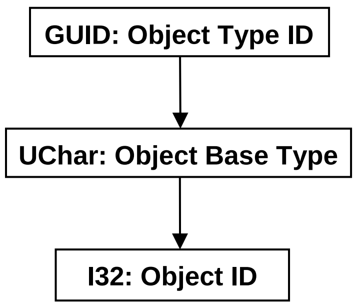


Figure 18 — Element Header data collection

GUID: Object Type ID

Object Type ID is the globally unique identifier for the object type. A complete list of the assigned GUID for all object types stored in a JT file can be found in Annex A. If the GUID is not found in Annex A, the reader should skip Element Length + 1 number of bytes.

UChar: Object Base Type

Object Base Type identifies the base object type. If the Object Base Type is not present in Object Base Types table then the loader should simply skip (read pass) Element Length number of bytes.

Valid Object Base Types include the following:

Table 7 — Object Base Types

Base Type	Description	Base Type's Data Format
255	None	None
0	Base Graph Node Object	Base Node Data
1	Group Graph Node Object	Group Node Data
2	Shape Graph Node Object	Base Shape Data
3	Base Attribute Object	Base Attribute Data
4	Shape LOD	None
5	Base Property Object	Base Property Atom Data
6	JT Object Reference Object	JT Object Reference Property Atom Element without the Logical Element Header Compressed data collection.
8	JT Late Loaded Property Object	Late Loaded Property Atom Element without the Logical Element Header Compressed data collection.
9	JtBase (none)	None

I32: Object ID

Object ID is the identifier for this Object. Other objects referencing this particular object do so using the Object ID.

Object Data

The interpretation of the Object Data section depends upon the Object Type ID stored in the Logical Element Header (see Logical Element Header).

Logical Element Header Compressed

Logical Element Header Compressed data collection is the format of Element Header data used by all Elements within Segment Types that support compression on all data in the Segment. See the Segment Types table for information on whether a particular Segment Type supports compression on all data in the Segment.

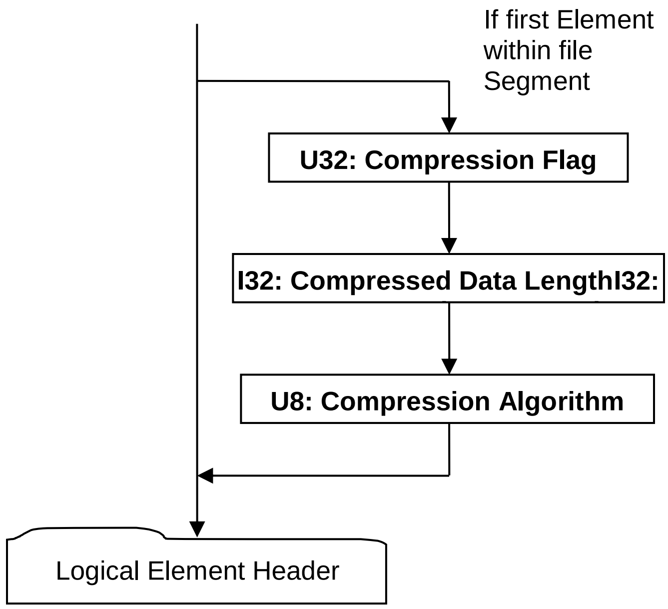


Figure 19 — Logical Element Header Compressed data collection

Complete description for Logical Element Header can be found in Logical Element Header. Note that if Compression Flag indicates that compression is ON for all element data in the Segment, then the Logical Element Header data collection is also compressed accordingly.

U32: Compression Flag

Compression Flag is a flag indicating whether compression is ON/OFF for all data elements in the file Segment. Valid values include the following:

Table 8 — Compression flag values

= 3	LZMA compression is ON
!= 3	LZMA No Compression

I32: Compressed Data Length

Compressed Data Length specifies the compressed data length in number of bytes. Note that data field Compression Algorithm is included in this count.

U8: Compression Algorithm

Compression Algorithm specifies the compression algorithm applied to all data in the Segment. Valid values include the following:

Table 9 — Compression algorithm values

= 1	No compression
= 3	LZMA compression

Logical Element Header

Object Data

The interpretation of the Object Data section depends upon the Object Type ID stored in the Logical Element Header (see Logical Element Header).

6.2 Data Segments

A JT file may consists of the following segments of data:

- LSG segment contains a collection of objects (i.e. elements) connected through directed references to form a directed acyclic graph structure (i.e. the LSG). The LSG is the graphical description of the model and contains graphics shapes and attributes representing the model's physical components, properties identifying arbitrary metadata (e.g. names, semantic roles) of those components, and a hierarchical structure expressing the component relationships.

- Meta Data Segment is used to store large collections of meta-data in separate addressable segments of the JT File, including PMI. Storing meta-data in a separate addressable segment allows references (from within the JT file) to these segments to be constructed such that the meta-data can be late-loaded.

- Shape LOD segment contains an element that defines the geometric shape definition data (e.g. vertices, polygons, normals, etc) for a particular shape level of detail or alternative representation.

- XT B-Rep Segment contains an element that defines the recommended precise geometric boundary representation data for a particular part in boundary representation format.

- Wireframe Segment contains an element that defines the precise 3D wireframe data for a particular part.

- JT ULP Segment contains an element that defines the semi-precise geometric boundary representation data for a particular part in JT ULP format.

- JT LWPA Segment contains an element that defines light weight precise analytic data for a particular part. More specifically LWPA contains the collection of analytic surfaces in the b-rep definition of the part.

- Multi XT B-Rep Segment contains an element that defines the precise geometric boundary representation data for one or more parts in boundary representation format.

- Info Segment : contains text strings with authoing information for the JT file it exists in.

- STT Segment JT Smart Topology Table (hereafter referred to as STT) Segment contains an Element that defines the lightweight B-Rep description for a particular Part.

The following segments are deprecated as of this specification description; they are included here for completeness but should be considered to be read only for legacy purposes:

- JT B-Rep Segment containing an element that defines the precise geometric boundary representation data for a particular Part in JT B-Rep format;

- The PMI manager meta data element can sometimes also be represented in a PMI data segment. This can occur when a pre JT 8 version file is migrated to JT 9.5 version file. So from a parsing point of view a PMI data segment should be treated exactly the same as a meta data segment.

For completed information on all the segments of a JT file see the segment descriptions for each segment in their specific sections in this document.

7 LSG Segment

LSG Segment contains a collection of objects (i.e. Elements) connected through directed references to form a directed acyclic graph structure (i.e. the LSG). The LSG is the graphical description of the model and contains graphics shapes and attributes representing the model's physical components, properties identifying arbitrary metadata (e.g. names, semantic roles) of those components, and a hierarchical structure expressing the component relationships. The "directed" nature of the LSG references implies that there is by default "state/attribute" inheritance from ancestor to descendant (i.e. predecessor to successor).

The first Graph Element in a LSG Segment should always be a Partition Node. The LSG Segment type supports compression on all element data, so all elements in LSG Segment use the Logical Element Header Compressed form of element header data. i.e this means the 3 compression related fields in the Logical Element Header Compressed appear in the first graph element only.

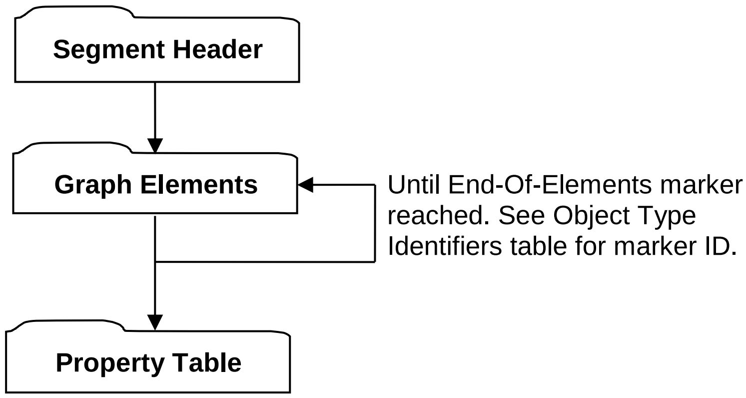


Figure 20 — LSG Segment data collection

Complete description for Segment Header can be found in the Segment Header description.

Graph Elements

Graph Elements form the backbone of the LSG directed acyclic graph structure and in doing so serve as the JT model's fundamental description. There are two general classifications of Graph elements, Node Elements and Attribute Elements.

Node Elements are nodes in the LSG and in general can be categorized as either an internal or leaf node. The leaf nodes are typically shape nodes used to represent a model's physical components and as such either contain or reference some graphical representation or geometry. The internal nodes define the hierarchical organization of the leaf nodes, forming both spatial and logical model relationships, and often contain or reference information (e.g. Attribute Elements) that is inherited down the LSG to all descendant nodes.

Attribute Elements represent graphical data (like appearance characteristics (e.g. colour), or positional transformations) that can be attached to a node, and inherit down the LSG.

Each of these general Graph Element classifications (i.e. Node/Attribute Elements) is sub-typed into specific/concrete types based on data content and implied specialized behaviour. The following sub-sections describe each of the Node and Attribute Element types.

7.1 Node Elements

Node Elements represent the relationships of a model's components. The model's component hierarchy is formed via certain types of Node Elements containing collections of references to other Node Elements who in turn may reference other collections of Node Elements. Node Elements are also the holders (either directly or indirectly) of geometric shape, properties, and other information defining a model's components and representations.

7.1.1 Base Node Element

Object Type ID: 0x10dd1035, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Base Node Element represents the simplest form of a node that can exist within the LSG. The Base Node Element has no implied LSG semantic behaviour nor can it contain any children nodes.

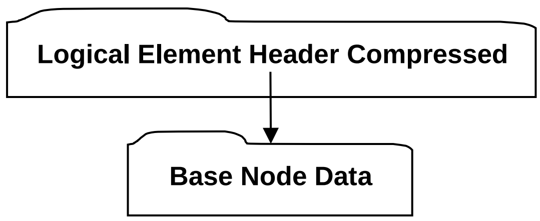


Figure 21 — Base Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

Base Node Data

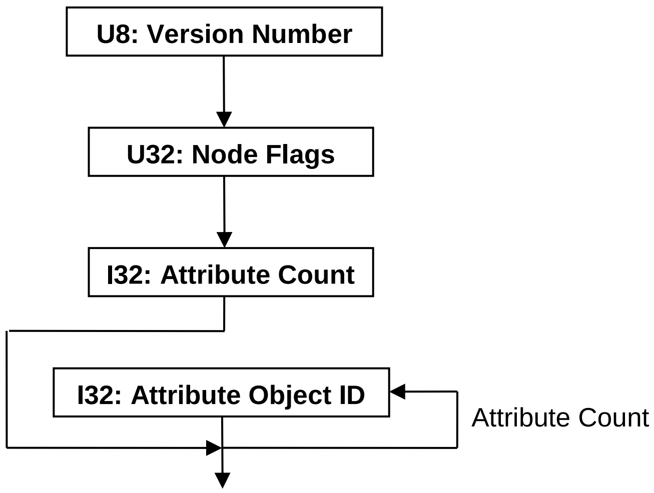


Figure 22 — Base Node Data collection

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U32: Node Flags

Node Flags is a collection of flags. The flags are combined using the binary OR operator. These flags store various state information of the node object. All bits fields that are not defined as in use should be set to "0".

Table 10 — Node Flag values

0x00000001	Ignore Flag = 0 – Algorithms traversing the LSG structure should include/process this node. = 1 – Algorithms traversing the LSG structure should skip the whole subgraph rooted at this node. Essentially the traversal should be pruned.

I32: Attribute Count

Attribute Count indicates the number of Attribute Objects referenced by this Node Object. A node may have zero Attribute Object references.

I32: Attribute Object ID

Attribute Object ID is the identifier for a referenced Attribute Object.

7.1.2 Partition Node Element

Object Type ID: 0x10dd103e, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A partition in a JT file must always be either the root or leaf node. A leaf partition node represents an external JT file reference and provides a means to partition a model into multiple physical JT files (e.g. separate JT file per part in an assembly).

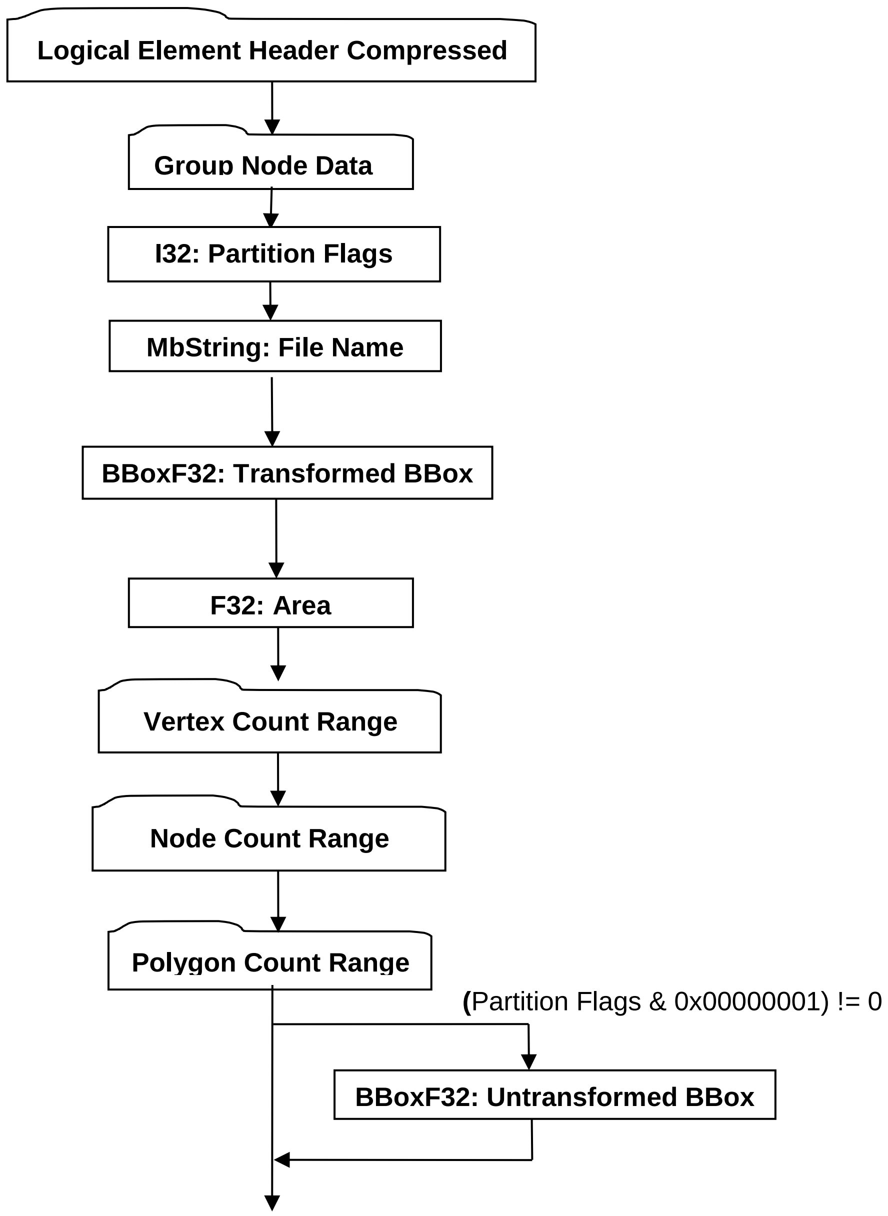


Figure 23 — Partition Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

Complete description for Group Node Data can be found in Group Node Data.

I32: Partition Flags

Partition Flags is a collection of flags. The flags are combined using the binary OR operator. These flags store various state information of the Partition Node Object such as indicating the presence of optional data. All bits fields that are not defined as in use should be set to “0”.

Table 11 — Partition flag bits

0x00000001	Untransformed bounding box is written.

MbString: File Name

File Name is the relative path portion of the Partition’s file location. Where “relative path” should be interpreted to mean the string contains the file name along with any additional path information that locates the partition JT file relative to the location of the referencing JT file.

BBoxF32: Transformed BBox

The Transformed BBox is an MCS axis aligned bounding box and represents the transformed geometry extents for all geometry contained in the Partition Node. This bounding box information may be used by a renderer of JT data to determine whether to load the data contained within the Partition node (i.e. is any part of the bounding box within the view frustum).

F32: Area

Area is the total surface area for this node and all of its descendants. This value is stored in MCS coordinate space (i.e. values scaled by MCS scaling).

Vertex Count Range

Vertex Count Range is the aggregate minimum and maximum vertex count for all descendants of the Partition Node. There is a minimum and maximum value to accommodate descendant branches having LOD nodes, which encompass a range of count values within the branch, and to accommodate nodes that can themselves generate varying representations. The minimum value represents the least vertex count that can be achieved by the Partition Node’s descendants. The maximum value represents the greatest vertex count that can be achieved by the Partition Node’s descendants.

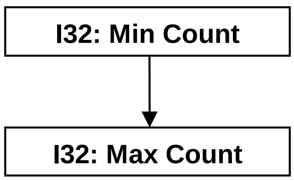


Figure 24 — Vertex Count Range data collection

I32: Min Count

Min Count is the least vertex count that can be achieved by the Partition Node’s descendants.

I32: Max Count

Max Count is the maximum vertex count that can be achieved by the Partition Node’s descendants.

Node Count Range

Node Count Range is the aggregate minimum and maximum count of all node descendants of the Partition Node. There is a minimum and maximum value to accommodate descendant branches having LOD nodes, which encompass a range of descendant node count values within the branch. The minimum value represents the least node count that can be achieved by the Partition Node’s descendants. The maximum value represents the greatest node count that can be achieved by the Partition Node’s descendants.

The data format for Node Count Range is the same as that described in Vertex Count Range.

Polygon Count Range

Polygon Count Range is the aggregate minimum and maximum polygon count for all descendants of the Partition Node. There is a minimum and maximum value to accommodate descendant branches having LOD nodes, which encompass a range of count values within the branch, and to accommodate nodes that can themselves generate varying representations. The minimum value represents the least polygon count that can be achieved by the Partition Node's descendants. The maximum value represents the greatest polygon count that can be achieved by the Partition Node's descendants.

The data format for Polygon Count Range is the same as that described in Vertex Count Range.

BBoxF32: Untransformed BBox

The Untransformed BBox is only present if Bit 0x00000001 of Partition Flags data field is ON. The Untransformed BBox is an LCS axis-aligned bounding box and represents the untransformed geometry extents for all geometry contained in the Partition Node. This bounding box information may be used by a renderer of JT data to determine whether to load the data contained within the Partition node (i.e. is any part of the bounding box within the view frustum).

7.1.3 Group Node Element

Object Type ID: 0x10dd101b, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Group Nodes contain an ordered list of references to other nodes, called the group's *children*. Group nodes may contain zero or more children; the children may be of any node type. Group nodes may not contain references to themselves or their ancestors.

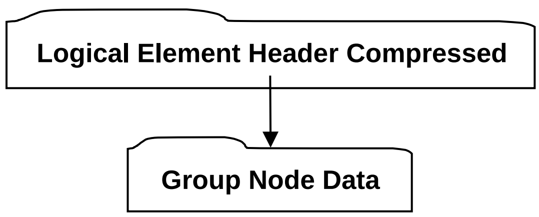


Figure 25 — Group Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

Group Node Data

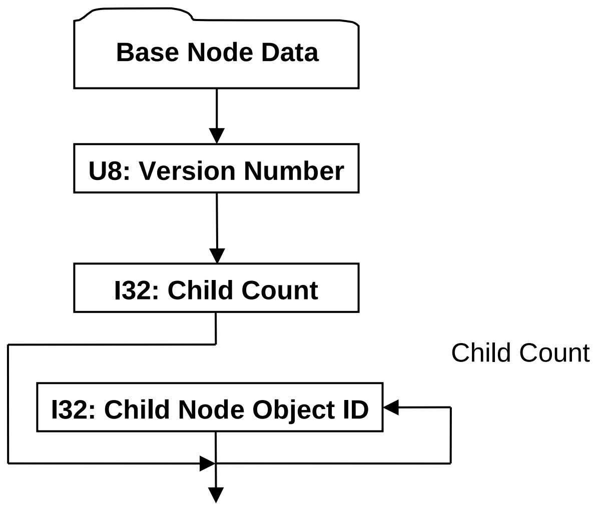


Figure 26 — Group Node Data collection

A complete description of Base Node Data can be found in the LSG Segment section of this document under Base Node Elements in the logical collection describing Base Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: Child Count

Child Count indicates the number of child nodes for this Group Node Object. A node may have zero children.

I32: Child Node Object ID

Child Node Object ID is the identifier for the referenced Node Object.

7.1.4 Instance Node Element

Object Type ID: 0x10dd102a, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

An Instance Node contains a single reference to another node. Their purpose is to allow sharing of nodes and assignment of instance-specific attributes for the instanced node. Instance Nodes may not contain references to themselves or their ancestors.

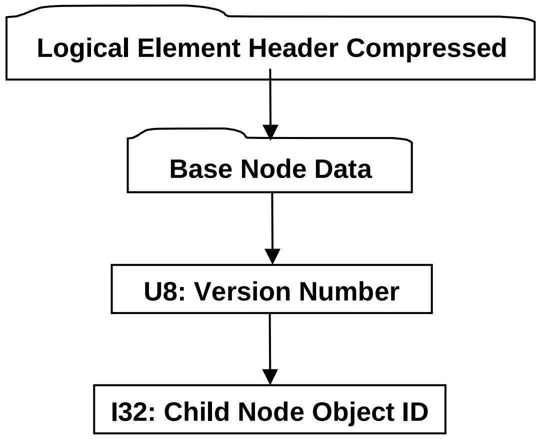


Figure 27 — Instance Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

A complete description of Base Node Data can be found in the LSG Segment section of this document under Base Node Elements in the logical collection describing Base Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: Child Node Object ID

Child Node Object ID is the identifier for the instanced Node Object.

7.1.5 Part Node Element

Object Type ID: 0xce357244, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1

A Part Node Element represents the root node for a particular Part within a LSG structure. Every unique Part represented within a LSG structure should have a corresponding Part Node Element. A Part Node Element typically references (using Late Loaded Property Atoms) additional Part specific geometric data and/or properties (e.g. B-Rep data, PMI data).

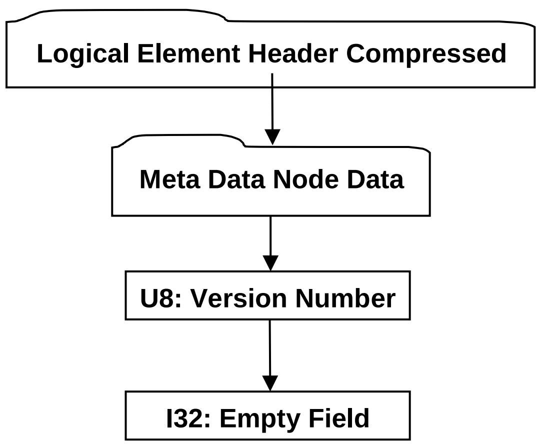


Figure 28 — Part Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

Complete description for Meta Data Node Data can be found in Meta Data Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field description.

7.1.6 Meta Data Node Element

Object Type ID: 0xce357245, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1

The Meta Data Node Element is a node type used for storing references to specific “late loaded” meta-data (e.g. properties, PMI). The referenced meta-data is stored in a separate addressable segment of the JT File (see Meta Data Segment) and thus the use of this Meta Data Node Element is in support of the JT file loader/reader “best practice” of late loading data (i.e. storing the referenced meta-data in separate addressable segment of the JT file allows a JT file loader/reader to ignore this node’s meta-data on initial load and instead late-load the node’s meta-data upon demand so that the associated meta-data does not consume memory until needed).

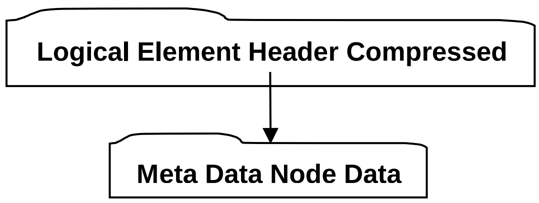


Figure 29 — Meta Data Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

Meta Data Node Data

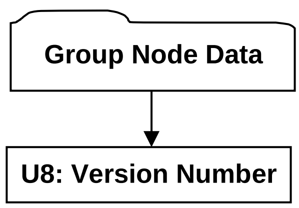


Figure 30 — Meta Data Node Data collection

Complete description for Group Node Data can be found in Group Node Data.

U8: Version Number

Version Number is the version identifier for this data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

7.1.7 LOD Node Element

Object Type ID: 0x10dd102c, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

An LOD Node holds a list of alternate representations. The list is represented as the children of a base group node, however, there are no implicit semantics associated with the ordering. Traversers of LSG may apply semantics to the ordering as part of alternative representation selection.

Each alternative representation could be a sub-assembly where the alternative representation is a group node with an assembly of children.

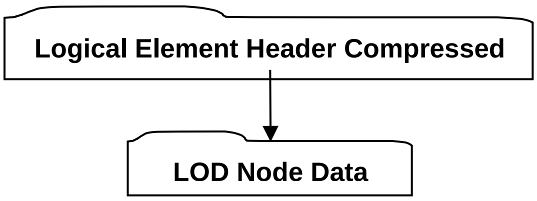


Figure 31 — LOD Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

LOD Node Data

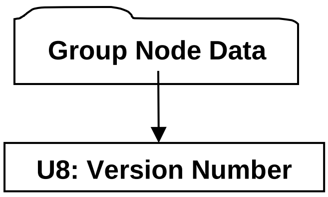


Figure 32 — LOD Node Data collection

Complete description for Group Node Data can be found in Group Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

7.1.8 Range LOD Node Element

Object Type ID: 0x10dd104c, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Range LOD Nodes hold a list of alternate representations and the ranges over which those representations are appropriate. Range Limits indicate the distance between a specified centre point and the eye point, within which the corresponding alternate representation is appropriate. Traversers of LSG consult these range limit values when making an alternative representation selection.

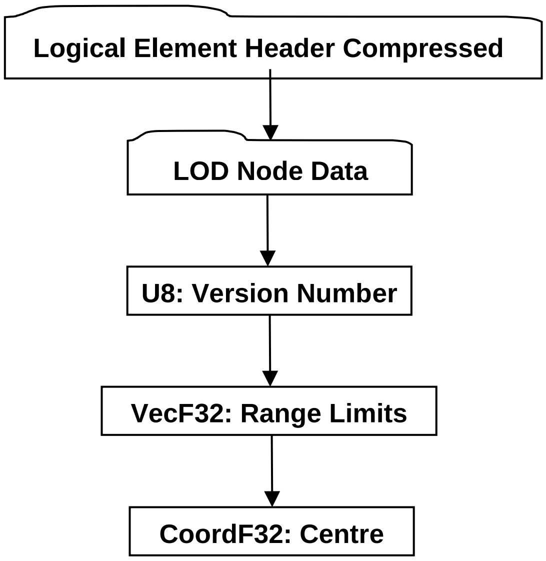


Figure 33 — Range LOD Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for LOD Node Data can be found in LOD Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

VecF32: Range Limits

Range Limits indicate the WCS distance between a specified centre point and the eye point, within which the corresponding alternate representation is appropriate. It is not required that the count of range limits is equivalent to the number of alternative representations. These values are considered “soft values” in that loaders/viewers of JT data are free to throw these values away and compute new values based on their desired LOD selection semantics.

Best practices suggest that LSG traversers apply the following strategy, at Range LOD Nodes, when making alternative representation selection decisions based on Range Limits: The first alternate representation is valid when the distance between the centre and the eye point is less than or equal to the first range limit (and when no range limits are specified). The second alternate representation is valid when the distance is greater than the first limit and less than or equal to the second limit, and so on. The last alternate representation is valid for all distances greater than the last specified limit.

CoordF32: Centre

Centre specifies the X,Y,Z coordinates for the MCS centre point upon which alternative representation selection eye distance computations are based. Typically this location is the centre of the highest-detail alternative representation. These values are considered “soft values” in that loaders/viewers of JT data are free to throw these values away and compute new values based on their desired LOD selection semantics.

7.1.9 Switch Node Element

Object Type ID: 0x10dd10f3, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

The Switch Node is very much like a Group Node in that it contains an ordered list of references to other nodes, called the *children* nodes. The difference is that a Switch Node also contains additional data indicating which child (one or none) a LSG traverser should process/traverse.

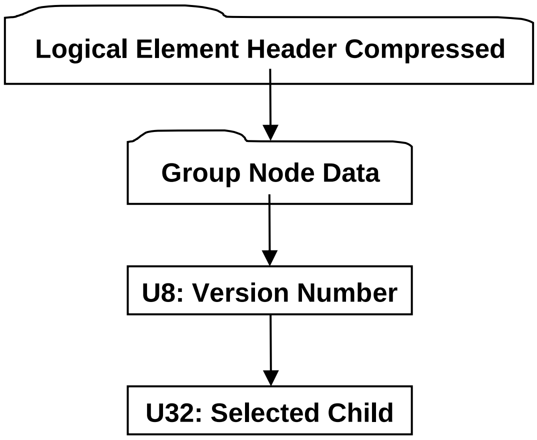


Figure 34 — Switch Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Group Node Data can be found in Group Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U32: Selected Child

Selected Child is the index for the selected child node. Valid Selected Child values reside within the following range: “-1 < Selected Child < Child Count”. Where “-1” indicates that no child is to be selected and “Child Count” is the data field value Group Node Data.

Shape Node Elements

Shape Node Elements are “leaf” nodes within the LSG structure and contain or reference the geometric shape definition data (e.g. vertices, polygons, normals, etc.).

Typically Shape Node Elements do not directly contain the actual geometric shape definition data, but instead reference (using Late Loaded Property Atoms) Shape LOD Segments within the file for the actual geometric shape definition data. Storing the geometric shape definition data within separate independently addressable data segments in the JT file, allows a JT file reader to be structured to support the “best practice” of delaying the loading/reading of associated data until it is actually needed. Complete descriptions for Late Loaded Property Atom Elements and Shape LOD Segments can be found in Late Loaded Property Atom Element and Property Atom Elements respectively.

There are several types of Shape Node Elements which the JT format supports. The following sub-sections document the various Shape Node Element types.

7.1.10 Base Shape Node Element

Object Type ID: 0x10dd1059, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Base Shape Node Element represents the simplest form of a shape node that can exist within the LSG.

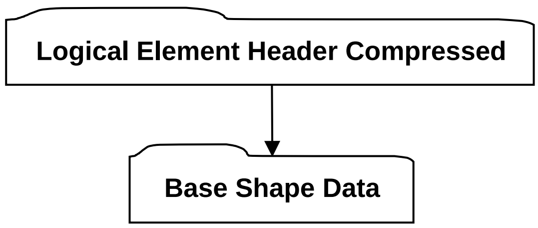


Figure 35 — Base Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data .

Base Shape Data

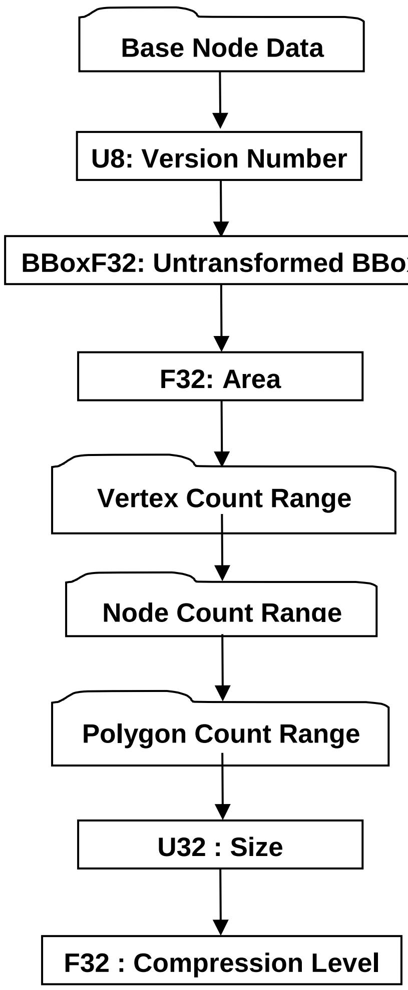


Figure 36 — Base Shape Data collection

A complete description of Base Node Data can be found in the LSG Segment section of this document under Base Node Elements in the logical collection describing Base Node Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

BBoxF32: Untransformed BBox

The Untransformed BBox is an axis-aligned LCS bounding box and represents the untransformed geometry extents for all geometry contained in the Shape Node.

F32: Area

Area is the total surface area for this node and all of its descendants. This value is stored in MCS coordinate space (i.e. values scaled by MCS scaling).

Vertex Count Range

Vertex Count Range is the aggregate minimum and maximum vertex count for this Shape Node. There is a minimum and maximum value to accommodate shape types that can themselves generate varying representations. The minimum value represents the least vertex count that can be achieved by the Shape Node. The maximum value represents the greatest vertex count that can be achieved by the Shape Node.

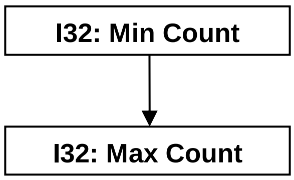


Figure 37 — Vertex Count Range data collection

I32: Min Count

Min Count is the least vertex count that can be achieved by this Shape Node.

I32: Max Count

Max Count is the maximum vertex count that can be achieved by this Shape Node. A value of “-1” indicates maximum vertex count is unknown.

Node Count Range

Node Count Range is the aggregate minimum and maximum count of all node descendants of the Shape Node. The minimum value represents the least node count that can be achieved by the Shape Node’s descendants. The maximum value represents the greatest node count that can be achieved by Shape Node’s descendants. For Shape Nodes the minimum and maximum count values should always be equal to “1”.

Polygon Count Range

Polygon Count Range is the aggregate minimum and maximum polygon count for this Shape Node. There is a minimum and maximum value to accommodate shape types that can themselves generate varying representations. The minimum value represents the least polygon count that can be achieved by the Shape Node. The maximum value represents the greatest polygon count that can be achieved by the Shape Node.

The data format for Polygon Count Range is the same as that described in Vertex Count Range.

U32 : Size

Size specifies the in memory length in bytes of the associated/referenced Shape LOD Element. This Size value has no relevancy to the on-disk (JT File) size of the associated/referenced Shape LOD Element. A value of zero indicates that the in memory size is unknown. See Shape LOD Element for complete description of Shape LOD Elements. JT file loaders/readers can leverage this Size value during late load processing to help pre-determine if there is sufficient memory to load the Shape LOD Element.

F32: Compression Level

Compression Level specifies the qualitative compression level applied to the associated/referenced Shape LOD Element. See the chapter on Shape LOD Segment for complete description of Shape LOD Elements. This compression level value is a qualitative representation of the compression applied to the Shape LOD Element. The absolute compression (derived from this qualitative level) applied to the Shape LOD Element is physically represented in the JT format by other data stored with both the Shape Node and the Shape LOD Element (e.g. Quantization Parameters), and thus it's not necessary to understand how to map this qualitative value to absolute compression values in order to uncompress/decode the data.

Table 12 — Compression level values

= 0.0	"Lossless" compression used.
= 0.1	"Minimally Lossy" compression used. This setting generally results in modest compression ratios with little if any visual difference when compared to the same images rendered from "Lossless" compressed Shape LOD Element.
= 0.5	"Moderate Lossy" compression used. The setting results in more data loss than "Minimally Lossy" and thus higher compression ratio is obtained. Some visual difference will likely be noticeable when compared to the same images rendered from "Lossless" compressed Shape LOD Element.
= 1.0	"Aggressive Lossy" compression used. With this setting as much data as possible will be thrown away, resulting in highest compression ratio, while still maintaining a modestly useable representation of the underlying data. Visual differences may be evident when compared to the same images rendered from "Lossless" compressed Shape LOD Element.

7.1.11 Vertex Shape Node Element

Object Type ID: 0x10dd107f, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Vertex Shape Node Element represents shapes defined by collections of vertices.

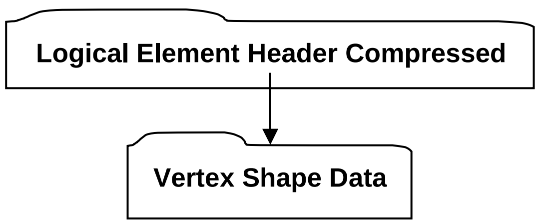


Figure 38 — Vertex Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Vertex Shape Data

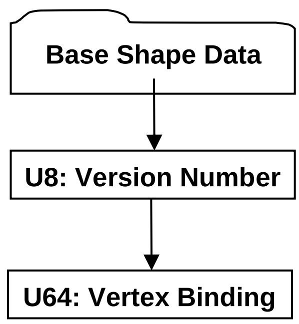


Figure 39 — Vertex Shape Data collection

A complete description of Base Shape Data can be found in the LSG Segment section of this document under Base Shape Node Element in the logical collection describing Base Shape Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U64: Vertex Binding

Vertex Bindings is a collection of normal, texture coordinate, and colour binding information encoded within a single U64. All bits fields that are not defined as in use should be set to “0”. For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

7.1.12 Tri-Strip Set Shape Node Element

Object Type ID: 0x10dd1077, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Tri-Strip Set Shape Node Element defines a collection of independent and unconnected triangle strips. Each strip constitutes one primitive of the set and is defined by one list of vertex coordinates.

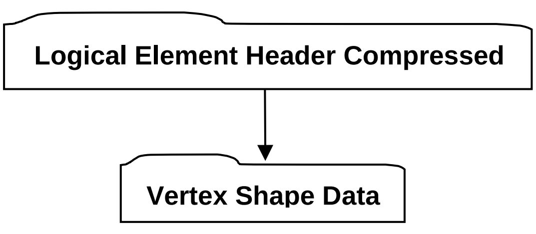


Figure 22 — Tri-Strip Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Vertex Shape Data can be found in the LSG Segment section of this document under Vertex Shape Node Element in the logical collection describing Vertex Shape Data.

7.1.13 Polyline Set Shape Node Element

Object Type ID: 0x10dd1046, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Polyline Set Shape Node Element defines a collection of independent and unconnected polylines. Each polyline constitutes one primitive of the set and is defined by one list of vertex coordinates.

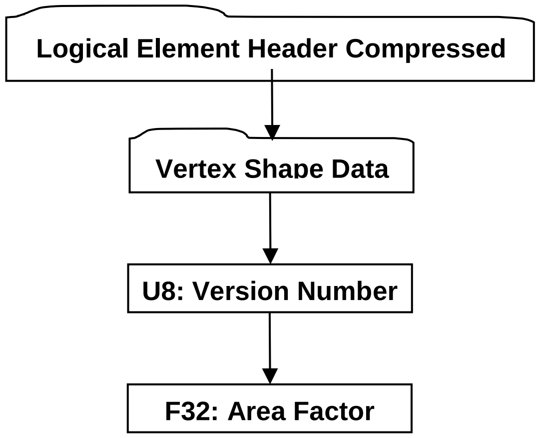


Figure 40 — Polyline Set Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Vertex Shape Data can be found in the LSG Segment section of this document under Vertex Shape Node Element in the logical collection describing Vertex Shape Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

F32: Area Factor

Area Factor specifies a multiplier factor applied to a Polyline Set computed surface area. In JT data viewer applications there may be LOD selection semantics that are based on screen coverage calculations. The so-called ”surface area” of a polyline is computed as if each line segment were a square. This Area Factor turns each edge into a narrow rectangle. Valid Area Factor values lie in the range (0,1].

7.1.14 Point Set Shape Node Element

Object Type ID: 0x98134716, 0x0010, 0x0818, 0x19, 0x98, 0x08, 0x00, 0x09, 0x83, 0x5d, 0x5a

A Point Set Shape Node Element defines a collection of independent and unconnected points. Each point constitutes one primitive of the set and is defined by one vertex coordinate.

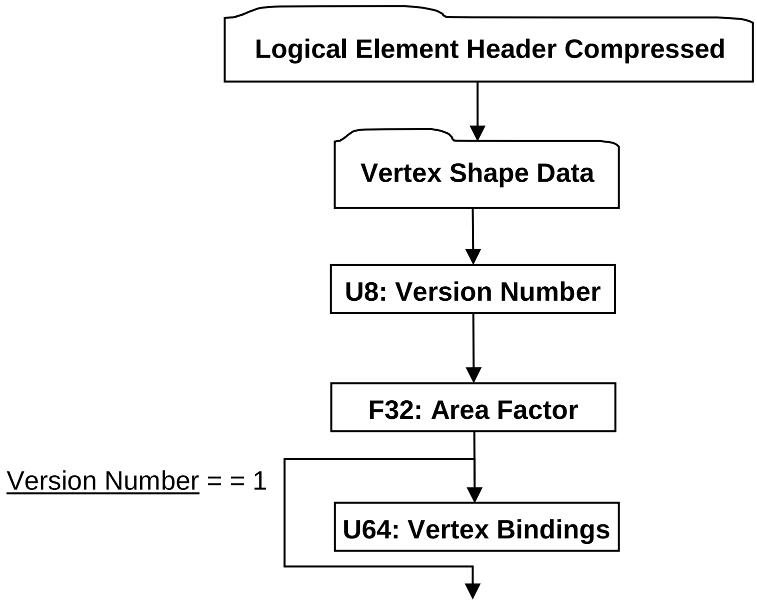


Figure 41 — Point Set Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Vertex Shape Data can be found in the LSG Segment section of this document under Vertex Shape Node Element in the logical collection describing Vertex Shape Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

F32: Area Factor

Area Factor specifies a multiplier factor applied to the Point Set computed surface area. In JT data viewer applications there may be LOD selection semantics that are based on screen coverage calculations. The computed “surface area” of a Point Set is equal to the larger (i.e. whichever is greater) of either the area of the Point Set’s bounding box, or “1.0”. Area Factor scales the result of this “surface area” computation.

U64: Vertex Bindings

Vertex Bindings is a collection of normal, texture coordinate, and colour binding information encoded within a single U64. All bits fields that are not defined as in use should be set to “0”. For more information see Vertex Shape LOD Data U64: Vertex Bindings.

7.1.15 Polygon Set Shape Node Element

Object Type ID: 0x10dd1048, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Polygon Set Shape Node Element defines a collection of independent and unconnected polygons. Each polygon constitutes one primitive of the set and is defined by one list of vertex coordinates.

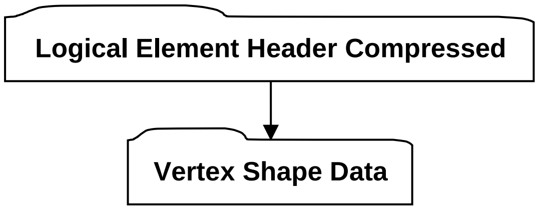


Figure 42 — Polygon Set Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Vertex Shape Data can be found in the LSG Segment section of this document under Vertex Shape Node Element in the logical collection describing Vertex Shape Data.

7.1.16 NULL Shape Node Element

Object Type ID: 0xd239e7b6, 0xdd77, 0x4289, 0xa0, 0x7d, 0xb0, 0xee, 0x79, 0xf7, 0x94, 0x94

A NULL Shape Node Element defines a shape which has no direct geometric primitive representation (i.e. it is empty/NULL). NULL Shape Node Elements are often used as “proxy/placeholder” nodes within the serialized LSG when the actual Shape LOD data is run time generated (i.e. not persisted).

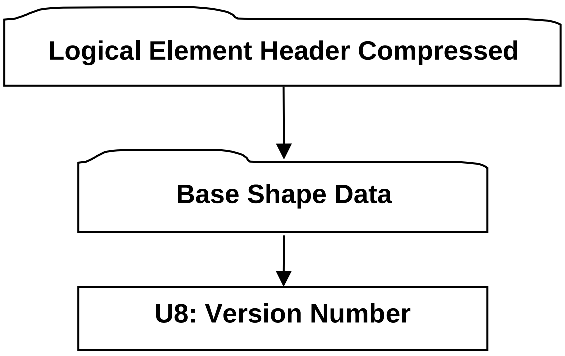


Figure 43 — NULL Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Shape Data can be found in the LSG Segment section of this document under Base Shape Node Element in the logical collection describing Base Shape Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

7.1.17 Primitive Set Shape Node Element

Object Type ID: 0xe40373c1, 0x1ad9, 0x11d3, 0x9d, 0xaf, 0x0, 0xa0, 0xc9, 0xc7, 0xdd, 0xc2

A Primitive Set Shape Node Element represents a list/set of primitive shapes (e.g. box, cylinder, sphere, etc.) whose LODs can be procedurally generated. “Procedurally generate” means that the raw geometric shape definition data (e.g. vertices, polygons, normals, etc) for LODs is not directly stored;

instead some basic shape information is stored (e.g. sphere centre and radius) from which LODs can be generated.

Primitive Set Shape Node Elements actually do not even directly contain this basic shape definition data, but instead reference (using Late Loaded Property Atoms) Primitive Set Shape Node Element within the file for the actual basic shape definition data. Storing the basic shape definition data within separate independently addressable data segments in the JT file, allows a JT file reader to be structured to support the “best practice” of delaying the loading/reading of associated data until it is actually needed. Complete descriptions for Late Loaded Property Atom Elements and Primitive Set Shape Element can be found in Late Loaded Property Atom Element and Primitive Set Shape Element respectively.

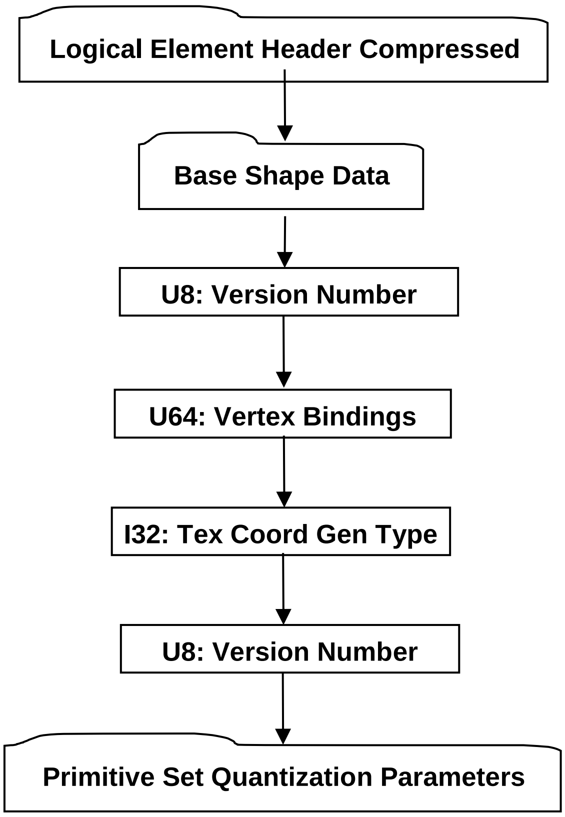


Figure 44 — Primitive Set Shape Node Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Shape Data can be found in the LSG Segment section of this document under Base Shape Node Element in the logical collection describing Base Shape Data.

U8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U64: Vertex Bindings

Vertex Bindings is a collection of normal, texture coordinate, and colour binding information encoded within a single U64. All bits fields that are not defined as in use should be set to “0”. For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

I32: Tex Coord Gen Type

Texture Coord Gen Type specifies how a texture is applied to each face of the primitive. Single tile means one copy of the texture will be stretched to fit the face, isotropic means that the texture will be duplicated on the longer dimension of the face in order to maintain the texture's aspect ratio.

Table 13 — Texture Coord Gen Type values

= 0	Single Tile…Indicates that a single copy of a texture image will be applied to significant primitive features (i.e. cube face, cylinder wall, end cap) no matter how eccentrically shaped.
= 1	Isotropic…Implies that multiple copies of a texture image may be mapped onto eccentric surfaces such that a mapped texel stays approximately square.

U8: Version Number

Version Number is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow.

Table 14 — Version Number values

= 0	Version 0 Format
= 1	Version 1 Format

Primitive Set Quantization Parameters

Primitive Set Quantization Parameters specifies for the two shape data type grouping (i.e. Vertex, Colour) the number of quantization bits used for given qualitative compression level. Although these values are saved in the associated/referenced Shape LOD Element, they are also saved here so that a JT File loader/reader does not have to load the Shape LOD Element in order to determine the Shape quantization level. See Shape LOD Element for complete description of Shape LOD Elements.

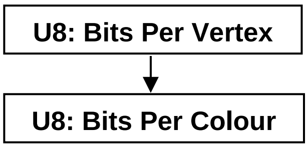


Figure 45 — Primitive Set Quantization Parameters data collection

U8: Bits Per Vertex

Bits Per Vertex specifies the number of quantization bits per vertex coordinate component. Value shall be within range [0:24] inclusive.

U8: Bits Per Colour

Bits Per Colour specifies the number of quantization bits per colour component. Value shall be within range [0:24] inclusive.

7.2 Attribute Elements

Attribute Elements (e.g. colour, texture, material, lights, etc.) are placed in LSG as objects associated with nodes. Attribute Elements are not nodes themselves, but can be associated with any node.

For applications producing or consuming JT format data, it is important that the JT format semantics of how attributes are meant to be applied and accumulated down the LSG are followed. If not followed, then consistency between the applications in terms of 3D positioning and rendering of LSG model data will not be achieved.

To that end each attribute type defines its own application and accumulation semantics, but in general attributes at lower levels in the LSG take precedence and replace or accumulate with attributes set at higher levels. Nodes without associated attributes inherit those of their parents. Attributes inherit only from their parents, thus a node’s attributes do not affect that node’s siblings. The root of a partition inherits the attributes in effect at the referring partition node.

In previous version of the JT file format, Attributes held a single “final” bit denoting that no further accumulations were to take place into that attribute type by Attributes of the same type lying below it in the scene graph. JT V10.5 replaces this single bit with separate “field final” bits for each field within the Attribute. Different Attributes have different fields, and are documented accordingly in the following sections. Only three Attributes define more than one internal field (i.e. Material Attribute Element, Texture Image Attribute Element, and Draw Style Attribute Element). All other Attributes merely define a single default field that encompasses their entire state.

In addition to “field final” bits, each Attribute also defines a parallel set of “field inhibit” bits. These bits denote, on a field-by-field basis, whether a field is allowed to accumulate. Said differently, if a field inhibit bit is set to 0, the field accumulates normally; if the bit is set to 1, then the field will not accumulate, and is ignored.

Descendants can explicitly do a one-shot override of “final” using the attribute “force” flag (see Base Attribute Data), but do not by default. Note that “force” does not turn OFF “final” – it is simply a one-shot override of “final” for the specific attribute marked as “forcing.” Note that the “force” flag is attribute-wide – not on a field-by-field basis like field-finals and field-inhibits. An analogy for this “force” and “final” interaction is that “final” is a back-door in the attribute accumulation semantics, and that “force” is a doggy-door in the back-door!

Base Attribute Data

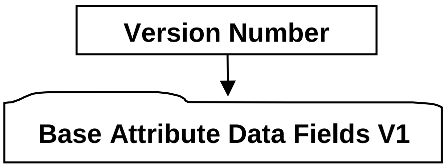


Figure 46 — Base Attribute Data collection

I8: Version Number

Version Number can have the value one or two. If the value is two then the Logical Collection Base Attribute Data Fields V2 will be read after each derived attribute type (i.e.. Material, Texture etc.) In all cases Base Attribute Data Fields V1 must be read first.

Base Attribute Data Fields V1

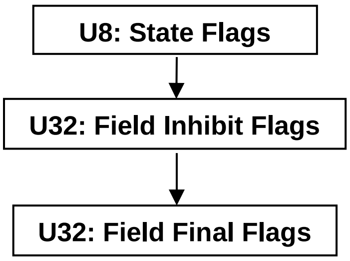


Figure 47 — Base Attribute Data Fields V1

U8: State Flags

State Flags is a collection of flags. The flags are combined using the binary OR operator and store various state information for Attribute Elements; such as indicating that the attributes accumulation is final. All bits fields that are not defined as in use should be set to “0”.

Table 15 — State Flag values

0x01	Unused
0x02	Accumulation Force flag. Provides a way to assign nodes in LSG, attributes that shall not be overridden by ancestors. = 0 – Accumulation of this attribute obeys ancestor’s Final flag setting. = 1 – Accumulation of this attribute is forced (overrides ancestor’s Final flag setting)
0x04	Accumulation Ignore Flag. Provides a way to indicate that the attribute is to be ignored (not accumulated). = 0 – Attribute is to be accumulated normally (subject to values of Force/Final flags) = 1 – Attribute is to be ignored.
0x08	Attribute Persistable Flag. Provides a way to indicate that the attribute is to be persistable to a JT file. = 0 – Attribute is to be non-persistable. = 1 – Attribute is to be persistable.

U32: Field Inhibit Flags

Field Inhibit Flags is a collection of flags, each flag corresponding to a collection of state data within a particular Attribute type. Each value (or semantically related set of values) present in an Attribute Element is given a field number ranging from 0 to 31. If the field’s corresponding bit in Inhibit Flags is set, then the field should not participate in attribute accumulation. All bits are reserved.

See each particular Attribute Element (e.g. Material Attribute Element) for a description of bit field assignments for each attribute value.

U32: Field Final Flags

Field Final Flags is a collection of flags, each flag being parallel to the corresponding flag in the Field Inhibit Flags. If the field’s bit in Field Final Flags is set, then that field within the Attribute will become “final” and will not allow any subsequent accumulation into the specified field. All bits are reserved.

See each particular Attribute Element for a description of bit field assignments for each Attribute value.

Base Attribute Data Fields V2

This logical collection is found in the Attribute Element data descriptions for; Material, Texture Image, Draw Style, Light Set Linestyle, Pointstyle, Geometric Transform, and Palette Map.

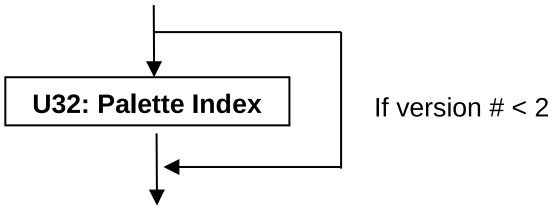


Figure 48 — Base Attribute Data Fields V2

U32: Palette Index

The palette index field is used to implicitly create a State Palette that may later be indexed by a PaletteMap attribute. For a full description see the Per Face Group Attribute Annex in this document.

Palette Index default value is -1

7.2.1 Material Attribute Element

Object Type ID: 0x10dd1030, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Material Attribute Element defines the material properties of a object. JT format LSG traversal semantics state that material attributes accumulate down the LSG by replacement.

The Field Inhibit flag (see Base Attribute Data) bit assignments for the Material Attribute Element data fields, are as follows:

Table 16 — Material Attribute data field inhibit values

Field Inhibit Flag Bit	Data Field(s) Bit Applies To
0	Ambient Common RGB Value, Ambient Colour
1	Specular Common RGB Value, Specular Colour
2	Emission Common RGB Value, Emission Colour
3	Blending Flag, Source Blending Factor, Destination Blending Factor
4	Override Vertex Colour Flag
5	Material Reflectivity
6	Diffuse Colour
7	Diffuse Alpha
8	

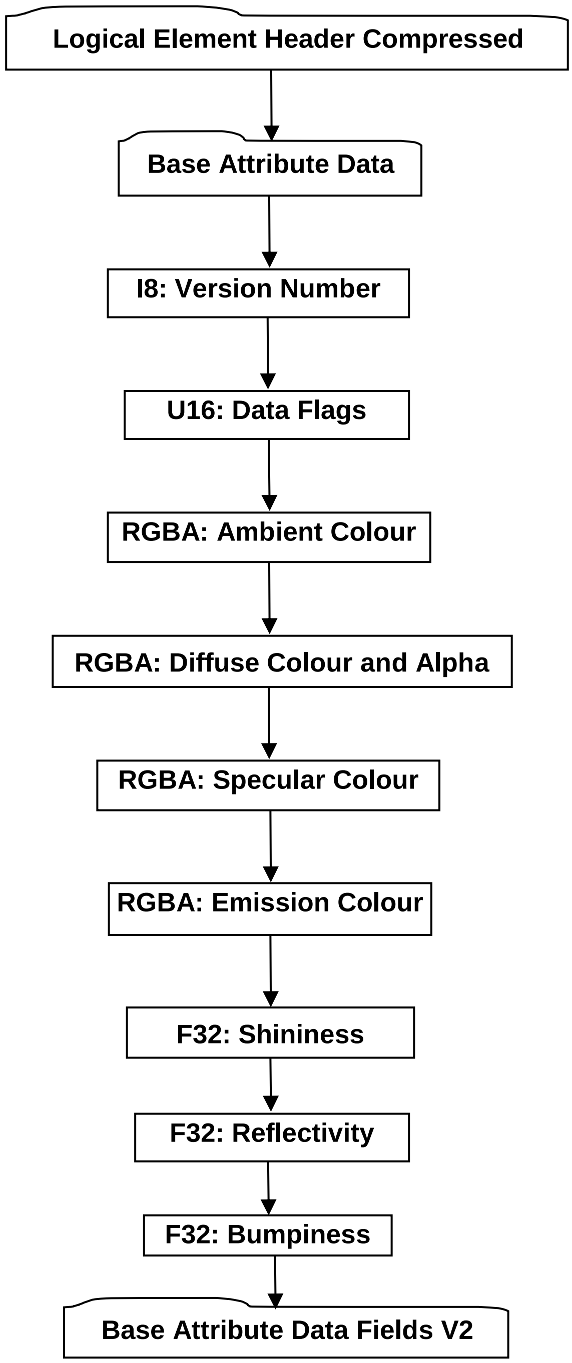


Figure 49 — Material Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow.

Table 17 — Material Attribute Version number value

= 1	Version-1 Format

U16: Data Flags

Data Flags is a collection of flags and factor data. The flags and factor data are combined using the binary OR operator. The flags store information to be used for interpreting how to read subsequent Material data fields. All bits fields that are not defined as in use should be set to "0".

Table 18 — Material Attribute Data Flag values

0x0010	Blending Flag. Blending is a colour combining operation in the graphics pipeline that happens just before writing a colour to the framebuffer. If Blending is ON then incoming fragment RGBA colour values are used (based on Source Blend Factor) and existing framebuffer's RGBA colour values are used (based on Destination Blend Factor) to blend between the incoming fragment RGBA and the current frame buffer RGBA to arrive at a new RGBA colour to write into the framebuffer. If Blending is OFF then incoming fragment RGBA colour is written directly into framebuffer unmodified (i.e. completely overriding existing framebuffer RGBA colour). = 0 – Blending OFF. = 1 – Blending ON
0x0020	Override Vertex Colours Flag. If ON, then a shape's per vertex colours are to be overridden by the accumulated Material colour. = 0 – Override OFF = 1 – Override ON
0x07C0	Source Blend Factor (stored in bits 6 – 10 or in binary notation 0000011111000000). If Blending Flag enabled, this value indicates how the incoming fragment's (i.e. the source) RGBA colour values are to be used to blend with the current framebuffer's (i.e. the destination) RGBA colour values. Additional information on the interpretation of the Blending Factor values and how one might leverage them to render an image can be found in reference [1] listed in the bibliography section. = 0 – Interpret same as OpenGL **GL_ZERO** Blending Factor = 1 – Interpret same as OpenGL **GL_ONE** Blending Factor = 2 – Interpret same as OpenGL **GL_DST_COLOUR** Blending Factor = 3 – Interpret same as OpenGL **GL_SRC_COLOUR** Blending Factor = 4 – Interpret same as OpenGL **GL_ONE_MINUS_DST_COLOUR** Blending Factor = 5 – Interpret same as OpenGL **GL_ONE_MINUS_SRC_COLOUR** Blending Factor = 6 – Interpret same as OpenGL **GL_SRC_ALPHA** Blending Factor = 7 – Interpret same as OpenGL **GL_ONE_MINUS_SRC_ALPHA** Blending Factor = 8 – Interpret same as OpenGL **GL_DST_ALPHA** Blending Factor = 9 – Interpret same as OpenGL **GL_ONE_MINUS_DST_ALPHA** Blending Factor = 10 – Interpret same as OpenGL **GL_SRC_ALPHA_SATURATE** Blending Factor
0xF800	Destination Blend Factor (stored in bits 11 – 15 or in binary notation 1111100000000000).). If Blending Flag enabled, this value indicates how the current framebuffer's (the destination) RGBA colour values are to be used to blend with the incoming fragment's (the source) RGBA colour values. Additional information on the interpretation of the Blending Factor values and how one might leverage them to render an image can be found in reference [1] listed the bibliography section. = 0 – Interpret same as OpenGL **GL_ZERO** Blending Factor = 1 – Interpret same as OpenGL **GL_ONE** Blending Factor = 2 – Interpret same as OpenGL **GL_DST_COLOUR** Blending Factor = 3 – Interpret same as OpenGL **GL_SRC_COLOUR** Blending Factor = 4 – Interpret same as OpenGL **GL_ONE_MINUS_DST_COLOUR** Blending Factor = 5 – Interpret same as OpenGL **GL_ONE_MINUS_SRC_COLOUR** Blending Factor = 6 – Interpret same as OpenGL **GL_SRC_ALPHA** Blending Factor = 7 – Interpret same as OpenGL **GL_ONE_MINUS_SRC_ALPHA** Blending Factor

	= 8 – Interpret same as OpenGL **GL_DST_ALPHA** Blending Factor = 9 – Interpret same as OpenGL **GL_ONE_MINUS_DST_ALPHA** Blending Factor = 10 – Interpret same as OpenGL **GL_SRC_ALPHA_SATURATE** Blending Factor

RGBA: Ambient Colour

Ambient Colour specifies the ambient red, green, blue, alpha colour values of the material.

RGBA: Diffuse Colour and Alpha

Diffuse Colour and Alpha specify the diffuse red, green, blue colour components and alpha value of the material.

RGBA: Specular Colour

Specular Colour specifies the specular red, green, blue, alpha colour values of the material.

RGBA: Emission Colour

Emission Colour specifies the emissive red, green, blue, alpha colour values of the material.

F32: Shininess

Shininess is the exponent associated with specular reflection and highlighting of the Phong specular lighting model. Shininess controls the degree with which the specular highlight decays. Only values in the range [1,128] are valid.

F32: Reflectivity

Reflectivity specifies the material reflectivity of the material. It represents the fraction of light reflected in the mirror direction by the material. Only values in the range [0.0, 1.0] are valid.

F32: Bumpiness

Bumpiness is used to control bump mapping, and specifies the degree to which bump mapping modifies the local normal vector. A value of 1.0 is the default. Values larger than 1.0 are intended to make the shaded object look as if it is more highly embossed; values between 0.0 and 1.0 make it look less so. Negative values are legal and make the object appear to be *engraved* rather than embossed.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

7.2.2 Texture Image Attribute Element

Object Type ID: 0x10dd1073, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Texture Image Attribute Element defines a texture image and its mapping environment. JT format LSG traversal semantics state that texture image attributes accumulate down the LSG by replacement on a *per texture channel* basis. See below for more information on texture image channels.

Note that additional information on the interpretation of the various Texture Image Attribute Element data fields can be found in the OpenGL references listed in the bibliography section [1].

The Field Inhibit and Field Final flag (see Base Attribute Data) bit assignments for the Texture Image Attribute Element data fields, are as follows:

Table 19 — Texture Image Attribute data field inhibit values

Field Inhibit Flag Bit	Data Field(s) Bit Applies To
0	I32 : Texture Type, Mipmap Image Texel Data, MBString:External Storage Name, Shared Image Flag
1	Border Mode, Border Colour
2	Mipmap Minification Filter, Mipmap Magnification Filter
3	S-Dimen Wrap Mode, T-Dimen Wrap Mode, R-Dimen Wrap Mode
4	Blend Type, Blend Colour
5	Texture Transform
6	Tex Coord Gen Mode, Tex Coord Reference Plane
7	Internal Compression Level

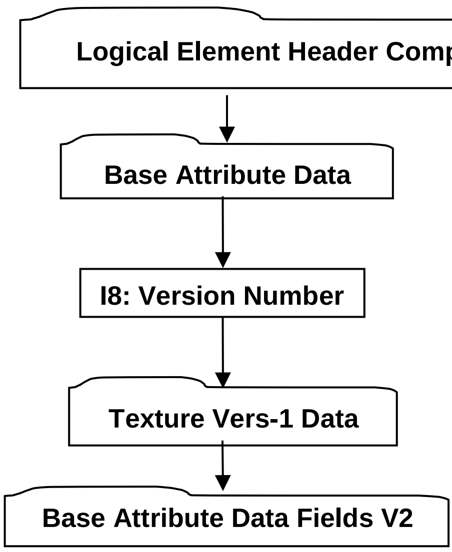


Figure 50 — Texture Image Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow.

Table 20 — Texture Image Version Number values

= 1	Version-1 Format

When a data element in the JT file is versioned, it is for the purpose of adding a few pieces of new data onto the end of the existing data format. In this way, older viewers and readers of the JT file that do not yet know about higher local versions will naturally read the lower-numbered version blocks and ignore the higher-numbered ones they do not know how to read. At present, this mechanism is not being used in JT V10.5, but experience has shown from previous versions of JT that it probably will become useful at some point during the life of this version.

Texture Vers-1 Data

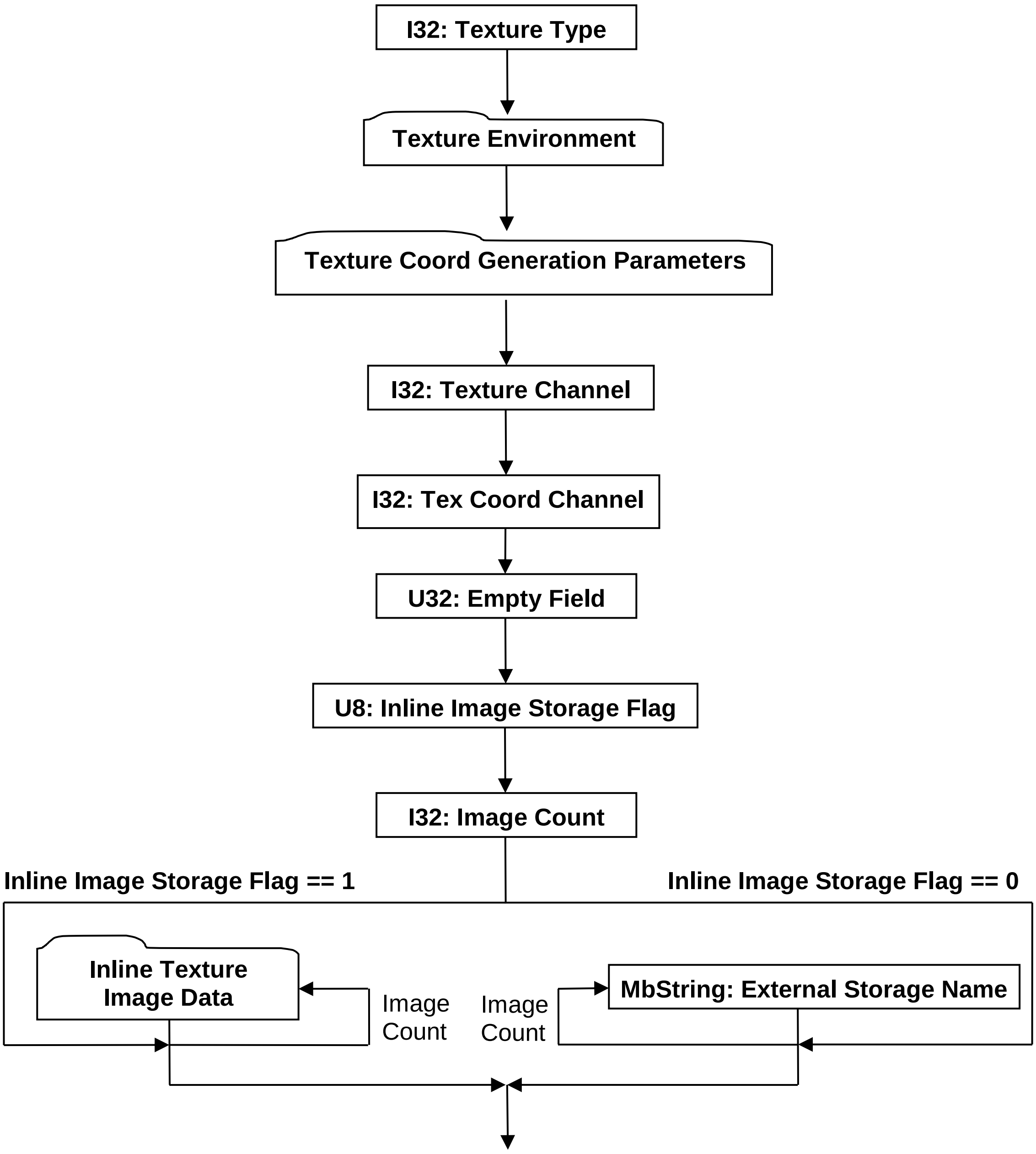


Figure 51 — Texture Vers-1 Data collection

Complete details for Texture Environment can be found in Texture Environment.

Complete details for Texture Coord Generation Parameters can be found in Texture Coord Generation Parameters.

Complete details for Inline Texture Image Data can be found in Inline Texture Image Data.

I32: Texture Type

Texture Type specifies the type of texture. A new texture type, separator texture, is defined in Texture Vers-1 Data to support resetting the texture accumulation state mid-graph. Shadow maps and pre-filtered light maps, however, are a general exception to this rule. In the following list, “image” refers to an image texture, “pre-lit” indicates that the image texture is to be applied before lighting when rendering the object to which it is applied, and “post-lit” indicates that the image texture is to be applied after lighting. A gloss map is a pre-lit texture that applies itself to the specular material component of lighting instead of the diffuse component. A light map is an environment texture (texture at infinity surrounding the whole model) that serves as a source of illumination during shading calculations.

Table 21 — Texture Vers-1 Texture Type values

Texture Type	Description	Explicit Channel	Auto Channel
= 0	None.	N/A	N/A
= 1	One-Dimensional post-lit image texture.	Yes	No
= 2	Two-Dimensional post-lit image texture.	Yes	No
= 3	Three-Dimensional post-lit image texture.	Yes	No
= 4	Two-Dimensional 3-component tangent-space normal map.	No	Yes
= 5	Cube post-lit image texture.	Yes	No
= 7	Cube pre-lit image texture.	Yes	No
= 8	One-Dimensional pre-lit image texture.	Yes	No
= 9	Two-Dimensional pre-lit image texture.	Yes	No
= 10	Three-Dimensional pre-lit image texture.	Yes	No
= 11	Cube environment map.	No	Yes
= 12	One-Dimensional gloss map (specular) texture.	No	Yes
= 13	Two-Dimensional gloss map (specular) texture.	No	Yes
= 14	Three-Dimensional gloss map (specular) texture.	No	Yes
= 15	Cube gloss map (specular) texture.	No	Yes
= 16	Two-Dimensional 1-component bumpmap.	No	Yes
= 17	Two-Dimensional 3-component world-space normal map.	No	Yes
= 18	Two-Dimensional sphere environment map.	No	Yes
= 19	Two-Dimensional latitude/longitude environment map.	No	Yes
= 20	Two-Dimensional spherical diffuse light map.	No	Yes
= 21	Cube diffuse light map.	No	Yes
= 22	Two-Dimensional latitude/longitude diffuse light map.	No	Yes
= 23	Two-Dimensional spherical specular light map.	No	Yes
= 24	Cube specular light map.	No	Yes
= 25	Two-Dimensional latitude/longitude specular light map.	No	Yes
= 26	Resets texture state except shadow map and light maps.	N/A	N/A

I32: Texture Channel

Texture Channel specifies the texture channel number for the Texture Image Element. For purposes of multi-texturing, the JT concept of a texture channel corresponds to the OpenGL concept of a “texture unit.” The Texture Channel value shall be between -1 and 2,147,483,647 inclusive. The value -1 is accepted to denote a texture whose channel number is to be automatically assigned. This assignment will never displace another texture with an explicit texture channel assignment from its slot. Best practices suggest that a renderer of JT data ignore all but channel-0 if the renderer does not support multi-textured geometry. Also for purposes of blending, any renderer of JT data should ensure that higher numbered texture channels “blend over” lower numbered ones.

Pre- and post-lit image textures shall specify an explicit texture channel. All other texture types shall specify -1 for their texture channel.

U32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field description.

U8: Inline Image Storage Flag

Inline Image Storage Flag is a flag that indicates whether the texture image is stored within the JT File (i.e. inline) or in some other external file.

Table 22 — Texture Vers-1 Inline Image Storage Flag values

= 0	Texture image stored in an external file.
= 1	Texture image stored inline in this JT file.

I32: Image Count

Image Count specifies the number of texture images. A “Cube Map” I32: Texture Type shall have six images while all other Texture Types should only have one image.

MbString: External Storage Name

External Storage Name is a string identifying the name of an external texture image storage. External Storage Name is only present if data field Inline Image Storage Flag equals “0.” If present there will be data field Image Count number of External Storage Name instances. This External Storage Name string is a relative path based name for the texture image file. Where “relative path” should be interpreted to mean the string contains the file name along with any additional path information that locates the texture image file relative to the location of the referencing JT file.

I32: Tex Coord Channel

Tex Coord Channel specifies the channel number for texture coordinate generation. Value shall be within range [-1, 2147483647] inclusive.

Texture Environment

The Texture Environment is a collection of data defining various aspects of how a texture image is to be mapped/applied to a surface.

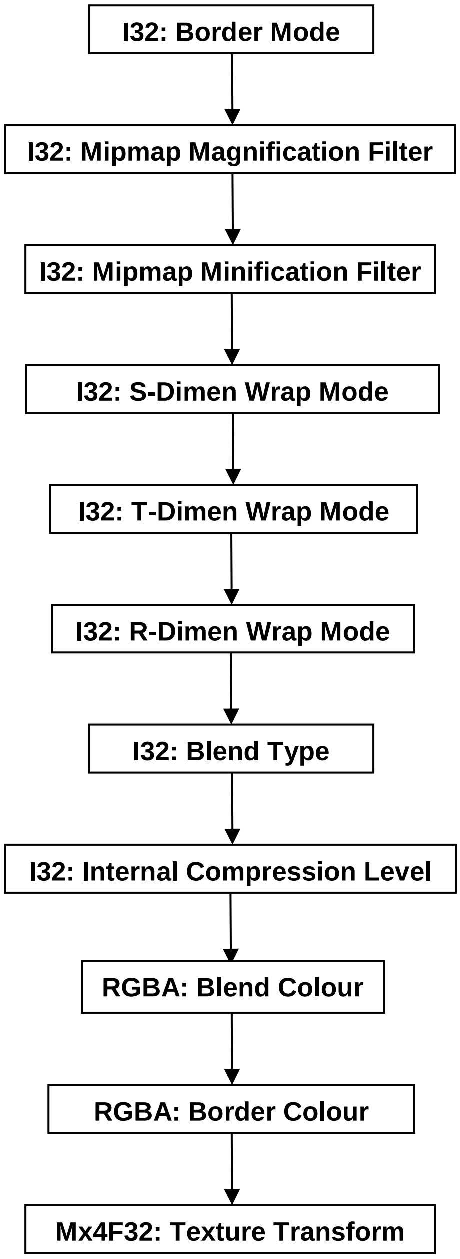


Figure 52 — Texture Environment data collection

I32: Border Mode

Border Mode specifies the texture border mode.

Table 23 — Texture Vers-1 Texture Environment Border Mode values

= 0	No border.
= 1	Constant Border Colour. Indicates that the texture has a constant border colour whose value is defined in data field Border Colour.
= 2	Explicit. Indicates that a border texel ring is present in the texture image definition.

I32: Mipmap Magnification Filter

Mipmap Magnification Filter specifies the texture filtering method to apply when a single pixel on screen maps to a tiny portion of a texel.

Table 24 — Texture Vers-1 Texture Environment Mipmap Magnification Filter values

= 0	None.
= 1	Nearest. Texel with coordinates nearest the centre of the pixel is used.
= 2	Linear. A weighted linear average of the 2 x 2 array of texels nearest to the centre of the pixel is used. For one-dimensional texture is average of 2 texels. For three dimensional texel is 2 x 2 x 2 array.

I32: Mipmap Minification Filter

Mipmap Minification Filter specifies the texture filtering method to apply when a single pixel on screen maps to a large collection of texels.

Table 25 — Texture Vers-1 Texture Environment Mipmap Minification Filter values

= 0	None.
= 1	Nearest. Texel with coordinates nearest the centre of the pixel is used.
= 2	Linear. A weighted linear average of the 2 x 2 array of texels nearest to the centre of the pixel is used. For one-dimensional texture is average of 2 texels. For three-dimensional texture is 2 x 2 x 2 array.
= 3	Nearest in Mipmap. Within an individual mipmap, the texel with coordinates nearest the centre of the pixel is used.
= 4	Linear in Mipmap. Within an individual mipmap, a weighted linear average of the 2 x 2 array of texels nearest to the centre of the pixel is used. For one-dimensional texture is average of 2 texels. For three-dimensional texture is 2 x 2 x 2 array
= 5	Nearest between Mipmaps. Within each of the adjacent two mipmaps, selects the texel with coordinates nearest the centre of the pixel and then interpolates linearly between these two selected mipmap values.
= 6	Linear between Mipmaps. Within each of the two adjacent mipmaps, computes value based on a weighted linear average of the 2 x 2 array of texels nearest to the centre of the pixel and then interpolates linearly between these two computed mipmap values.

I32: S-Dimen Wrap Mode

S-Dimen Wrap Mode specifies the mode for handling texture coordinates S-Dimension values outside the range [0, 1].

Table 26 — Texture Vers-1 Texture Environment S-Dimen Wrap Mode values

= 0	None.
= 1	Clamp. Any values greater than 1.0 are set to 1.0; any values less than 0.0 are set to 0.0.
= 2	Repeat Integer parts of the texture coordinates are ignored (i.e. retains only the fractional component o texture coordinates greater than 1.0 and only one-minus the fractional component of values less than zero). Resulting in copies of the texture map tiling the surface.
= 3	Mirror Repeat. Like Repeat, except the surface tiles “flip-flop” resulting in an alternating mirror pattern of surface tiles.

= 4	Clamp to Edge. Border is always ignored and instead texel at or near the edge is chosen for coordinates outside the range [0, 1]. Whether the exact nearest edge texel or some average of the nearest edge texels is used is dependent upon the mipmap filtering value.
= 5	Clamp to Border. Nearest border texel is chosen for coordinates outside the range [0, 1]. Whether the exact nearest border texel or some average of the nearest border texels is used is dependent upon the mipmap filtering value.

I32: T-Dimen Wrap Mode

T-Dimen Wrap Mode specifies the mode for handling texture coordinates T-Dimension values outside the range [0, 1]. Same mode values as documented for S-Dimen Wrap Mode.

I32: R-Dimen Wrap Mode

R-Dimen Wrap Mode specifies the mode for handling texture coordinates R-Dimension values outside the range [0, 1]. Same mode values as documented for S-Dimen Wrap Mode.

I32: Blend Type

Blend Type contains information indicating how the values in the texture map are to be modulated/combined/blended with the original colour of the surface or some other alternative colour to compute the final colour to be painted on the surface. Additional information on the interpretation of the Blend Type values and how one might leverage them to render an image can be found in reference [1] listed in the bibliography section.

Table 27 — Texture Vers-1 Texture Environment Blend Type values

= 0	None.
= 1	Decal. Interpret same as OpenGL **GL_DECAL** environment mode.
= 2	Modulate. Interpret same as OpenGL **GL_MODULATE** environment mode.
= 3	Replace. Interpret same as OpenGL **GL_REPLACE** environment mode.
= 4	Blend. Interpret same as OpenGL **GL_BLEND** environment mode.
= 5	Add. Interpret same as OpenGL **GL_ADD** environment mode.
= 6	Combine. Interpret same as OpenGL **GL_COMBINE** environment mode.

I32: Internal Compression Level

Internal Compression Level specifies a data compression hint/recommendation that a JT file loader is free to follow for internally (in memory) storing texel data. This setting does not affect how image texel data is actually stored in JT files or other externally referenced files.

Table 28 — Texture Vers-1 Texture Environment Internal Compression Level values

= 0	None. No compression of texel data.
= 1	Conservative. Lossless compression of texel data.
= 2	Moderate. Texel components truncated to 8-bits each.
= 3	Aggressive. Texel components truncates to 4-bits each (or 5 bits for RGB images).

RGBA: Blend Colour

Blend Colour specifies the colour to be used for the “Blend” mode of Blend Type operations.

RGBA: Border Colour

Border Colour specifies the constant border colour to use for “Clamp to Border” style wrap modes when the texture itself does not have a border.

Mx4F32: Texture Transform

Texture Transform defines the texture coordinate transformation matrix. A renderer of JT data would typically apply this transform to texture coordinates prior to applying the texture.

Texture Coord Generation Parameters

Texture Coord Generation Parameters contains information indicating if and how texture coordinate components should be automatically generated for each of the 4 components (S, T, R, Q) of a texture coordinate.

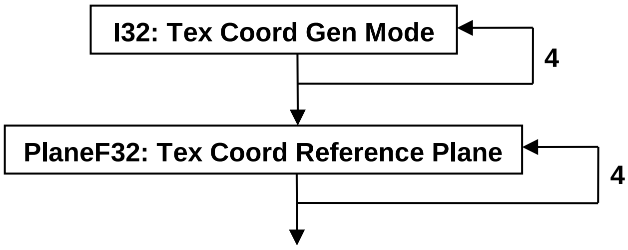


Figure 53 — Texture Coord Generation Parameters data collection

I32: Tex Coord Gen Mode

Tex Coord Gen Mode specifies the texture coordinate generation mode for each component (S, T, R, Q) of texture coordinate. There are four mode values stored, one for each component of texture coordinate. The mode values are stored in S, T, R, Q order.

Table 29 — Texture Vers-1 Texture Coord Generation Gen Mode values

= 0	None. No texture coordinates automatically generated.
= 1	Model Coordinate System Linear. Texture coordinates computed as a distance from a reference plane specified in model coordinates.
= 2	View Coordinate System Linear. Texture coordinates computed as a distance from a reference plane specified in view coordinates.
= 3	Sphere Map. Texture coordinates generated based on spherical environment mapping.
= 4	Reflection Map. Texture coordinates generated based on cubic environment mapping.
= 5	Normal Map. Texture coordinates computed/set by copying vertex normal in view coordinates to S, T, R.

PlaneF32: Tex Coord Reference Plane

Reference Plane specifies the reference plane used for “Model Coordinate System Linear” and “View Coordinate System Linear” texture coordinate generation modes. There are four Reference Planes stored, one for each component of texture coordinate. The Reference Planes are stored in S, T, R, Q order. Even if a components “Tex Coord Gen Mode” is one that does not require a reference plane, dummy reference planes are still stored in JT file.

Inline Texture Image Data

Inline Texture Image Data is a collection of data defining the texture format properties and image texel data for one texture image. Inline Texture Image Data is only present if data field Inline Image Storage Flag equals “1.” If present there will be data field Image Count number of Inline Texture Image Data instances.

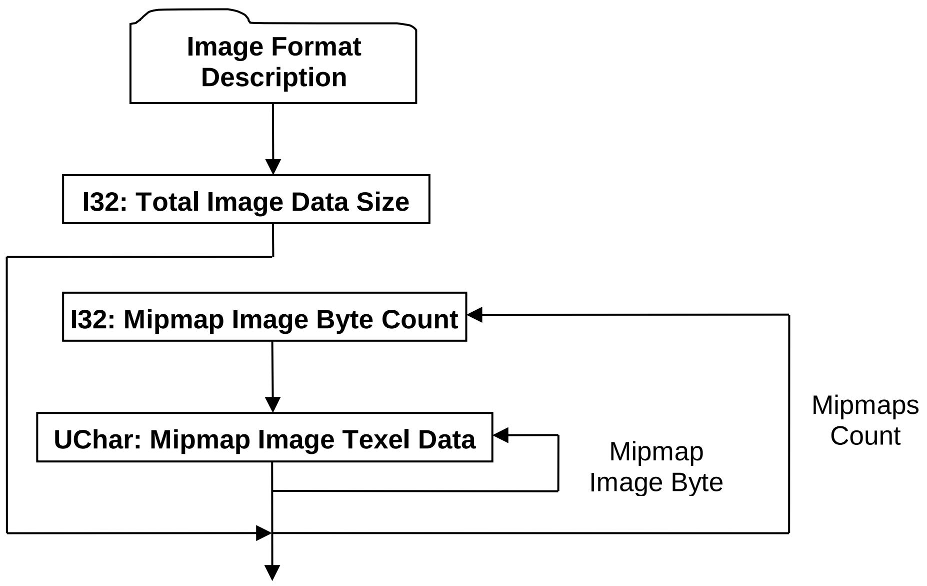


Figure 54 — Inline Texture Image Data collection

Complete description for Image Format Description can be found in Image Format Description.

I32: Total Image Data Size

Total Image Data Size specifies the total length, in bytes, of the on-disk representation for all mipmap images. This byte total does not include the I32: Mipmap Image Byte Count data field storage (4 bytes per) for each mipmap.

I32: Mipmap Image Byte Count

Mipmap Image Byte Count specifies the length, in bytes, of the on-disk representation of the next mipmap image.

UChar: Mipmap Image Texel Data

Mipmap Image Texel Data is the mipmap's block of image data. The length of this field in bytes is specified by the value of data field Mipmap Image Byte Count.

Image Format Description

The Image Format Description is a collection of data defining the pixel format, data type, size, and other miscellaneous characteristics of the texel image data.

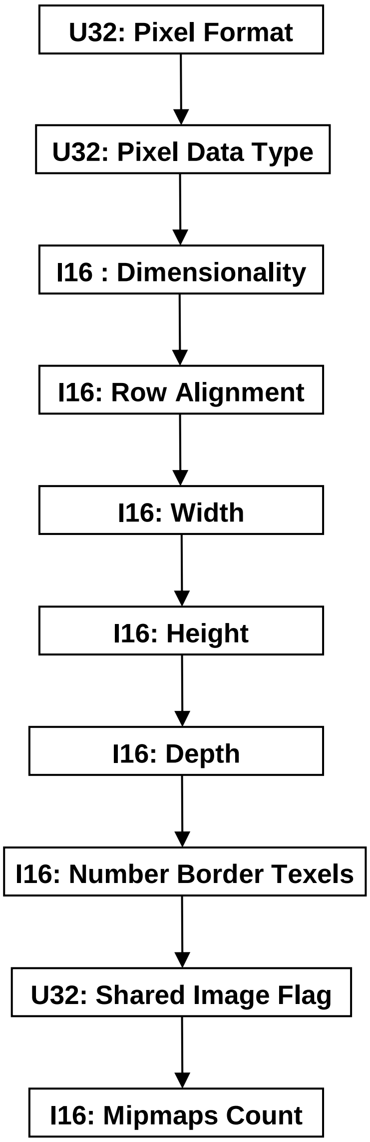


Figure 55 — Image Format Description data collection

U32: Pixel Format

Pixel format specifies the format of the texture image pixel data. Depending on the format, anywhere from one to four elements of data exists per texel.

Table 30 — Texture Vers-1 Image Format Description Pixel Format values

= 0	No format specified. Texture mapping is not applied.
= 1	RGB: A red colour component followed by green and blue colour components
= 2	RGBA: A red colour component followed by green, blue, and alpha colour components
= 3	LUM: A single luminance component
= 4	LUMA: A luminance component followed by an alpha colour component
= 5	A single stencil index
= 6	A single depth component
= 7	A single red colour component
= 8	A single green colour component
= 9	A single blue colour component

= 10	A single alpha colour component
= 11	A blue colour component, followed by green and red colour components
= 12	A blue colour component, followed by green, red, and alpha colour components
= 13	A depth component, followed by a stencil component

U32: Pixel Data Type

Pixel Data Type specifies the data type used to store the per texel data. If the Pixel Format represents a multi component value (e.g. red, green, blue) then each value requires the Pixel Data Type number of bytes of storage (e.g. a Pixel Format Type of "1" with Pixel Data Type of "3" would require 3 bytes of storage for each texel).

Table 31 — Texture Vers-1 Image Format Description Pixel Data values

= 0	No type specified. Texture mapping is not applied.
= 1	Signed 8-bit integer
= 2	Single-precision 32-bit floating point
= 3	Unsigned 8-bit integer
= 4	Single bits in unsigned 8-bit integers
= 5	Unsigned 16-bit integer
= 6	Signed 16-bit integer
= 7	Unsigned 32-bit integer
= 8	Signed 32-bit integer
= 9	16-bit floating point according to IEEE-754 format (i.e. 1 sign bit, 5 exponent bits, 10 mantissa bits)

I16 : Dimensionality

Dimensionality specifies the number of dimensions the texture image has. Valid values include:

Table 32 — Texture Vers-1 Image Format Description Dimensionality values

= 1	One-dimensional texture
= 2	Two-dimensional texture
= 3	Three-dimensional texture

I16: Row Alignment

Row Alignment specifies the byte alignment for image data rows. This data field shall have a value of 1, 2, 4, or 8. If set to 1 then all bytes are used (i.e. no bytes are wasted at end of row). If set to 2, then if necessary, an extra wasted byte(s) is/are stored at the end of the row so that the first byte of the next row has an address that is a multiple of 2 (multiple of four for Row Alignment equal 4 and multiple of 8 for row alignment equal 8). The actual formula (using C syntax) to determine number of bytes per row is as follows:

$$\mathrm{BytesPerRow} = (\mathrm{numBytesPerPixel} * \mathrm{ImageWidth} + \mathrm{RowAlignment} - 1) \ \& \ \sim(\mathrm{RowAlignment} - 1)$$

I16: Width

Width specifies the width dimension (number of texel columns) of the texture image in number of pixels.

I16: Height

Height specifies the height dimension (number of texel rows) of the texture image in number of pixels. Height is 1 for one-dimensional images.

I16: Depth

Depth specifies the depth dimension (number of texel slices) of the texture image in number of pixels. Depth is 1 for one-dimensional and two-dimensional images.

I16: Number Border Texels

Number Border Texels specifies the number of border texels in the texture image definition. Valid values are 0 and 1.

U32: Shared Image Flag

Shared Image Flag is a flag indicating whether this texture image is shareable with other Texture Image Element attributes.

Table 33 — Texture Vers-1 Image Format Description Shared Image Flag values

= 0	Image is not shareable with other Texture Image Elements.
= 1	Image is shareable with other Texture Image Elements.

I16: Mipmaps Count

Mipmaps Count specifies the number of mipmap images. A value of 1 indicates that no mipmaps are used. A value greater than 1 indicates that mipmaps are present all the way down to a 1-by-1 texel.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

7.2.3 Draw Style Attribute Element

Object Type ID: 0x10dd1014, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Draw Style Attribute Element contains information defining various aspects of the graphics state/style that should be used for rendering associated geometry. JT format LSG traversal semantics state that draw style attributes accumulate down the LSG by replacement.

The Field Inhibit flag (see Base Attribute Data) bit assignments for the Draw Style Attribute Element data fields, are as follows:

Table 34 — Draw Style Attribute Field Inhibit flag values

Field Inhibit Flag Bit	Data Field(s) Bit Applies To
0	Two Sided Lighting Flag
1	Back-face Culling Flag
2	Outlined Polygons Flag
3	Lighting Enabled Flag
4	Flat Shading Flag
5	Separate Specular Flag

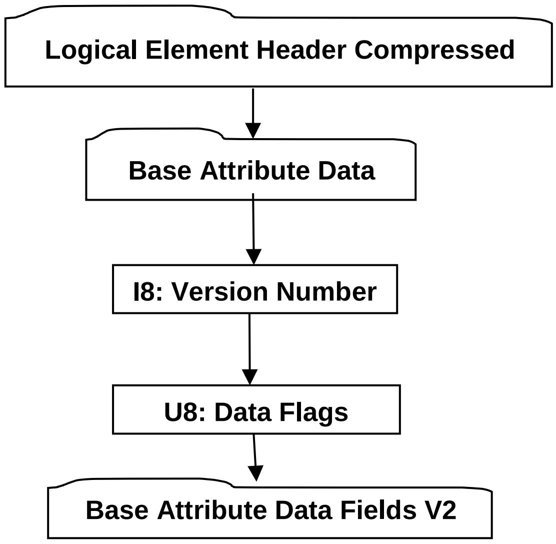


Figure 56 — Draw Style Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers. Only version 1 is defined for JT V10.5.

U8: Data Flags

Data Flags is a collection of flags. The flags are combined using the binary OR operator and store various state settings for Draw Style Attribute Elements. All bits fields that are not defined as in use should be set to “0”.

Table 35 — Draw Style Attribute Data Flag values

0x01	Back-face Culling Flag. Indicates if back-facing polygons should be discarded (culled). = 0 – Back-facing polygons not culled. = 1 – Back-facing polygons culled.
0x02	Two Sided Lighting Flag. Indicates if two sided lighting should be enabled to insure that polygons are illuminated on both sides. = 0 – Disable two sided lighting. = 1 – Enable two sided lighting.
0x04	Outlined Polygons Flag. Indicates if polygons should be draw as “wireframes” i.e. not filled. = 0 – Polygons drawn as filled. = 1 – Only polygon’s outline drawn.
0x08	Lighting Enabled Flag. Indicates if lighting should be enabled. If lighting disabled, then renderer should perform no calculations concerning normals, light sources, material properties, etc. = 0 – Disable lighting. = 1 – Enable lighting.

0x10	Flat Shading Flag. Indicates if the geometry should be rendered with single colour (flat shading) or with many different colour (smooth/Gouraud) shading. = 0 – Disable flat shading (i.e. use smooth/Gouraud shading). = 1 – Enable flat shading.
0x20	Separate Specular Flag. Indicates if the application of the specular colour should be delayed until after texturing. If no texture mapping then this flag setting is irrelevant. = 0 – Apply specular colour contribution before texture mapping. = 1 – Apply specular colour contribution after texture mapping.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

Palette Index default value is -1

7.2.4 Light Set Attribute Element

Object Type ID: 0x10dd1096, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Light Set Attribute Element holds an unordered list of Lights. JT format LSG traversal semantics state that light set attributes accumulate down the LSG through addition of lights to an attribute list.

Light Set Attribute Element does not have any Field Inhibit flag (see Base Attribute Data) bit assignments.

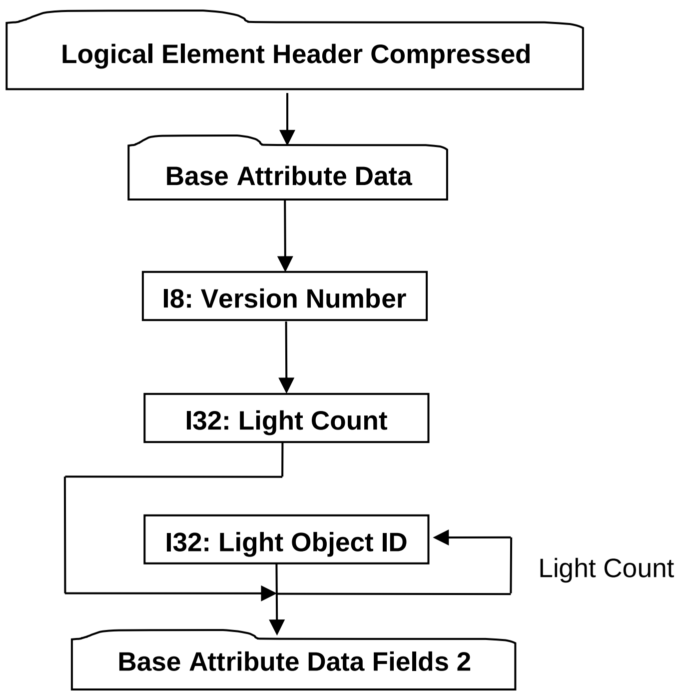


Figure 57 — Light Set Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers. Only version 1 is defined for JT V10.5.

I32: Light Count

Light Count specifies the number of lights in the Light Set.

I32: Light Object ID

Light Object ID is the identifier for a referenced Light Object.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

7.2.5 Linestyle Attribute Element

Object Type ID: 0x10dd10c4, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Linestyle Attribute Element contains information defining the graphical properties to be used for rendering polylines. JT format LSG traversal semantics state that Linestyle attributes accumulate down the LSG by replacement.

Linestyle Attribute Element does not have any Field Inhibit flag (see Base Attribute Data) bit assignments.

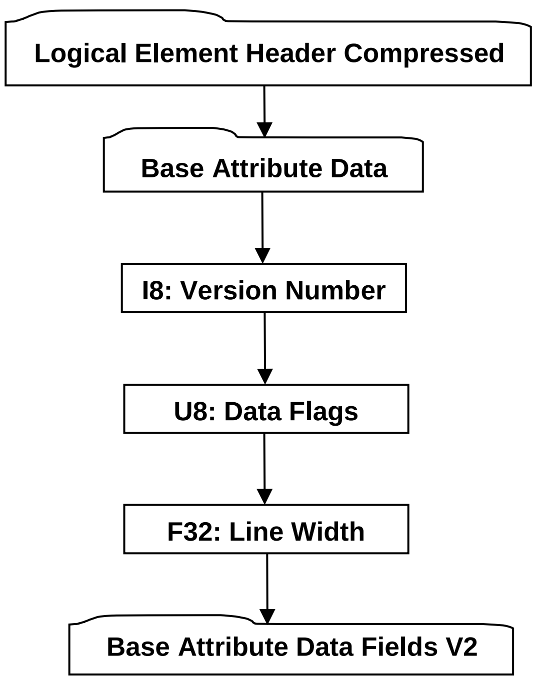


Figure 58 — Linestyle Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U8: Data Flags

Data Flags is a collection of flags and line type data. The flags and line type data are combined using the binary OR operator and store various polyline rendering attributes. All bits fields that are not defined as in use should be set to “0”.

Table 36 — Linestyle Attribute Data Flag values

0x0F	Line Type (stored in bits 0 – 3 or in binary notation 00001111) Line type specifies the polyline rendering stipple-pattern. = 0 - Solid = 1 – Dash = 2 – Dot = 3 – Dash_Dot = 4 – Dash_Dot_Dot = 5 – Long_Dash = 6 – Centre_Dash = 7 – Centre_Dash_Dash
0x10	Antialiasing Flag (stored in bit 4 or in binary notation 00010000) Indicates if antialiasing should be applied as part of rendering polylines. = 0 – Antialiasing disabled. = 1 – Antialiasing enabled.

F32: Line Width

Line Width specifies the width in pixels that should be used for rendering polylines. The value of this field shall be greater than 0.0.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

7.2.6 Pointstyle Attribute Element

Object Type ID: 0x8d57c010, 0xe5cb, 0x11d4, 0x84, 0xe, 0x00, 0xa0, 0xd2, 0x18, 0x2f, 0x9d

Pointstyle Attribute Element contains information defining the graphical properties that should be used for rendering points. JT format LSG traversal semantics state that Pointstyle attributes accumulate down the LSG by replacement.

Pointstyle Attribute Element does not have any Field Inhibit flag (see Base Attribute Data) bit assignments.

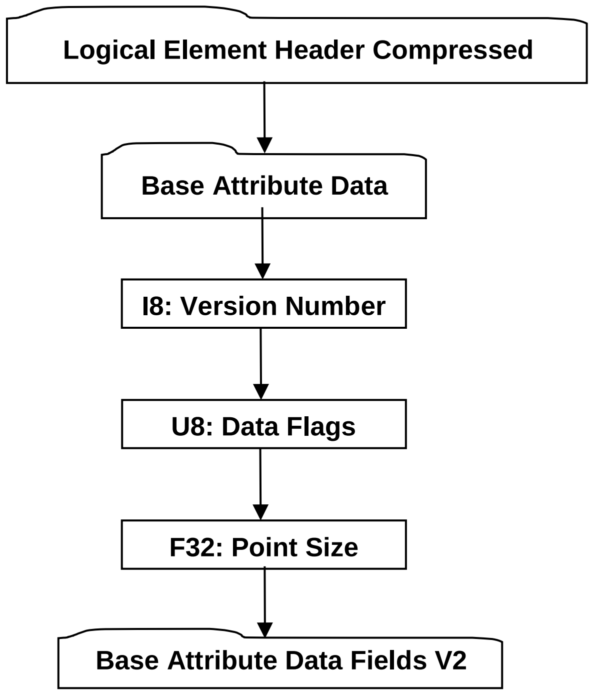


Figure 59 — Pointstyle Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U8: Data Flags

Data Flags is a collection of flags and point type data. The flags and point type data are combined using the binary OR operator and store various point rendering attributes. All bits fields that are not defined as in use should be set to “0”.

Table 37 — Pointstyle Attribute Data Flag values

0x0F	Point Type (stored in bits 0 – 3 or in binary notation 00001111) These bits are reserved for future expansion of the format to support Point Types.
0x10	Antialiasing Flag (stored in bit 4 or in binary notation 00010000) Indicates if antialiasing should be applied as part of rendering points. = 0 – Antialiasing disabled. = 1 – Antialiasing enabled.

F32: Point Size

Point Size specifies the size in pixels that should be used for rendering points. The value shall be greater than 0.0.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

7.2.7 Geometric Transform Attribute Element

Object Type ID: 0x10dd1083, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Geometric Transform Attribute Element contains a 4x4 homogeneous transformation matrix that positions the associated LSG node's coordinate system relative to its parent LSG node. JT format LSG traversal semantics state that geometric transform attributes accumulate down the LSG through matrix multiplication as follows:

p' = pAM

Where *p* is a point of the model, *p'* is the transformed point, *M* is the current modelling transformation matrix inherited from ancestor LSG nodes and previous Geometric Transform Attribute Element, and *A* is the transformation matrix of this Geometric Transform Attribute Element. The matrix is allowed to contain translation, rotation, and uniform- and non-uniform scaling factors, including negative scales. It is not allowed to contain shearing or projective components, or scaling factors of zero (which would make the matrix singular).

Geometric Transform Attribute Element does not have any Field Inhibit flag (see Base Attribute Data) bit assignments.

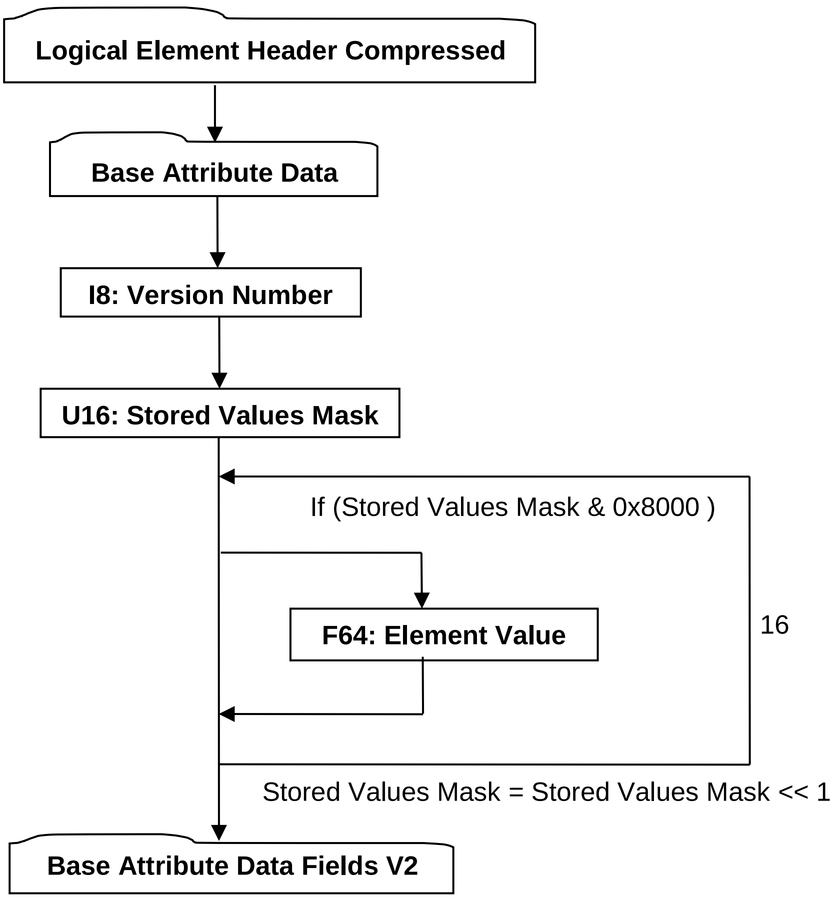


Figure 60 — Geometric Transform Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8: Version Number

Version Number is the version identifier for this node. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U16: Stored Values Mask

Stored Values mask is a 16-bit mask where each bit is a flag indicating whether the corresponding element in the matrix is different from the identity matrix. Only elements which are different from the identity matrix are actually stored. The bits are assigned to matrix elements as follows:

Bit15	Bit14	Bit13	Bit12
Bit11	Bit10	Bit9	Bit8
Bit7	Bit6	Bit5	Bit4
Bit3	Bit2	Bit1	Bit0

The individual bit-flag values are interpreted as follows:

Table 38 — Geometric Transform Attribute Stored Value Mask individual bit-flag values

= 0	Value not stored (matrix value same as corresponding element in identity matrix)
= 1	Value stored

F64: Element Value

Element Value specifies a particular matrix element value.

Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 are defined when Base Attribute Version Number is set to two.

7.2.8 Palette Map Attribute Element

Object Type ID: 0x10dd1106,0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Palette Map Attribute is used on a shape such that any face group can be rendered with a chosen entry from the palette. Each Attribute entry in the palette is inherited down the scene graph independent from any other palette entry. Attributes Elements in the scene graph are able to specify to which palette entry they apply.

For a complete description of per face group attributes and palette map se the Perface Group Attributes Annex in this document.

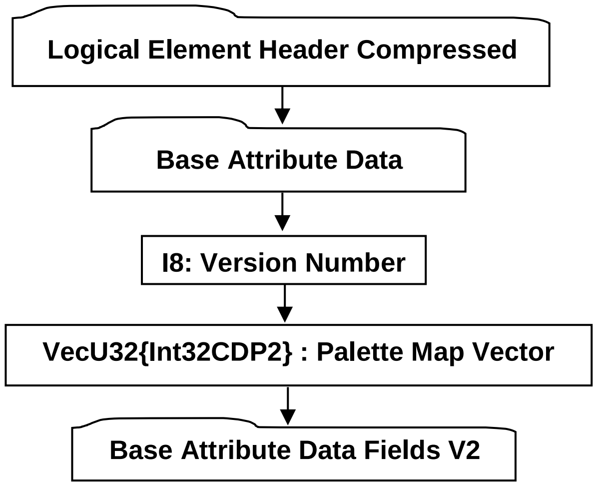


Figure 61 — PaletteMap Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

A complete description of Base Attribute Data can be found in the LSG Segment section of this document under Attribute Elements in the logical collection describing Base Attribute Data.

I8:Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

VecU32{Int32CDP2}: Palette Map Vector

A Palette Map is a set of accumulated states belonging to an instance of a node in the LSG. Each state in a palette is accumulated by different levels and types of attributes with the same palette index.

The Palette Map Vector contains a mapping vector indexed by face group number storing the palette index of the state to be assigned to that face group.A Palette Map entry of -1 indicates that the "fallback" state is to be used for the corresponding face group. A Palette Map entry of -2 indicates that the corresponding face group is to be inhibited entirely, and not rendered at all.

For a complete description of this topic see the Perface Group Attributes Annex in this document.

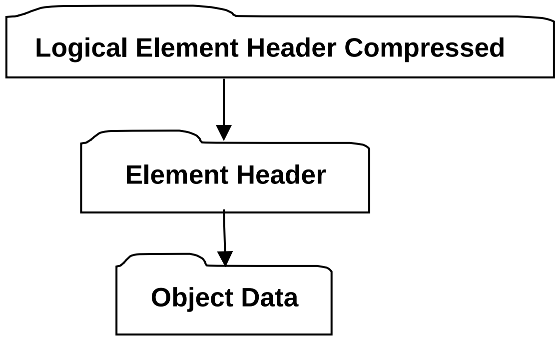
Base Attribute Data Fields V2

See Common Data Attribute Containers for Attribute Elements. Base Attribute Data Fields V2 must be set to -1 for the PaletteMap Attribute element.

7.2.9 Sabot V104 Attribute Element

Object Type ID: `0x96603dd3,0x5a0f,0x40f5,0x97,0xcb,0x1e,0x96,0x47,0xcb,0xd3,0x7e);`

Sabot V104 is used to insulate pre JT V10.5 readers from attributes with non-fallback pallet Index attributes in order to preserve forward compatibility.



A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

The complete description for Element Header can also be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Data Object

The Object Data contained in the Sabot V104 object is an attribute element. The Sabot V104 object exists only to insulate pre edition 3 readers from the contained attribute element. The attribute element contained is wrapped in a Sabot V104 attrbute because its palette index is not -1.Older readers must ignore such attributes to function as they did with previous version of JT.

Each time a Sabot V104 object is read, the Sabot V104 object must be discarded and the wrapped attribute read in its place. A Sabot V104 object never appears in the scenegraph, only in the persisted JT file.

When writing an JT V10.5 file, all attribtues having Palettet Index not set to -1 must be wrapped by a Sabot V104 attribute.

7.2.10 Infinite Light Attribute Element

Object Type ID: 0x10dd1028, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Infinite Light Attribute Element specifies a light source emitting unattenuated light in a single direction from every point on an infinite plane. The infinite location indicates that the rays of light can be considered parallel by the time they reach an object.

JT format LSG traversal semantics state that infinite light attributes accumulate down the LSG through addition of lights to an attribute list.

Infinite Light Attribute Element does not have any Field Inhibit flag (see Base Attribute Data) bit assignments.

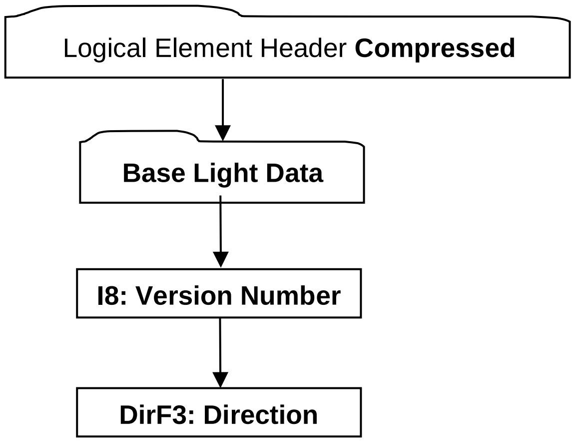


Figure 62 — Infinite Light Attribute Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Light Data can be found in Base Light Data.

I8: Version Number

Version Number is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow.

Table 39 — Light Set Attribute Version Number values

= 1	Version-1 Format

DirF3: Direction

Direction specifies the direction the light is pointing in.

Base Light Data

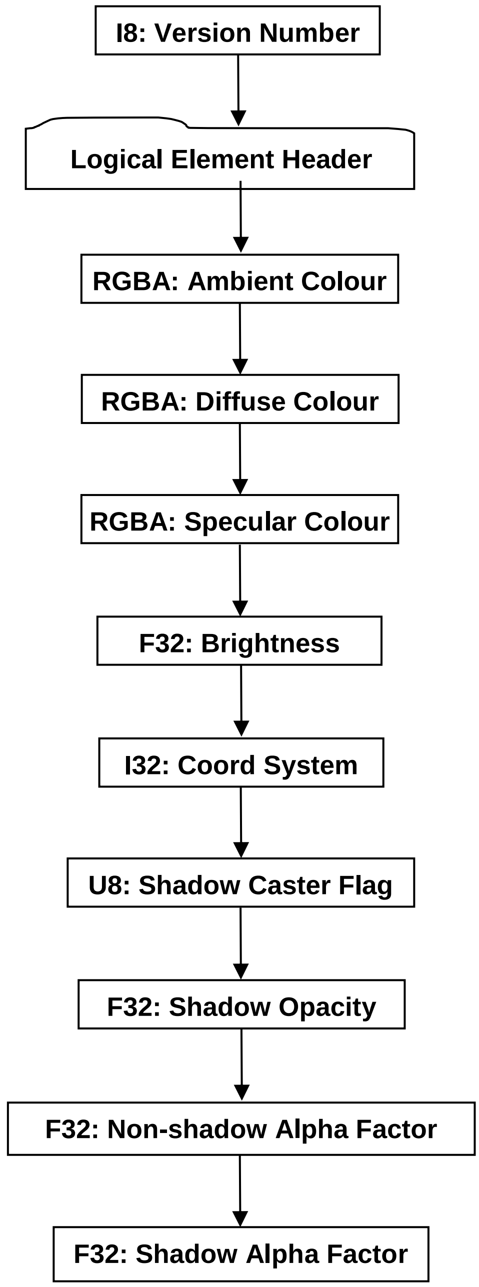


Figure 63 — Base Light Data collection

I8: Version Number

Version number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers. Only version 1 is defined for JT V10.5.

Logical Element Header

A complete description of Logical Element Header can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

RGBA: Ambient Colour

Ambient Colour specifies the ambient red, green, blue, alpha colour values of the light.

RGBA: Diffuse Colour

Diffuse Colour specifies the diffuse red, green, blue, alpha colour values of the light.

RGBA: Specular Colour

Specular Colour specifies the specular red, green, blue, alpha colour values of the light.

F32: Brightness

Brightness specifies the Light brightness. The Brightness value shall be greater than or equal to “-1”.

I32: Coord System

Coord System specifies the coordinate space in which Light source is defined. Valid values include the following:

Table 40 — Base Light Data Cood System values

= 1	Viewpoint Coordinate System. Light source is to move together with the viewpoint.t
= 2	Model Coordinate System. Light source is affected by whatever model transforms that are current when the light source is encountered in LSG.
= 3	World Coordinate system. Light source is not affected by model transforms in the LSG.

U8: Shadow Caster Flag

Shadow Caster Flag is a flag that indicates whether the light is a shadow caster or not.

Table 41 — Base Light Data Shadow Caster Flag values

= 0	Light source is not a shadow caster.
= 1	Light source is a shadow caster.

F32: Shadow Opacity

Shadow Opacity specifies the shadow opacity factor on Light source. Value shall be within range [0.0, 1.0] inclusive. Shadow Opacity is intended to convey how dark a shadow cast by this light source are to be rendered. A value of 1.0 means no light from this light source reaches a shadowed surface, resulting in a black shadow.

F32: Non-shadow Alpha Factor

Non-shadow Alpha Factor is one of a matched pair of fields intended to govern how a shadowing light source (one whose Shadow Caster Flag is set) casts "alpha light" into areas that it directly illuminates (i.e. are not in shadow). Those fragments directly lit by this light source will have their alpha values scaled by Non-shadow Alpha Factor. Non-shadow Alpha Factor value shall lie on the range [0.0, 1.0] inclusive.

This field can be used to create "drop shadows" by setting its value to 0. The effect being that all geometry illuminated by the light source will be "burned away," leaving behind only those parts lying in shadow. Naturally, implementing this intended behaviour implies extensive viewer support.

F32: Shadow Alpha Factor

Shadow Alpha Factor is one of a matched pair of fields intended to govern how a shadowing light source (one whose Shadow Caster Flag is set) casts "alpha light" into areas that it does not illuminate (i.e. are in shadow). Those fragments in shadow from this light source will have their alpha values scaled by Shadow Alpha Factor. Shadow Alpha Factor value shall lie on the range [0.0, 1.0] inclusive.

This field has the opposite effect of Non-shadow Alpha Factor. If set to a value of 0, for example, it will cause all geometry shadowed from the light source to be burned away, leaving behind only those parts directly illuminated by the light source. Naturally, implementing this intended behaviour implies extensive viewer support.

7.2.11 Point Light Attribute Element

Object Type ID: 0x10dd1045, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Point Light Attribute Element specifies a light source emitting light from a specified position, along a specified direction, and with a specified spread angle.

JT format LSG traversal semantics state that point light attributes accumulate down the LSG through addition of lights to an attribute list.

Point Light Attribute Element does not have any Field Inhibit flag (see Base Attribute Data) bit assignments.

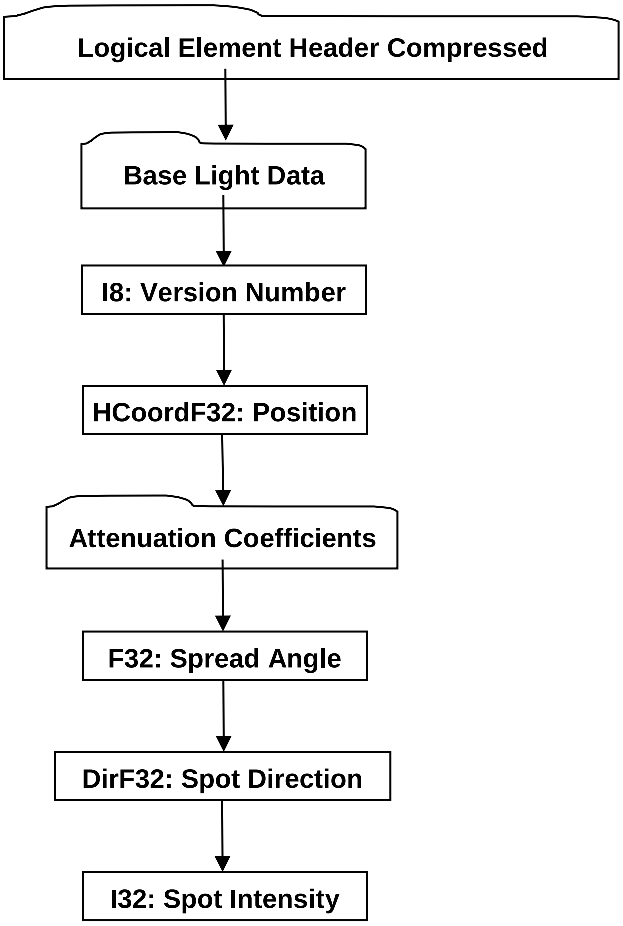


Figure 64 — Point Light Attribute Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

Complete description for Base Light Data can be found in Base Light Data.

Complete description for Attenuation Coefficients can be found in Attenuation Coefficients.

I8: Version Number

Version Number is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow.

Table 42 — Point Light Attribute Version Number values

= 1	Version-1 Format

HCoordF32: Position

Position specifies the light position in homogeneous coordinates.

F32: Spread Angle

Spread Angle, as shown in the figure below with respect to the light cone, specifies in degrees the half angle of the light cone. Valid Spread Angle values are clamped and interpreted as follows:

Table 43 — Point Light Attribute Spread Angle values

angle = = 180.0	Simple point light
0.0 >= angle <= 90.0	Spot Light

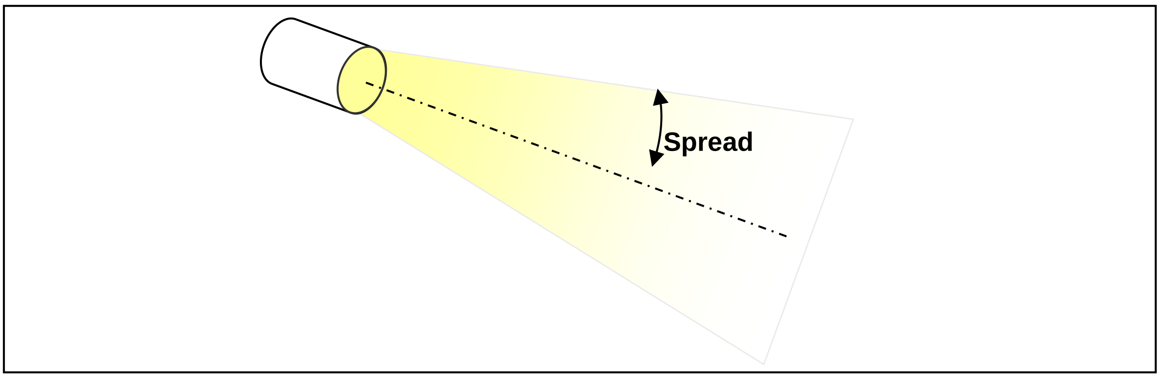


Figure 65 — Spread Angle value with respect to the light cone

DirF32: Spot Direction

Spot Direction specifies the direction the spot light is pointing in.

I32: Spot Intensity

Spot Intensity specifies the intensity distribution of the light within the spot light cone. Spot Intensity is really a “spot exponent” in a lighting equation and indicates how focused the light is at the centre. The larger the value, the more focused the light source. Only non-negative Spot intensity values are valid.

Attenuation Coefficients

Attenuation Coefficients data collection contains the coefficients for how light intensity decreases with distance.

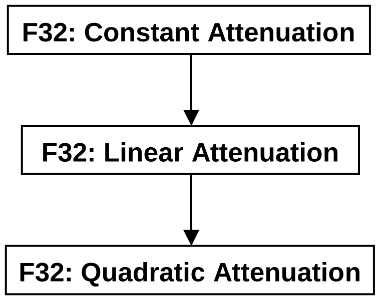


Figure 66 — Attenuation Coefficients data collection

F32: Constant Attenuation

Constant Attenuation specifies the constant coefficient for how light intensity decreases with distance. Value shall be greater than or equal to “0”.

F32: Linear Attenuation

Linear Attenuation specifies the linear coefficient for how light intensity decreases with distance. Value shall be greater than or equal to “0”.

F32: Quadratic Attenuation

Quadratic Attenuation specifies the quadratic coefficient for how light intensity decreases with distance. Value shall be greater than or equal to “0”.

7.2.12 Mapping Plane Element

Object Type ID: 0xa3cfb921, 0xbdeb, 0x48d7, 0xb3, 0x96, 0x8b, 0x8d, 0xe, 0xf4, 0x85, 0xa0

Mapping Plane Element defines the mapping plane for texture coordinate generation.

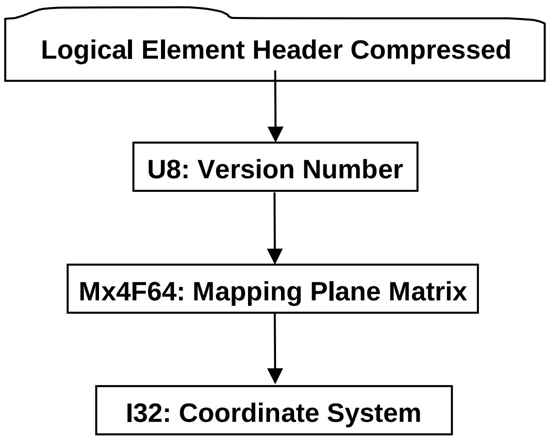


Figure 67 — Mapping Plane Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

U8: Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

Mx4F64: Mapping Plane Matrix

Mx4F64: Mapping Plane Matrix specifies the transformation matrix and mapping parameters for the mapping plane. The transformation matrix defines the mapping coordinate system transformed from I32: Coordinate System. The mapping parameters specifies the width and height of the mapping plane. The mapping plane is defined in the + xy-plane of the mapping coordinate system. In the mapping process, the geometry vertex coordinates in Model Coordinate System are transformed to the mapping coordinate system at first, and then the transformed vertex coordinates are mapped to texture coordinates as following:

s-coordinate = x-coordinate of the transformed vertex / the width of the mapping plane
t-coordinate = y-coordinate of the transformed vertex / the height of the mapping plane

I32: Coordinate System

Coordinate System specifies the coordinate space in which mapping plane is defined. Valid values include the following:

Table 44 — Mapping Plane Matrix Coordinate System values

= 0	Undefined Coordinate System.
= 1	Viewpoint Coordinate System. Mapping plane is to move together with the viewpoint.
= 2	Model Coordinate System. Mapping plane is affected by whatever model transforms that are current when the mapping plane is encountered in LSG.
= 3	World Coordinate system. Mapping plane is not affected by model transforms in the LSG.

7.2.13 Mapping Cylinder Element

Object Type ID: 0x3e70739d, 0x8cb0, 0x41ef, 0x84, 0x5c, 0xa1, 0x98, 0xd4, 0x0, 0x3b, 0x3f

Mapping Cylinder Element defines the mapping cylinder for texture coordinate generation.

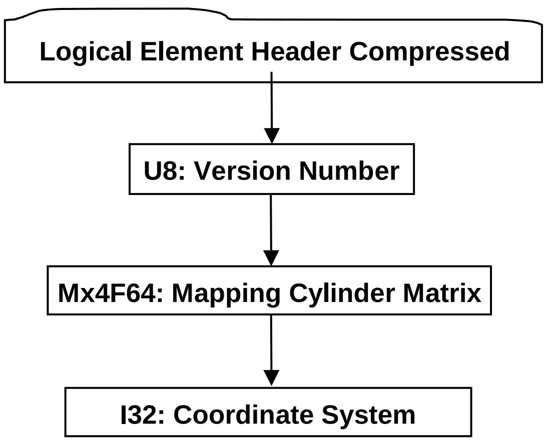


Figure 68 — Mapping Cylinder Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

U8: Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

Mx4F64: Mapping Cylinder Matrix

Mx4F64: Mapping Cylinder Matrix specifies the transformation matrix and mapping parameters for the mapping cylinder. The transformation matrix defines the mapping coordinate system transformed from I32: Coordinate System. The mapping parameters specifies the horizontal sweep angle and height of the mapping cylinder. The mapping cylinder's axis is parallel to the z-axis of the mapping coordinate system, and the horizontal sweep angle starts from the +x-axis in a counter clockwise direction. In the mapping process, the geometry vertex coordinates in Model Coordinate System are transformed to the mapping coordinate system at first, and then the transformed vertex coordinates are mapped to texture coordinates as following:

s-coordinate = the horizontal sweep angle of the vertex / the horizontal sweep angle of the mapping cylinder

t-coordinate = the z-coordinate of the vertex / height of the mapping cylinder

Mapping Cylinder Element implements the strategy to handle texture coordinates who cross the seam of the texture in the mapping process.

I32: Coordinate System

Coordinate system specifies the coordinate space in which mapping cylinder is defined. Valid values include the following:

Table 45 — Mapping Cylinder Matrix Coordinate System values

= 0	Undefined Coordinate System.
= 1	Viewpoint Coordinate System. Mapping cylinder is to move together with the viewpoint.
= 2	Model Coordinate System. Mapping cylinder is affected by whatever model transforms that are current when the mapping cylinder is encountered in LSG.
= 3	World Coordinate system. Mapping cylinder is not affected by model transforms in the LSG.

7.2.14 Mapping Sphere Element

Object Type ID: 0x72475fd1, 0x2823, 0x4219, 0xa0, 0x6c, 0xd9, 0xe6, 0xe3, 0x9a, 0x45, 0xc1

Mapping Sphere Element defines the mapping sphere for texture coordinate generation.

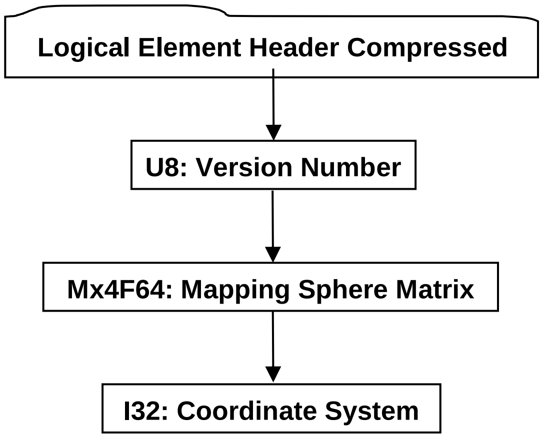


Figure 69 — Mapping Sphere Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

U8: Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

Mx4F64: Mapping Sphere Matrix

Mx4F64: Mapping Sphere Matrix specifies the transformation matrix and mapping parameters of the mapping sphere. The transformation matrix defines the mapping coordinate system transformed from I32: Coordinate System. The mapping parameters specify the horizontal sweep angle and vertical sweep angle of the mapping sphere. The mapping sphere's centre is at the origin of the mapping coordinate system, and the poles of the sphere are parallel to the z-axis of the coordinate system. The horizontal sweep angle starts from the +x-axis in a counter clockwise direction, and the vertical sweep angle is from the +z-axis to the –z-axis. In the mapping process, the geometric vertex coordinates in Model Coordinate System are transformed to the mapping coordinate system at first, and then the transformed vertex coordinates are mapped to texture coordinates as following:

s-coordinate = the horizontal sweep angle of the vertex / the horizontal sweep angle of the mapping sphere

t-coordinate = the vertical sweep angle of the vertex / the vertical sweep angle of the mapping sphere

I32: Coordinate System

Coordinate system specifies the coordinate space in which mapping sphere is defined. Valid values include the following:

Table 46 — Mapping Sphere Matrix Coordinate System values

= 0	Undefined Coordinate System.
= 1	Viewpoint Coordinate System. Mapping sphere is to move together with the viewpoint.
= 2	Model Coordinate System. Mapping sphere is affected by whatever model transforms that are current when the mapping sphere is encountered in LSG.
= 3	World Coordinate system. Mapping sphere is not affected by model transforms in the LSG.

7.2.15 Mapping TriPlanar Element

Object Type ID: 0x92f5b094, 0x6499, 0x4d2d, 0x92, 0xaa, 0x60, 0xd0, 0x5a, 0x44, 0x32, 0xcf

Mapping TriPlanar Element defines the mapping triplanar surface for texture coordinate generation.

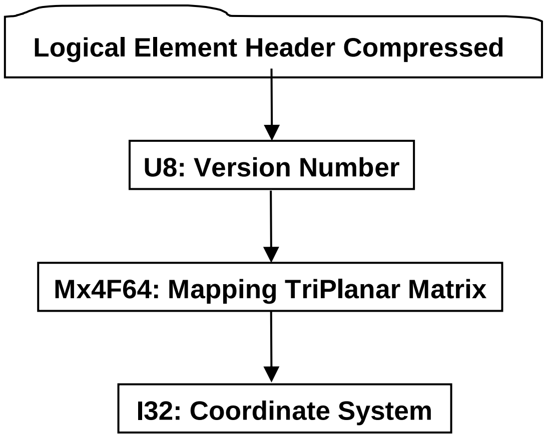


Figure 70 — Mapping TriPlanar Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

U8: Version Number

Version Number is the version identifier for this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

Mx4F64: Mapping TriPlanar Matrix

Mx4F64: Mapping TriPlanar Matrix specifies the transformation matrix and mapping parameter for the mapping triplanar. The transformation matrix defines the mapping coordinate system transformed from I32: Coordinate System. The mapping parameter specifies the planar length of the triplanar. The left bottom corner of the triplanar is located at the origin of the mapping coordinate system, and the three planes are in the + xy-plane, + yz-plane, and + xz-plane respectively. In the mapping process, the geometry vertex coordinates in Model Coordinate System are transformed to the mapping coordinate system at first, and then the transformed vertex coordinates are projected to the corresponding plane based on the maximum component of its normals, and at last the projected vertex coordinates are mapped to texture coordinates as following:

s-coordinate = the first-coordinate of the projected vertex / the planar length of the triplanar

t-coordinate = the second-coordinate of the projected vertex / the planar length of the triplanar

I32: Coordinate System

Coordinate system specifies the coordinate space in which mapping triplanar surface is defined. Valid values include the following:

Table 47 — Mapping TriPlanar Matrix Coordinate System values

= 0	Undefined Coordinate System.
= 1	Viewpoint Coordinate System. Mapping triplanar surface is to move together with the viewpoint.
= 2	Model Coordinate System. Mapping triplanar surface is affected by whatever model transforms that are current when the mapping triplanar surface is encountered in LSG.

= 3	World Coordinate system. Mapping triplanar surface is not affected by model transforms in the LSG.

7.3 Property Atom Elements

Property Atom Elements are meta-data objects associated with nodes or Attributes. Property Atom Elements are not nodes or attributes themselves, but can be associated with any node or Attribute to maintain arbitrary application- or enterprise information (meta-data) pertaining to that node or Attribute. Each Node Element or Attribute Element in an LSG may hold zero or more Property Atom Elements and this relationship information is stored within Property Table section of a JT file.

An individual property is specified as a *key/value* Property Atom Element pair, where the *key* identifies the type and meaning of the *value*. The JT format supports many different Property Atom Element key/value object types. The different Property Atom Element key/value object types are documented in the following subsections.

Some "Best Practices" for placing application or enterprise properties/meta-data on Nodes in JT files can be found in Metadata Conventions section of this reference.

7.3.1 Base Property Atom Element

Object Type ID: 0x10dd104b, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Base Property Atom Element represents the simplest form of a property that can exist within the LSG and has no type specific value data associated with it.

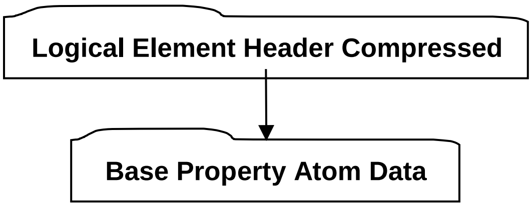


Figure 71 — Base Property Atom Element data collection

Complete description for Logical Element Header Compressed can be found in Logical Element Header Compressed in this document.

Base Property Atom Data

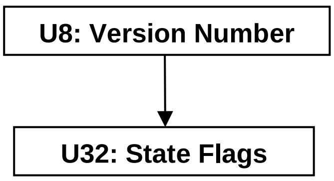


Figure 72 — Base Property Atom Data collection

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U32: State Flags

State Flags is a collection of flags. The flags are combined using the binary OR operator and store various state information for property atoms. Bits 0 – 7 are freely available for an application to store whatever property atom information desired. The topmost 0x40000000 bit must be set to 1 for general viewing support. All other bits are reserved for future expansion of the file format.

7.3.2 String Property Atom Element

Object Type ID: 0x10dd106e, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

String Property Atom Element represents a character string property atom.

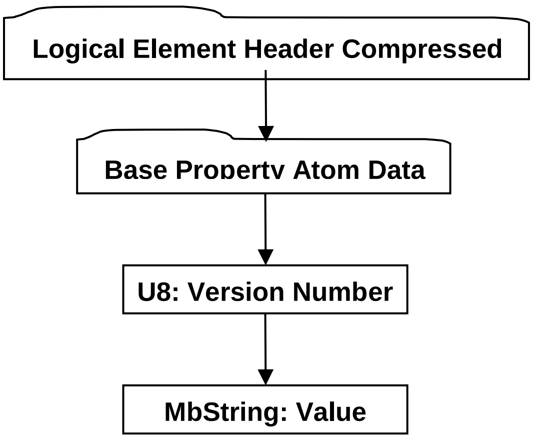


Figure 73 — String Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

MbString: Value

Value contains the character string value for this property atom.

7.3.3 Integer Property Atom Element

Object Type ID: 0x10dd102b, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Integer Property Atom Element represents a property atom whose value is of I32 data type.

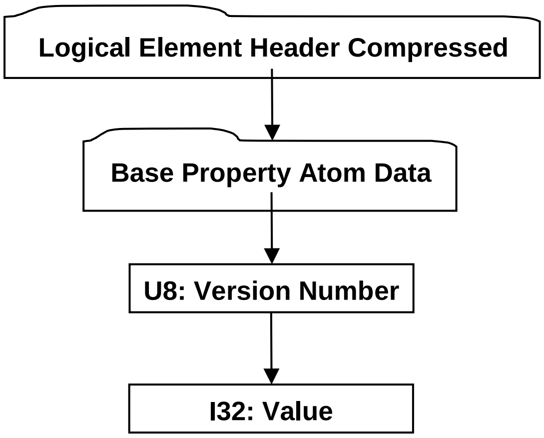


Figure 74 — Integer Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: Value

Value contains the integer value for this property atom.

7.3.4 Floating Point Property Atom Element

Object Type ID: 0x10dd1019, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

Floating Point Property Atom Element represents a property atom whose value is of F32 data type.

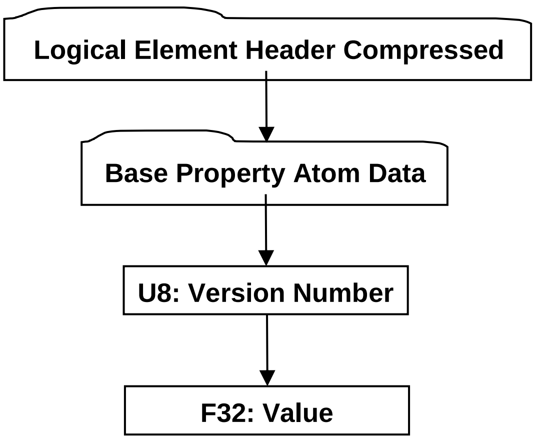


Figure 75 — Floating Point Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

F32: Value

Value contains the floating point value for this property atom.

7.3.5 JT Object Reference Property Atom Element

Object Type ID: 0x10dd1004, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

JT Object Reference Property Atom Element represents a property atom whose value is an object ID for another object within the JT file.

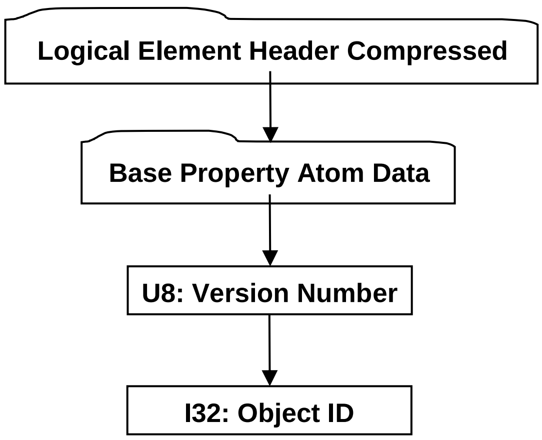


Figure 76 — JT Object Reference Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: Object ID

Object ID specifies the identifier within the JT file for the referenced object.

7.3.6 Date Property Atom Element

Object Type ID: 0xce357246, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1

Date Property Atom Element represents a property atom whose value is a “date”.

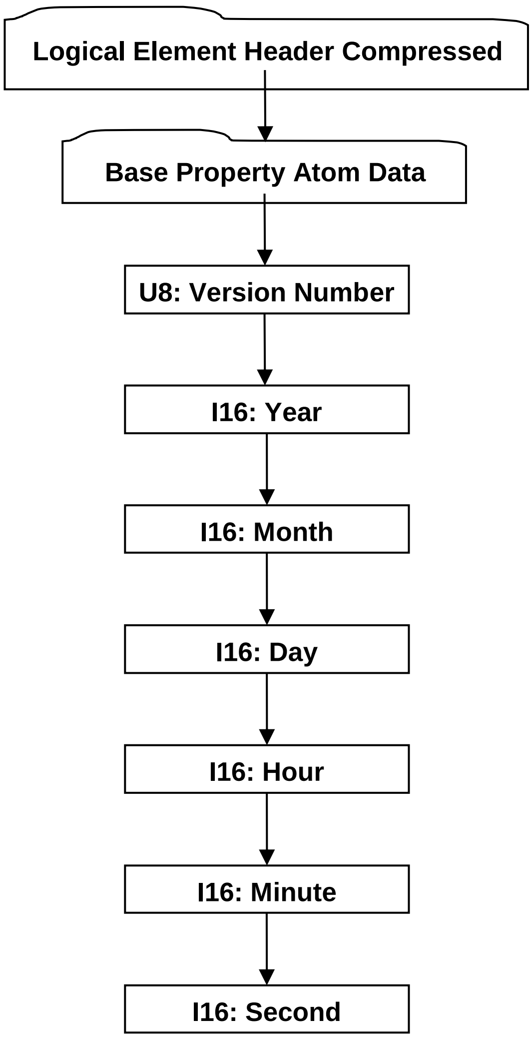


Figure 77 — Date Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I16: Year

Year specifies the date year value. Valid values are [1900, 2999] inclusive.

I16: Month

Month specifies the date month value. Valid values are [0, 11] inclusive.

I16: Day

Day specifies the date day value. Valid values are [1, 31] inclusive.

I16: Hour

Hour specifies the date hour value. Valid values are [0, 23] inclusive.

I16: Minute

Minute specifies the date minute value. Valid values are [0, 59] inclusive.

I16: Second

Second specifies the date Second value. Valid values are [0, 59] inclusive.

7.3.7 Late Loaded Property Atom Element

Object Type ID: 0xe0b05be5, 0xfbbd, 0x11d1, 0xa3, 0xa7, 0x00, 0xaa, 0x00, 0xd1, 0x09, 0x54

Late Loaded Property Atom Element is a property atom type used to reference an associated piece of atomic data in a separate addressable segment of the JT file. The “Late Loaded” connotation derives from the associated data being stored in a separate addressable segment of the JT file, and thus a JT file reader can be structured to support the “best practice” of delaying the loading/reading of the associated data until it is actually needed.

Late Loaded Property Atom Elements are used to store a variety of data, including, but not limited to, Shape LOD Segments and B-Rep Segments (see Shape LOD Element).

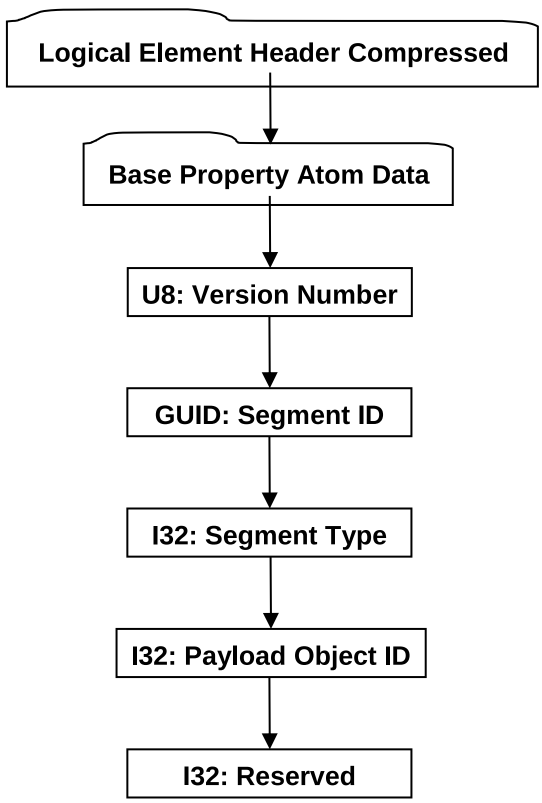


Figure 78 — Late Loaded Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

GUID: Segment ID

Segment ID is the globally unique identifier for the associated data segment in the JT file. See TOC Segment for additional information on how this Segment ID can be used in conjunction with the file TOC Entries to locate the associated data in the JT file.

The complete list of segment types can be found the Segment Types Table.

I32: Segment Type

Segment Type defines a broad classification of the associated data segment contents. For example, a Segment Type of “1” denotes that the segment contains Logical Scene Graph material; “2” denotes contents of a B-Rep, etc.

I32: Payload Object ID

Object ID is the identifier for the payload. Other objects referencing this particular payload will do so using the Object ID.

I32: Reserved

Reserved data field that is guaranteed to always be greater than or equal to 1.

7.3.8 Vector4f Property Atom Element

Object Type ID: 0x2e7db4be, 0xc71a, 0x4b18, 0x9d, 0x7, 0xc7, 0x22, 0x7e, 0x9f, 0xef, 0x76

Vector4f Property Atom Element represents a property atom whose value is of VecF32 data type with the length to be equal to 4 .

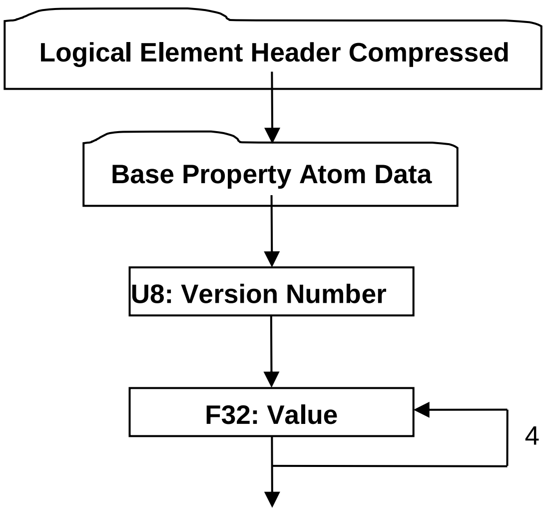


Figure 79 — Vector4f Property Atom Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Complete description for Base Property Atom Data can be found in Base Property Atom Data.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

F32: Value

Value contains the floating point value for this property atom.

7.4 Property Table

The Property Table is where the data connecting Node Elements and Attribute Elements with their associated Properties is stored. The Property Table contains an Element Property Table for each element in the JT File which has associated Properties. An Element Property Table is a list of key/value Property Atom Element pairs for all Properties associated with a particular Node Element Object or Attribute Element Object.

For a reference compliant JT File all Node Elements, Attribute Elements, and Property Atom Elements contained in a JT file should have been read by the time a JT file reader reaches the Property Table section of the file. This means that all Node Objects, Attribute Objects, and Property Atom Objects referenced in the Property Table (through Object IDs), should have already been read, and if not, then the file is corrupt (i.e. not reference compliant).

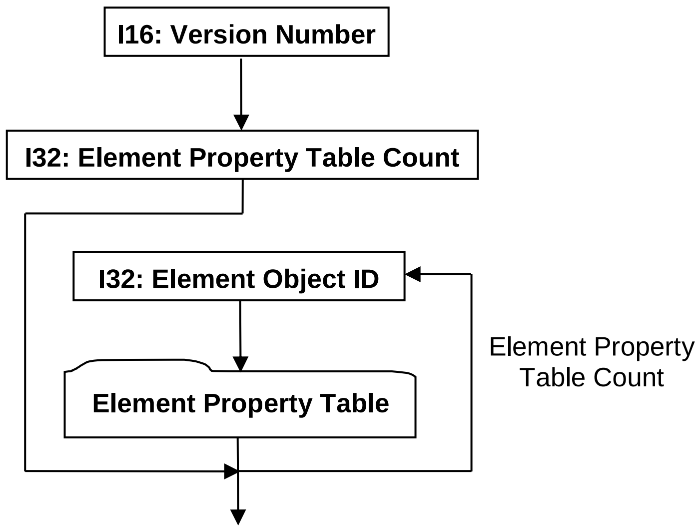


Figure 80 — Property Table data collection

I16: Version Number

Version Number is the version identifier for this Property Table. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: Element Property Table Count

Element Property Table Count specifies the number of Element Property Tables to follow. This value is equivalent to the total number of Node Elements (see Node Elements) and Attribute Elements (see Attribute Elements) that have associated Property Atom Elements (see Property Atom Elements).

I32: Element Object ID

Element Object ID is the identifier for the Node Element object (see Node Elements) or the Attribute Element object (see Attribute Elements) that the following Element Property Table is for (i.e. Node Element or Attribute Element that all properties in the following Element Property Table are associated with).

Element Property Table

The Element Property Table is a list of key/value Property Atom Element pairs for all properties associated with a particular Node Element Object or Attribute Element Object. The list is terminated by a "0" value for Key Property Atom Object ID.

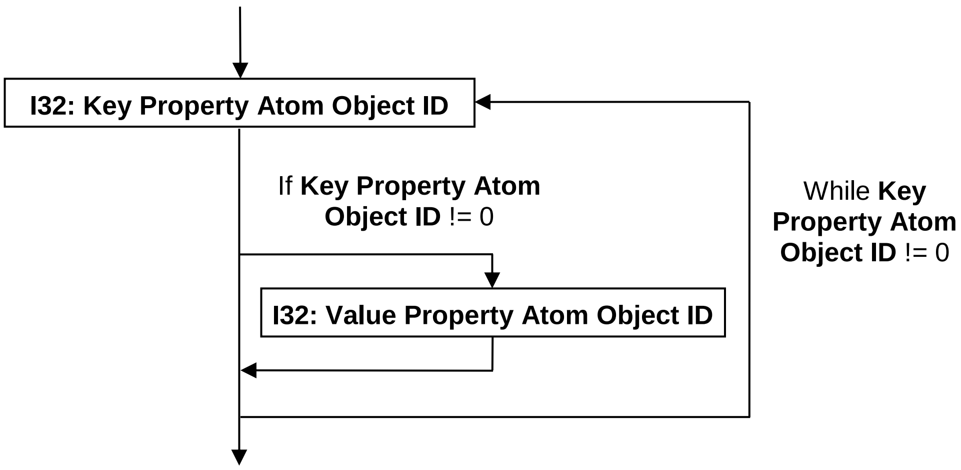


Figure 81 — Element Property Table data collection

I32: Key Property Atom Object ID

Key Property Atom Object ID is the identifier for the Property Atom Element object (see Property Atom Elements) representing the "key" part of the property key/value pair. A value of "0" indicates the end of the Node Property Table.

I32: Value Property Atom Object ID

Value Property Atom Object ID is the identifier for the Property Atom Element object (see Property Atom Elements) representing the "value" part of the property key/value pair. A value is not stored if I32: Key Property Atom Object ID has a value of "0".

8 Shape LOD Segment

Shape LOD Segment contains an Element that defines the geometric shape definition data (e.g. vertices, polygons, normals, etc) for a particular shape Level Of Detail or alternative representation. Shape LOD Segments are typically referenced by Shape Node Elements using Late Loaded Property Atom Elements (see Shape Node Elements and Late Loaded Property Atom Element respectively).

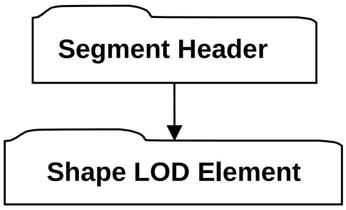


Figure 82 — Shape LOD Segment data collection

Complete description for Segment Header can be found in Segment Header.

8.1 Shape LOD Element

- A Shape LOD Element is the holder/container of the geometric shape definition data (e.g. vertices, polygons, normals, etc.) for a single LOD. Much of the “heavyweight” data contained within a Shape LOD Element may be optionally compressed and/or encoded. The compression and/or encoding state is indicated through other data stored in each Shape LOD Element.

There are several types of Shape LOD Elements which the JT format supports. The following sub-sections document the various Shape LOD Element types.

8.1.1 Tri-Strip Set Shape LOD Element

Object Type ID: 0x10dd10ab, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Tri-Strip Set Shape LOD Element contains the geometric shape definition data (e.g. vertices, polygons, normals, etc.) for a single LOD of a collection of independent and unconnected triangle strips. Each strip constitutes one primitive of the set and the ordering of the vertices in forming triangles, is the same as OpenGL’s triangle strip definition [1].

A Tri-Strip Set Shape LOD Element is typically referenced by a Tri-Strip Set Shape Node Element using Late Loaded Property Atom Elements (see Tri-Strip Set Shape Node Element and Late Loaded Property Atom Element respectively).

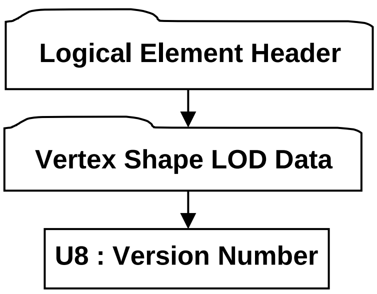


Figure 83 — Tri-Strip Set Shape LOD Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

Complete description for Vertex Shape LOD Data can be found in Vertex Shape LOD Data.

U8 : Version Number

Version Number is the version identifier for this Tri-Strip Set Shape LOD. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

8.1.2 Polyline Set Shape LOD Element

Object Type ID: 0x10dd10a1, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A 7.1.2 Polyline Set Shape LOD Element contains the geometric shape definition data (e.g. vertices, normals, etc.) for a single LOD of a collection of independent and unconnected polylines. Each polyline constitutes one primitive of the set.

A Polyline Set Shape LOD Element is typically referenced by a Polyline Set Shape Node Element using Late Loaded Property Atom Elements (see Polyline Set Shape Node Element and Late Loaded Property Atom Element respectively).

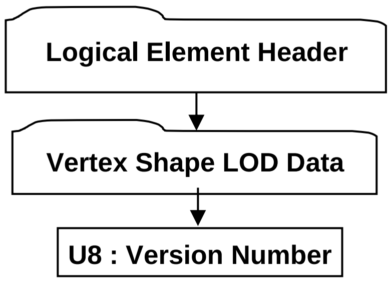


Figure 84 — Polyline Set Shape LOD Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

Complete description for Vertex Shape LOD Data can be found in Vertex Shape LOD Data.

U8 : Version Number

Version Number is the version identifier for this Polyline Set Shape LOD. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

8.1.3 Point Set Shape LOD Element

Object Type ID: 0x98134716, 0x0011, 0x0818, 0x19, 0x98, 0x08, 0x00, 0x09, 0x83, 0x5d, 0x5a

A Point Set Shape LOD Element contains the geometric shape definition data (e.g. coordinates, normals, etc.) for a collection of independent and unconnected points. Each point constitutes one primitive of the set.

A Point Set Shape LOD Element is typically referenced by a Point Set Shape Node Element using Late Loaded Property Atom Elements (see Point Set Shape Node Element and Late Loaded Property Atom Element respectively).

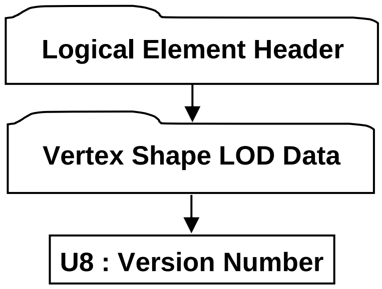


Figure 85 — Point Set Shape LOD Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

Complete description for Vertex Shape LOD Data can be found in Vertex Shape LOD Data.

U8 : Version Number

Version Number is the version identifier for this Point Set Shape LOD. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

8.1.4 Polygon Set LOD Element

Object Type ID: 0x10dd109f, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Polygon Set LOD Element contains the geometric shape definition data (e.g. vertices, polygons, normals, etc.) for a single LOD of a collection of independent and unconnected polygons. Each polygon constitutes one primitive of the set and the ordering of the vertices in forming polygons, is the same as OpenGL's polygon definition [1].

A Polygon Set LOD Element is typically referenced by a Polygon Set Shape Node Element using Late Loaded Property Atom Elements (see Polygon Set Shape Node Element and Late Loaded Property Atom Element respectively).

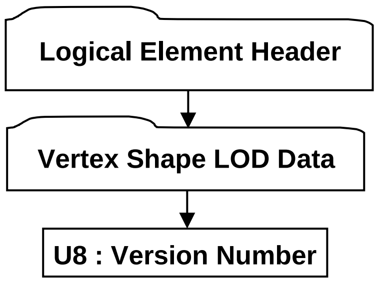


Figure 86 — Polygon Set LOD Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

Complete description for Vertex Shape LOD Data can be found in Vertex Shape LOD Data.

U8 : Version Number

Version Number is the version identifier for this Polygon Set LOD Element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

Vertex Shape LOD Data

Vertex Shape LOD Data collection is an abstract container for geometric *primitives* such as triangles, line strips, or points, depending on the specific type of Vertex Shape. The set of primitives are further partitioned into so-called "face groups." The Vertex Shape LOD Data also contains the vertex attribute bindings and quantization settings used to store the vertex records referenced by the primitives.

One use for face groups is to establish a correspondence between Brep faces and their triangle representation. A convention for mapping JTBrep and XTBrep faces to face groups is described in section B-Rep Face Group Associations.

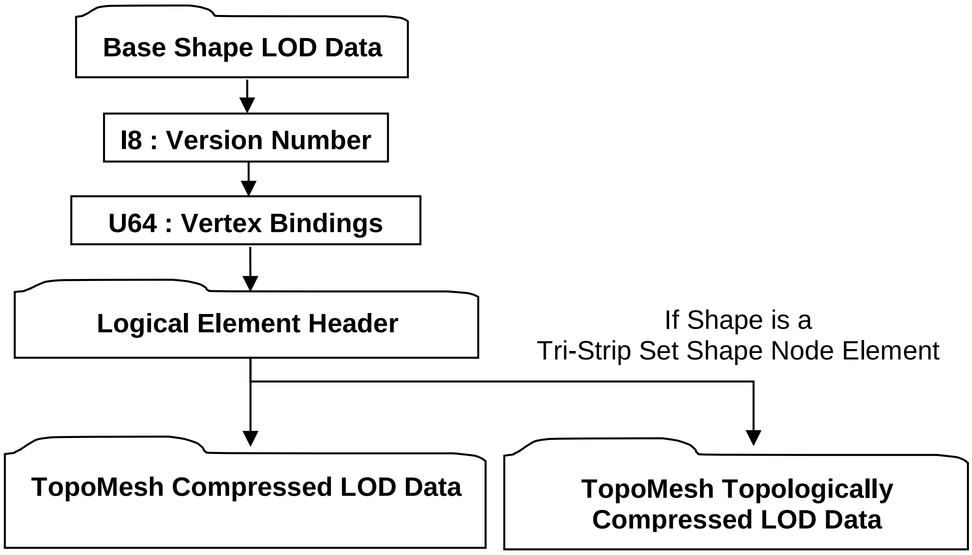


Figure 87 — Vertex Shape LOD Data collection

Complete description for Logical Element Header can be found in Logical Element Header.

Complete description for TopoMesh Compressed LOD Data and TopoMesh Topologically Compressed LOD Data can be found in TopoMesh Compressed LOD Data and TopoMesh Topologically Compressed LOD Data.

I8 : Version Number

Version Number is the version identifier for this Vertex Shape LOD Data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U64 : Vertex Bindings

Binding Attributes is a collection of normal, texture coordinate, and colour binding information encoded within a single U64 using the following bit allocation. All bits fields that are not defined as in use should be set to “0”.

Table 48 — Vertex Shape LOD Bindings values

Bits 1-3	Vertex Coordinate Binding. The Vertex Coordinate Binding denotes per vertex coordinate field data is present when one of the bits is set. Bit 1 - 2 Component Vertex Coordinates Bit 2 - 3 Component Vertex Coordinates Bit 3 - 4 Component Vertex Coordinates
Bit 4	Normal Binding. The Normal Binding denotes per vertex normal field data is present when the bit is set. Normal field data is always stored in 3 Component Normals when present.
Bits 5 -6	Colour Binding. The Colour Binding denotes per vertex colour field data

	is present when one of the bits is set. Bit 5 - 3 Component Colours Bit 6 - 4 Component Colour
Bit 7	Vertex Flag Binding. The Vertex Flag Binding denotes the per vertex flag field is present on the shape when the bit is set.
Bits 9-12	Texture Coordinate 0 Binding. The Texture Coordinate 0 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 9 - 1 Component Texture Coordinates Bit 10 - 2 Component Texture Coordinates Bit 11 - 3 Component Texture Coordinates Bit 12 - 4 Component Texture Coordinates
Bits 13-16	Texture Coordinate 1 Binding. The Texture Coordinate 1 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 13 - 1 Component Texture Coordinates Bit 14 - 2 Component Texture Coordinates Bit 15 - 3 Component Texture Coordinates Bit 16 - 4 Component Texture Coordinates
Bits 17-20	Texture Coordinate 2 Binding. The Texture Coordinate 2 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 17 - 1 Component Texture Coordinates Bit 18 - 2 Component Texture Coordinates Bit 19 - 3 Component Texture Coordinates Bit 20 - 4 Component Texture Coordinates
Bits 21-24	Texture Coordinate 3 Binding. The Texture Coordinate 3 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 21 - 1 Component Texture Coordinates Bit 22 - 2 Component Texture Coordinates Bit 23 - 3 Component Texture Coordinates Bit 24 - 4 Component Texture Coordinates
Bits 25-28	Texture Coordinate 4 Binding. The Texture Coordinate 4 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 25 - 1 Component Texture Coordinates Bit 26 - 2 Component Texture Coordinates Bit 27 - 3 Component Texture Coordinates Bit 28 - 4 Component Texture Coordinates
Bits 29-32	Texture Coordinate 5 Binding. The Texture Coordinate 5 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 29 - 1 Component Texture Coordinates Bit 30 - 2 Component Texture Coordinates Bit 31 - 3 Component Texture Coordinates Bit 32 - 4 Component Texture Coordinates
Bits 33-36	Texture Coordinate 6 Binding. The Texture Coordinate 6 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 33 - 1 Component Texture Coordinates Bit 34 - 2 Component Texture Coordinates Bit 35 - 3 Component Texture Coordinates Bit 36 - 4 Component Texture Coordinates
Bits 37-40	Texture Coordinate 7 Binding. The Texture Coordinate 7 binding denotes per vertex texture coordinates field data is present when one of the bits is set: Bit 37 - 1 Component Texture Coordinates Bit 38 - 2 Component Texture Coordinates

	Bit 39 - 3 Component Texture Coordinates Bit 40 - 4 Component Texture Coordinates
Bits 41-62	Unused
Bit 63	
Bit 64	Auxiliary Vertex Field Binding. The Auxiliary Vertex Field Binding denotes per vertex auxiliary field data is present on the shape when the bit is set.

Base Shape LOD Data

Base shape LOD data contains the common items to all shape LODs.

I8 : Version Number

Figure 88 — Base Shape LOD Data collection

I8 : Version Number

Version Number is the version identifier for this Base Shape LOD Data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

TopoMesh Compressed LOD Data

TopoMesh Compressed LOD Data collection contains the common items to all TopoMesh Compressed LOD data elements.

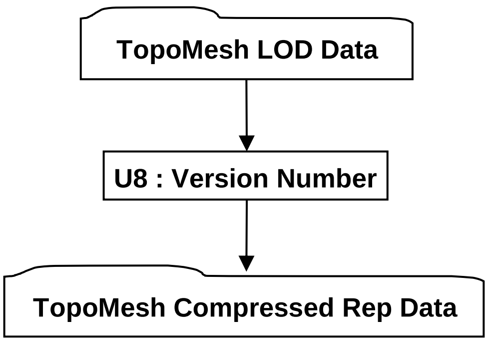


Figure 89 — TopoMesh Compressed LOD Data collection

Complete description for TopoMesh LOD Data, and TopoMesh Compressed Rep Data, can be found in TopoMesh LOD Data, TopoMesh Compressed Rep Data.

U8 : Version Number

Version Number is the version identifier for this TopoMesh Compressed LOD Data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

TopoMesh LOD Data

TopoMesh LOD Data collection contains the common items to all TopoMesh LOD elements.

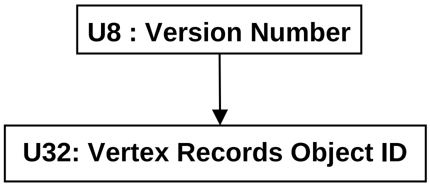


Figure 90 — TopoMesh LOD Data collection

U8 : Version Number

Version Number is the version identifier for this TopoMesh LOD Data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U32: Vertex Records Object ID

Vertex Records Object ID is the identifier for the vertex records associated with this Object. Other objects referencing these vertex records will do so using this Object ID. It is via this mechanism that multiple TopMeshes are able to reference the same set of vertex records.

TopoMesh Compressed Rep Data

TopoMesh Compressed Rep Data contains the geometric shape definition data (e.g. vertices, colours, normals, etc.) in a lossy or lossless compressed format. This format is used when the shape type is Polyline Set Shape Node Element, or Point Set Shape Node Element. For Tri-Strip Set Shape Node Element and Polygon Set Shape Node Element, please refer to Topologically Compressed Rep Data.

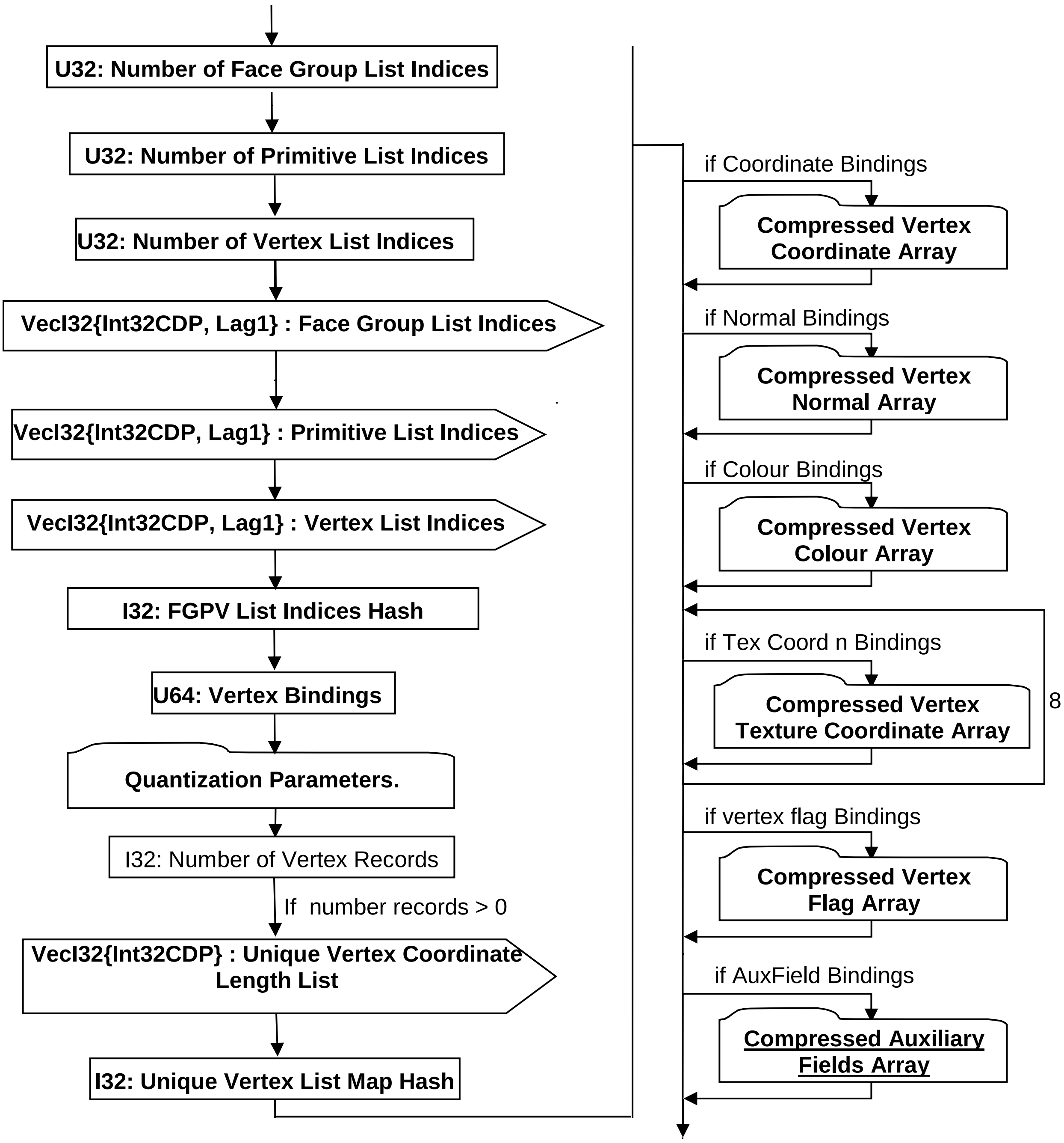


Figure 91 — TopoMesh Compressed Rep Data data collection

Complete description for Quantization Parameters can be found in Quantization Parameters.

Complete description for Compressed Vertex Coordinate Array, Compressed Vertex Normal Array, Compressed Vertex Colour Array, Compressed Vertex Texture Coordinate Array, Compressed Vertex Flag Array and Compressed Vertex Auxiliary Fields Array can be found in Data Compression and Encoding.

U32: Number of Face Group List Indices

Number of Face Group List Indices.

U32: Number of Primitive List Indices

Number of Primitive List Indices.

U32: Number of Vertex List Indices

Number of Vertex List Indices.

VecI32{Int32CDP, Lag1} : Face Group List Indices

Face Group List Indices is a vector of indices into the uncompressed Raw Primitive Data marking the start/beginning of Faces. Face Group List Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1} : Primitive List Indices

Primitive List Indices is a vector of indices into the uncompressed Raw Vertex Data marking the start/beginning of primitives. Primitive List Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1} : Vertex List Indices

Vertex List Indices is a vector of indices (one per vertex) into the uncompressed/dequantized unique vertex data arrays (Vertex Coords, Vertex Normals, Vertex Texture Coords, Vertex Colours) identifying each Vertex's data (i.e. for each Vertex there is an index identifying the location within the unique arrays of the particular Vertex's data). The Compressed Vertex Index List uses the Int32 version of the CODEC to compress and encode data.

I32: FGPV List Indices Hash

The FGPV Hash is the combined hash value of the Face Group List Indices (if Polyline), Primitive List Indices, and Vertex List Indices. Refer to section Annex C for a more detailed description on hashing.

```
UInt32 uHash    = 0;
UInt32 nFGIdx   = 0,
       nPrimIdx = 0,
       nVtxIdx  = 0;
vecI32 vFGIndices, vPrimIdices, vVertexIndices;
...
uHash = hash32( (UInt32*)(& vFGIndices), nFGIdx+1, uHash );
uHash = hash32( (UInt32*)(& vPrimIdices), nPrimIdx+1, uHash );
uHash = hash32( (UInt32*)(& vVertexIndices), nVtxIdx  , uHash );
```

U64: Vertex Bindings

Vertex Bindings is a collection of normal, texture coordinate, and colour binding information encoded within a single U64. All bits fields that are not defined as in use should be set to "0". For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

I32: Number of Vertex Records

Number of vertex records.

VecI32{Int32CDP} : Unique Vertex Coordinate Length List

The Unique Vertex Length List contains the number of vertex records containing each of the unique vertex coordinates and should sum to the number of vertex records. When read in the Compressed

Vertex Coordinate Array only contains a single value for each unique vertex coordinate value and is therefore parallel to the Unique Vertex Length List. In order to expand its coordinates into the vertex record space it unique coordinate value will need to be smeared out such that each unique vertex coordinate is repeated the number of times specified in the Unique Vertex Length List. The Compressed Vertex Normal, Colour, Texture, and Flag Arrays do not require the same expansion.

I32: Unique Vertex List Map Hash

The Unique Vertex List Map Hash is the hash value of Unique Vertex Coordinate Length List. Refer to section Annex C for a more detailed description on hashing.

```
UInt32 uHash    = 0;
UInt32 nUniqVtx = 0;
vecF32 vUniqVtxIndices;
...
uHash = hash32( (UInt32*)(&vUniqVtxIndices), nUniqVtx, uHash );
```

Quantization Parameters

Quantization Parameters specifies for each shape data type grouping (i.e. Vertex, Normal, Texture Coordinates, Colour) the number of quantization bits used for given qualitative compression level. Although these Quantization Parameters values are saved in the associated/referenced Shape LOD Element, they are also saved here so that a JT File loader/reader does not have to load the Shape LOD Element in order to determine the Shape quantization level. See Shape LOD Element for complete description of Shape LOD Elements.

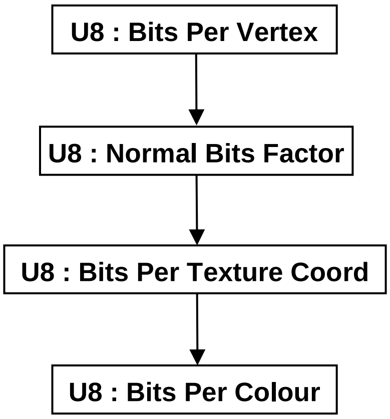


Figure 92 — Quantization Parameters data collection

U8 : Bits Per Vertex

Bits Per Vertex specifies the number of quantization bits per vertex coordinate component. Value shall be within range [0:24] inclusive.

U8 : Normal Bits Factor

Normal Bits Factor is a parameter used to calculate the number of quantization bits for normal vectors. Value shall be within range [0:13] inclusive. The actual number of quantization bits per normal is computed using this factor and the following formula: "BitsPerNormal = 6 + 2 * Normal Bits Factor".

U8 : Bits Per Texture Coord

Bits Per Texture Coord specifies the number of quantization bits per texture coordinate component. Value shall be within range [0:24] inclusive.

U8 : Bits Per Colour

Bits Per Colour specifies the number of quantization bits per colour component. Value shall be within range [0:24] inclusive.

TopoMesh Topologically Compressed LOD Data

TopoMesh Topologically Compressed LOD Data collection contains the common items to all TopoMesh Topologically Compressed LOD data elements such as Tri-Strip Set Shape LOD Element.

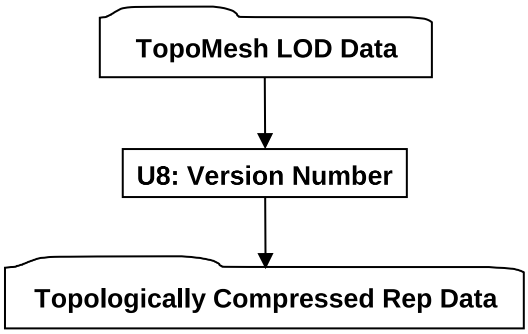


Figure 93 — TopoMesh Topologically Compressed LOD Data collection

Complete description for TopoMesh LOD Data and Topologically Compressed Rep Data can be found in TopoMesh LOD Data and Topologically Compressed Rep Data.

U8: Version Number

Version Number is the version identifier for this TopoMesh Topologically Compressed LOD Data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

8.1.4.1 Topologically Compressed Rep Data

This JT specification, like JT v9, represents triangle strip data very differently than it does in the JT v8 format. The scheme stores the triangles from a TriStripSet or polygons from a PolygonSet as a topologically-connected mesh. Even though *more* information is stored to the JT file, the additional structure provided by storing the full topological adjacency information actually provides a handsome reduction in the number of bytes needed to encode the triangles or polygons. More importantly, however, the topological information aids us in a more significant respect -- that of only storing the *unique* vertex records used by the TriStripSet or PolygonSet. Combined, these two effects reduce the typical storage footprint of TriStripSet data by approximately half relative to the JT v8 format.

The tristrip information itself is not stored in the JT file -- only the triangles themselves. The reader is expected to re-tristrip (or not) as she sees fit, as tristrips may no longer provide a performance advantage during rendering. There may, however, remain some memory savings for tristripping, and so the decision to tristrip is left to the user.

To begin the decoding process, first read the compressed data fields shown in figure below. These fields provide all the information necessary to reconstruct the per face-group organized sets of triangles. The first 22 fields represent the topological information, and the remaining fields constitute the set of unique vertex records to be used. The next step is to run the topological decoder algorithm detailed in provided in the annex on this data to reconstruct the topologically connected representation of the triangle mesh in a so-called "Dual VFMesh." The triangles or polygons in this heavy-weight data structure can then be exported to a lighter-weight form, and the dual VFMesh discarded if desired.

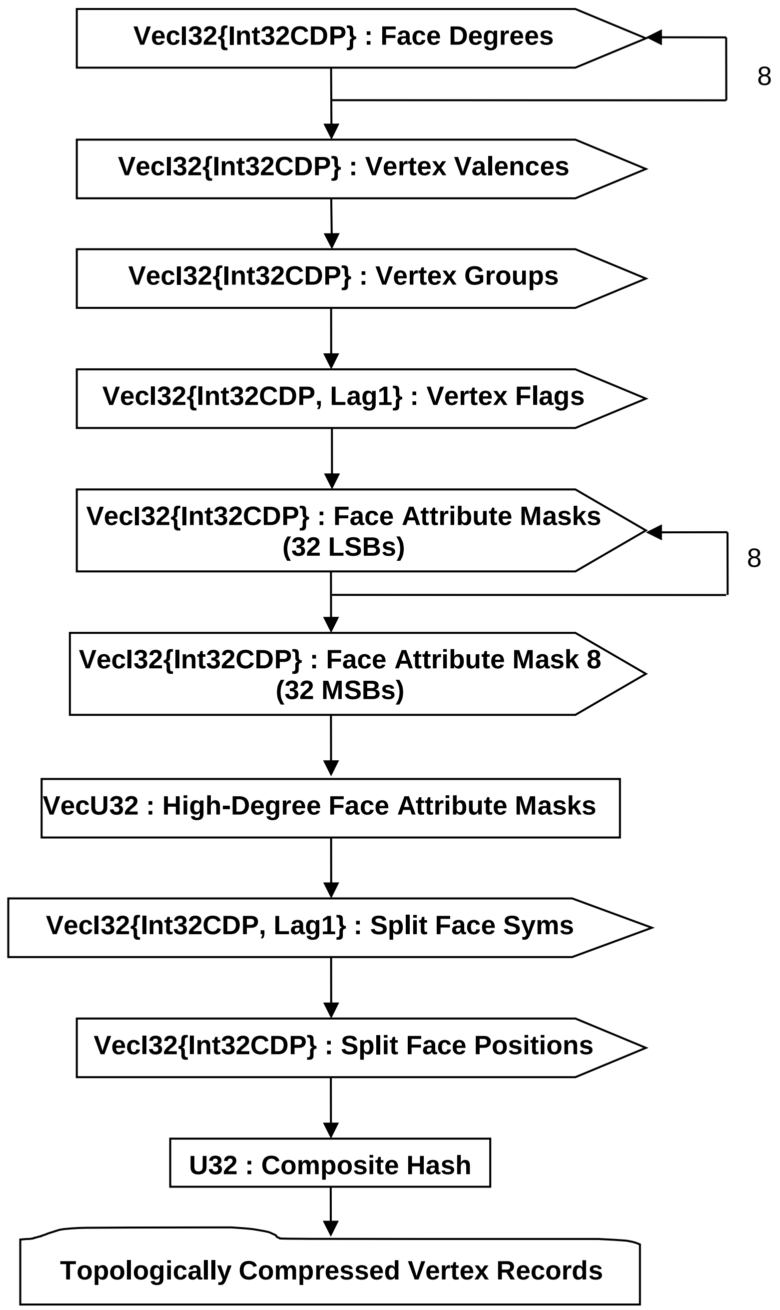


Figure 94 — Topologically Compressed Rep Data Collection

VecI32{Int32CDP} : Face Degrees

Similarly to the way valences are encoded, the topology encoder emits the *degree* (number of incident vertices) of each face *in the order they were visited*. The number of face degrees in this array is equal to the number of faces in the mesh.

VecI32{Int32CDP} : Vertex Valences

As the coder visits each vertex in the mesh, it emits the *valence* (number of incident faces) of each vertex. These valences are collect *in the order they were visited* into this array. The number of valences in this array is equal to the number of (topological) vertices in the mesh.

VecI32{Int32CDP} : Vertex Groups

This array is parallel to the Vertex Valences array above. As the coder emits the valence of each vertex, it also emits the face group number to which the dual vertex belongs into this array.

VecI32{Int32CDP, Lag1} : Vertex Flags

This array is also parallel to the Vertex Valences array, and contains a value of 0 when the dual face was present in the original triangle mesh, and a value of 1 if the dual face is a *cover face* that was added to artificially close the original mesh.

VecI32{Int32CDP} : Face Attribute Masks (32 LSBs)

This field is written 8 times – once for each of the 8 context groups listed above – and encodes the face attribute bit vector associated with a single face.

VecI32{Int32CDP} : Face Attribute Mask 8 (32 MSBs)

This field encodes the 32 most significant bits of the 8^{th} context group of face attribute bit vectors.

VecU32 : High-Degree Face Attribute Masks

This field encodes all remaining face attribute bit vectors, adjoined end-to-end, and encoded as a single array of unsigned integers.

VecI32{Int32CDP, Lag1} : Split Face Syms

Encodes the list of “split face” ID numbers in the order the coder encountered them.

VecI32{Int32CDP} : Split Face Positions

Encodes the list of “split face” positions in the active vertex queue in the order the code encountered them.

U32 : Composite Hash

This field is a hash value computed on all of the above data using the hash function described in Annex C. It is written into the JT file so that a reader can perform the same hash on the above data and compare against this value in order to guarantee that it has read and decoded correct data from the JT file. It is *highly* encouraged that all readers perform this check, as even a single bit error in the topology information above can have catastrophic consequences on the topology decoder and the resulting mesh. Any writers are *required* to write this field using the method provided so that other readers may validate the data they read.

```
UInt32 uHash        = 0;
UInt32 anDegSyms[8] = {0},
       nValSyms = 0,
       nVGrpSyms = 0,
       nVtxFlags = 0,
       anAttrMasks[8] = {0},
       nLrgAttrMasks = 0,
       nSplitVtxSyms = 0,
       nSplitVtxPos = 0;
VecI32 vFaceDegreeSymbols[8], vviValenceSymbols, vFaceGroupSyms,
       vvuAttrMasks[8], viSplitVtxSyms, viSplitVtxPos;
VecI16 vFaceFlags;
VecU32 vuTmp, vuAttrMasksLrg;
...
for (i=0 ; i<8 ;i++)
  uHash = hash32((UInt32*) vFaceDegreeSymbols[i].ptr(), anDegSyms[i], uHash );
uHash = hash32((UInt32*) vviValenceSymbols.ptr(), nValSyms, uHash );
```

```
uHash = hash32((UInt32*)vVtxGroupSyms.ptr(), nVGrpSyms, uHash );
uHash = hash16((UInt16*)vVtxFlags.ptr(), nFlags, uHash );
for (i=0 ; i<7 ;i++)
  uHash = hash32((UInt32*)vvuAttrMasks[i].ptr(), anAttrMasks[i], uHash );
vuTmp = vvuAttrMasks[7] & 0xffffffff;  // Lower 32 bits of each element
uHash = hash32(vuTmp.ptr(), anAttrMasks[7], uHash );
vuTmp = (vvuAttrMasks[7] >> 32) & 0xffffffff; // Next 32 bits of each element
uHash = hash32(vuTmp.ptr(), anAttrMasks[7], uHash );
uHash = hash32(vuAttrMasksLrg.ptr(), nLrgAttrMasks, uHash );
uHash = hash32((UInt32*)viSplitVtxSyms.ptr(), nSplitVtxSyms, uHash );
uHash = hash32((UInt32*)viSplitVtxPos.ptr(), nSplitVtxPos, uHash );
```

8.1.4.2 Topologically Compressed Vertex Records

Documented here is the format of the vertex data written by the topological encoder found in the annex. Some additional explanation is necessary, however, because only the *unique* vertex coordinates are written to the JT file, while the remaining vertex attributes (normals, colours, texture coordinates, vertex flags) may not be unique.

Vertex coordinates are written to the file in the order that they were visited by the topology encoder. Note that this means that the number of vertex coordinates written is equal to the number of topological vertices in the mesh (i.e. all vertex coordinates are unique).

By contrast one set of vertex attribute records is written to the file corresponding to each 1 bit across all encoded dual Face Attribute Masks. The vertex attribute records are written in the order that the topology encoder visited them. The reader shall then use the topology decoder's output to correctly associate each vertex attribute record to the correct vertex coordinate using the dual Face Attribute Masks.

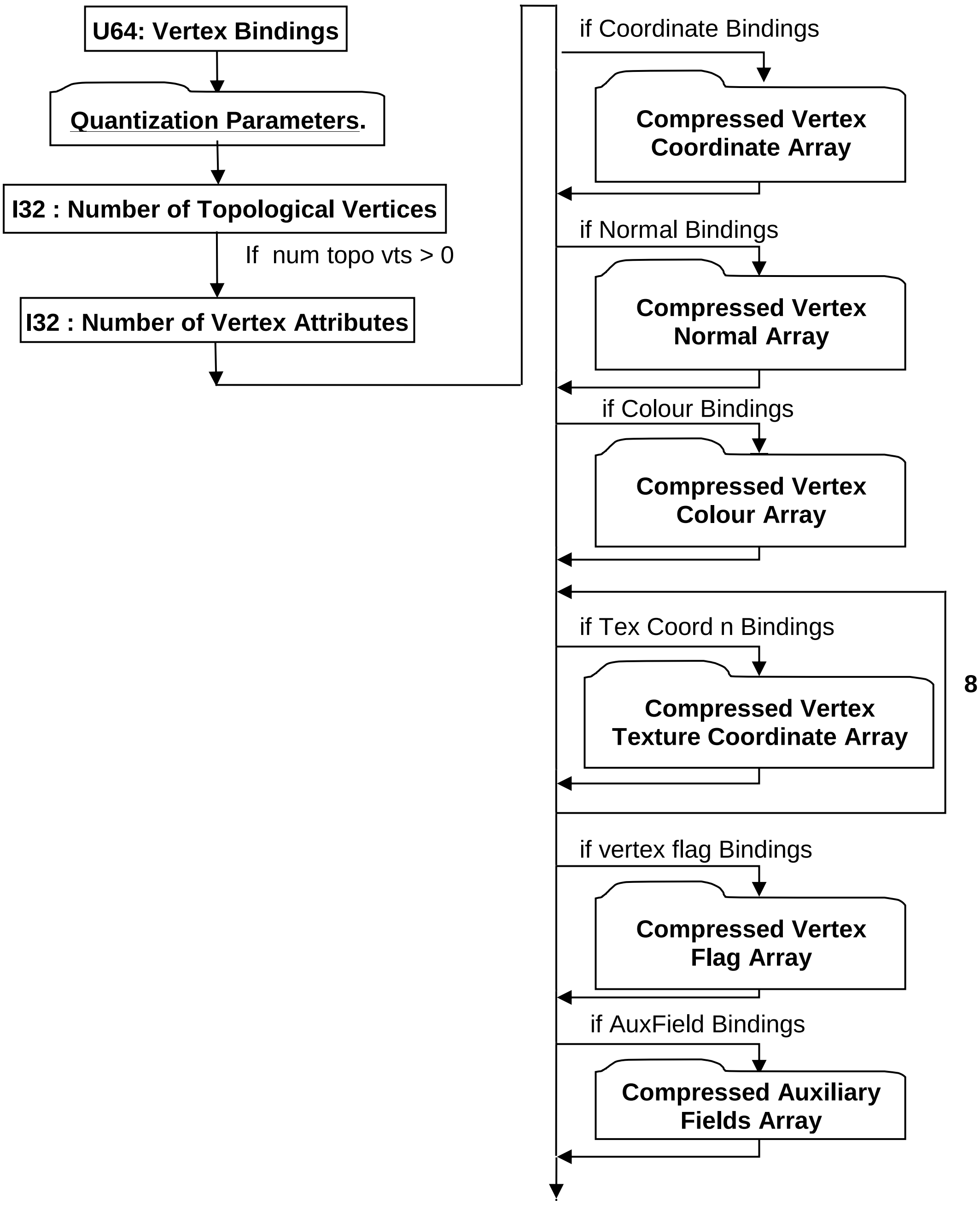


Figure 95 — Topologically Compressed Vertex Records data collection

Complete description for Compressed Vertex Coordinate Array, Compressed Vertex Normal Array, Compressed Vertex Colour Array, Compressed Vertex Texture Coordinate Array, Compressed Vertex Flag Array and Compressed Vertex Auxiliary Fields Array can be found in Data Compression and Encoding.

U64: Vertex Bindings

Vertex Bindings is a collection of normal, texture coordinate, and colour binding information encoded within a single U64. All bits fields that are not defined as in use should be set to “0”. For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

I32 : Number of Topological Vertices

This field is the number of topological vertices encoded by the topology encoder. This is the number of unique vertex coordinates that will be written in the later Compressed Vertex Coordinate Array field.

I32 : Number of Vertex Attributes

One set of vertex attribute records is written to the file corresponding to each 1 bit across all encoded dual Face Attribute Masks. The vertex attribute records are written in the order that the topology encoder visited them. The reader shall then use the topology decoder's output to correctly associate each vertex attribute record to the correct vertex coordinate using the dual Face Attribute Masks.

8.1.5 Null Shape LOD Element

Object Type ID: 0x3e637aed, 0x2a89, 0x41f8, 0xa9, 0xfd, 0x55, 0x37, 0x37, 0x3, 0x96, 0x82

A Null Shape LOD Element represents the pseudo geometric shape definition data for a NULL Shape Node Element. Although a NULL Shape Node Element has no real geometric primitive representation (i.e. is empty), its usage as a “proxy/placeholder” node within the LSG still supports the concept of having a defined bounding box and thus the existence of this Null Shape LOD Element type.

A Null Shape LOD Element is typically referenced by a NULL Shape Node Element using Late Loaded Property Atom Elements (see NULL Shape Node Element and Late Loaded Property Atom Element respectively).

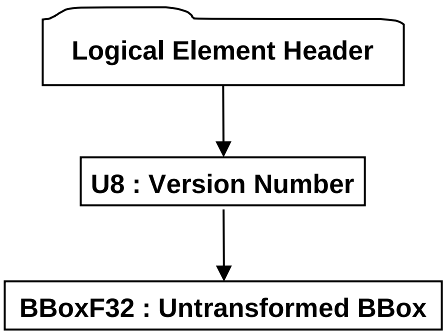


Figure 96 — Null Shape LOD Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

U8 : Version Number

Version Number is the version identifier for this Null Shape LOD Element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

BBoxF32 : Untransformed BBox

The Untransformed BBox is an axis-aligned LCS bounding box and represents the untransformed extents for this Null Shape LOD Element.

8.1.6 Primitive Set Shape Element

Object Type ID: 0xe40373c2, 0x1ad9, 0x11d3, 0x9d, 0xaf, 0x0, 0xa0, 0xc9, 0xc7, 0xdd, 0xc2

A Primitive Set Shape Element defines the minimum data necessary to procedurally generate LODs for a list of primitive shapes (e.g. box, cylinder, sphere, etc.). “Procedurally generate” means that the raw geometric shape definition data (e.g. vertices, polygons, normals, etc) for LODs is not directly stored; instead some basic shape information is stored (e.g. sphere centre and radius) from which LODs can be generated.

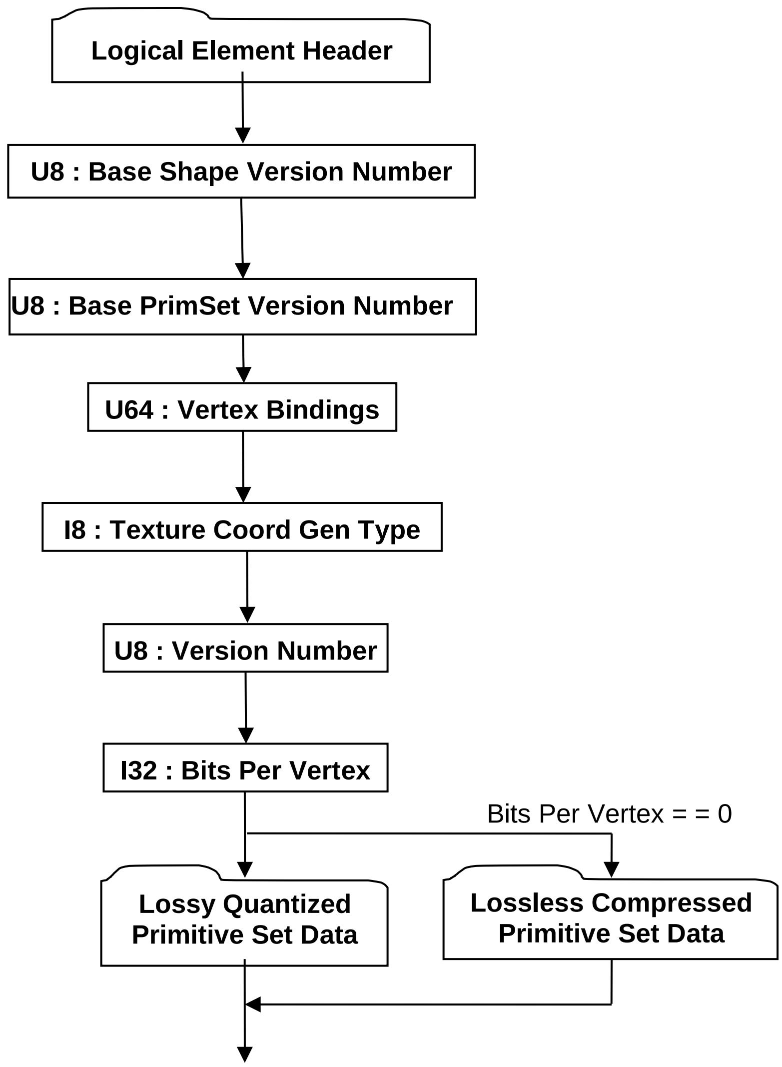


Figure 97 — Primitive Set Shape Element data collection

Complete description for Logical Element Header can be found in Logical Element Header.

U8 : Base Shape Version Number

Base Shape Version Number is the version identifier for the 2-level base class of this element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U8 : Base PrimSet Version Number

Base PrimSet Version Number is the version identifier for the immediate base class of element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

U64 : Vertex Bindings

Vertex Bindings is a collection of normal, texture coordinate, and colour binding information encoded within a single U64. All bits fields that are not defined as in use should be set to “0”. For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

U8 : Version Number

Version Number is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow.

Table 49 — Primitive Set Shape Version Number values

= 1	Version-1 Format

I32 : Bits Per Vertex

Bits Per Vertex specifies the number of quantization bits per vertex coordinate component. Value shall be within range [0:32] inclusive.

I8 : Texture Coord Gen Type

Texture Coord Gen Type specifies how a texture is applied to each face of the primitive. Single tile means one copy of the texture will be stretched to fit the face, isotropic means that the texture will be duplicated on the longer dimension of the face in order to maintain the texture’s aspect ratio.

Table 50 — Primitive Set Shape Texture Coord Gen Type values

= 0	Single Tile…Indicates that a single copy of a texture image will be applied to significant primitive features (i.e. cube face, cylinder wall, end cap) no matter how eccentrically shaped.
= 1	Isotropic…Implies that multiple copies of a texture image may be mapped onto eccentric surfaces such that a mapped texel stays approximately square.

Lossless Compressed Primitive Set Data

The Lossless Compressed Primitive Set Data collection contains all the per-primitive information stored in a “lossless” compression format for all primitives in the Primitive Set. The Lossless Compressed Primitive Set Data collection is only present when the Bits Per Vertex data field equals “0” (see Primitive Set Shape Element for complete description).

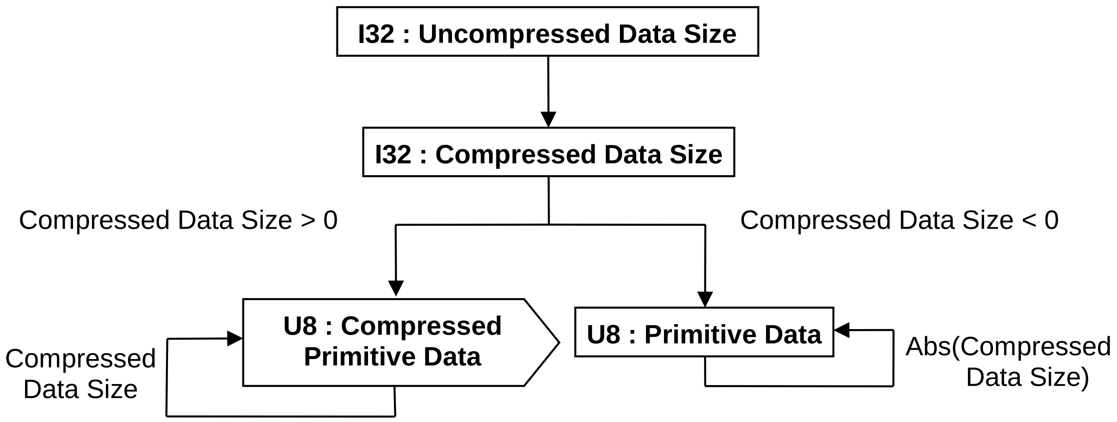


Figure 98 — Lossless Compressed Primitive Set Data **collection**

I32 : Uncompressed Data Size

Uncompressed Data size specifies the uncompressed size of Primitive Data or Compressed Primitive Data in bytes.

I32 : Compressed Data Size

Compressed Data Size specifies the compressed size of Primitive Data or Compressed Primitive Data in bytes. If the Compressed Data Size is negative, then the Compressed Primitive Data field is not present (i.e. data is not compressed) and the absolute value of Compressed Data Size should be equal to Uncompressed Data Size value.

U8 : Primitive Data

The Primitive Data field is a packed array of the raw per primitive data (i.e. reserved, params1, params2, params3, colour, type) sequentially for all primitives in the set. The Primitive Data field is only present if Compressed Data Size value is less than zero. The per primitive data is packed into Primitive Data array using an interleaved data schema/format as follows:

{[reserved], [params1], [params2], [params3], [colour], [type]}, …, **for all primitives**

Where the data elements have the following size and meaning:

Table 51 — Lossless Compressed Primitive Set Data Field values

Element	Data Type	Description
reserved	I32	This is a field reserved for future expansion of the JT Format.
params1	CoordF32	Interpretation is Primitive Type specific (see below table)
params2	DirF32	Interpretation is Primitive Type specific (see below table)
params3	Quaternion	Interpretation is Primitive Type specific (see below table)
Colour	RGB	Red, Green, Blue colour component values
Type	I32	Primitive Type = 0 – Box = 1 – Cylinder = 2 – Pyramid = 3 – Sphere = 4 – Tri-Prism

Given this format of the Primitive Data, and the previously read size fields, a reader can then implicitly compute the data stride (length of one primitive entry in Primitive Data), and number of primitives.

The interpretation of the three “params#” data fields is primitive type dependent as follows:

Table 52 — Primitive Set “params#” Data Fields Interpretation

Primitive Type	params1			params2			params3			
	[0]	[1]	[2]	[0]	[1]	[2]	[0]	[1]	[2]	[3]
Box	min X	min Y	min Z	length X	length Y	length Z	orientation in Quaternion form			
Cylinder	base centre X	base centre Y	base centre Z	spine X	spine Y	spine Z	radius 1	radius 2	N/A	N/A
Pyramid	base centre X	base centre Y	base centre Z	length X	length Y	length Z	orientation in Quaternion form			
Sphere	centre X	centre Y	centre Z	radius	N/A	N/A	N/A	N/A	N/A	N/A
Tri-Prism	bottom	bottom	bottom	length	length	length	orientation in Quaternion form			

Primitive Type	params1			params2			params3			
	[0]	[1]	[2]	[0]	[1]	[2]	[0]	[1]	[2]	[3]
	front X	front Y	front Z	X (to right)	Y (to back)	Z (to top)				

U8 : Compressed Primitive Data

The Compressed Primitive Data field represents the same data as documented in Primitive Data field above except that the data is compressed using the general "LZMA" method. The Compressed Primitive Data field is only present if Compressed Data Size value is greater than zero. See Data Compression and Encoding for more details on LZMA compression and LZMA library version used.

Lossy Quantized Primitive Set Data

The Lossy Quantized Primitive Set Data collection contains all the per-primitive information (i.e. reserved, params1, params2, params3, colour, type) stored in a "lossy" encoding/compression format for all primitives in the Primitive Set. The Lossy Quantized Primitive Set Data collection is only present when the Bits Per Vertex data field is NOT equal to "0" (See Primitive Set Shape Element for compete description).

The interpretation of the three per-primitive "params#" data fields is primitive type dependent. See Table 52 — Primitive Set "params#" Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the "params#" data fields.

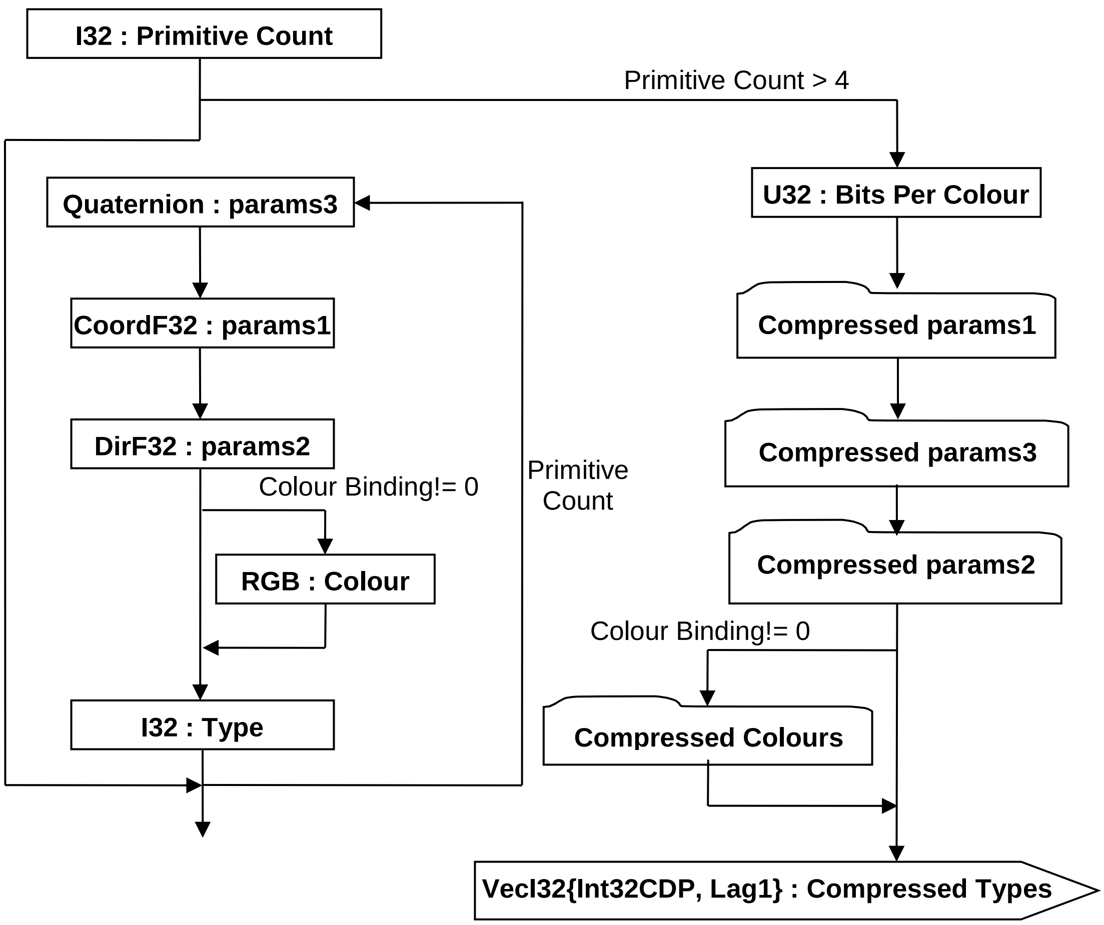


Figure 99 — Lossy Quantized Primitive Set Data collection

I32 : Primitive Count

Primitive Count specifies the number of primitives in the Primitive Set.

Quaternion : params3

Interpretation of params3 data field is primitive Type dependent. See Table for Primitive Set "params#" Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the params3 data fields.

CoordF32 : params1

Interpretation of params1 data field is primitive Type dependent. See Table forPrimitive Set "params#" Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the params1 data fields.

DirF32 : params2

Interpretation of params1 data field is primitive Type dependent. See Table for Primitive Set "params#" Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the params1 data fields.

RGB : Colour

Colour specifies the Red, Green Blue colour components for the primitive. This data field is only present if previously read Colour Binding (see Primitive Set Shape Element) is not equal to "0".

I32 : Type

Type specifies the primitive type. See Table for Lossless Compressed Primitive Set Data Field values in Lossless Compressed Primitive Set Data for valid primitive Type values.

U32 : Bits Per Colour

Bits Per Colour specifies the number of quantization bits per colour component. Value shall be within range [0:32] inclusive.

VecI32{Int32CDP, Lag1} : Compressed Types

The Compressed Types data field is a vector of Type data for all the primitives in the Primitive Set. Compressed Types uses the Int32 version of the CODEC to compress and encode data. In an uncompressed form the valid primitive Type vales are as documented in Table for Lossless Compressed Primitive Set Data Field values in Lossless Compressed Primitive Set Data.

Compressed params1

Compressed params1 is the compressed representation of the *params1* data for all the primitives in the Primitive Set. Note that the interpretation of the uncompressed *params1* data is primitive Type dependent. See Table for Primitive Set "params#" Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the *params1* data fields.

The *params1* data for all primitives in the Primitive Set is compressed/encoded on a per ordinate basis using a separate Uniform Quantizer (with Bits Per Vertex number of quantization bits) for each collection of ordinate values. Since *params1* is of type "CoordF32", it has three ordinate values (three F32 values), and thus three Uniform Quantizers (where a Uniform Quantizer is a scalar quantizer/encoder whose range is divided into levels of equal spacing). See Data Compression and Encoding for more complete description of Uniform Quantizer.

The JT Format packs all the *params1* data for all primitives into a single array using an ordinate dependent order (as shown below) and then encodes each of the lists of ordinate values using a separate Uniform Quantizer per ordinate list.

```
{prim1 params1[0], prim2 params1[0],…primN params1[0],
 prim1 params1[1], prim2 params1[1],…primN params1[1],
 prim1 params1[2], prim2 params1[2],…primN params1[2]}
```

The result of the Uniform Quantizer encoding is a range min and max floating point value pairs for each ordinate value collection, and an integer array of *params1* quantization codes that corresponds to the above described “ordinate dependent order” packed array of *params1* data.

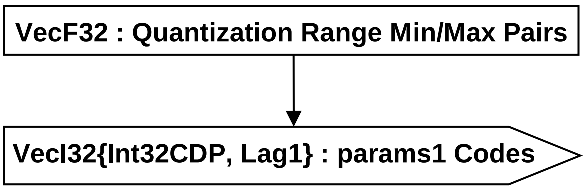


Figure 100 — Compressed params1 data collection

VecF32 : Quantization Range Min/Max Pairs

Quantization Range Min/Max Pairs is a vector of Uniform Quantizer range min/max value pairs. There shall be a min/max pair for each ordinate value collection (i.e. each Uniform Quantizer). Thus the length of this vector is “2 * num_ordinates” (so vector length would be “6” for *params1* data).

VecI32{Int32CDP, Lag1} : params1 Codes

The params1 Codes data field is a vector of quantizer “codes” for the *params1* data of all the primitives in the Primitive Set. The params1Codes also uses the Int32 version of the CODEC to compress and encode data.

Compressed params3

Compressed params3 is the compressed representation of the *params3* data for all the primitives in the Primitive Set. Note that the interpretation of the uncompressed *param31* data is primitive Type dependent. See Table 52 — Primitive Set “params#” Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the *params3* data fields.

The *params3* data for all primitives in the Primitive Set is compressed/encoded on a per ordinate basis using a separate Uniform Quantizer (with Bits Per Vertex number of quantization bits) for each collection of ordinate values. Since *params1* is of type “Quaternion”, it has four ordinate values (four F32 values), and thus four Uniform Quantizers (where a Uniform Quantizer is a scalar quantizer/encoder whose range is divided into levels of equal spacing). See Data Compression and Encoding for more complete description of Uniform Quantizer.

The JT Format packs all the *params3* data for all primitives into a single array using an ordinate dependent order (as shown below) and then encodes each of the lists of ordinate values using a separate Uniform Quantizer per ordinate list.

```
{prim1 params3[0], prim2 params3[0],…primN params3[0],
 prim1 params3[1], prim2 params3[1],…primN params3[1],
 prim1 params3[2], prim2 params3[2],…primN params3[2],
 prim1 params3[3], prim2 params3[3],…primN params3[3]}
```

The result of the Uniform Quantizer encoding is a range min and max floating point value pairs for each ordinate value collection, and an integer array of *params3* quantization codes that corresponds to the above described “ordinate dependent order” packed array of *params3* data.

The storage format of Compressed params3 is exactly the same as that documented in Figure for Compressed params1 data collection.

Compressed params2

Compressed params2 is the compressed representation of the *params2* data for all the primitives in the Primitive Set. Note that the interpretation of the uncompressed *params2* data is primitive Type dependent. See Table for Primitive Set "params#" Data Fields Interpretation in Lossless Compressed Primitive Set Data for per-primitive type description of the *params2* data fields.

The *params2* data for all primitives in the Primitive Set is compressed/encoded on a per ordinate basis using a separate Uniform Quantizer (with Bits Per Vertex number of quantization bits) for each collection of ordinate values. Since *params2* is of type "DirF32", it has three ordinate values (three F32 values), and thus three Uniform Quantizers (where a Uniform Quantizer is a scalar quantizer/encoder whose range is divided into levels of equal spacing). See Data Compression and Encoding for more complete description of Uniform Quantizer.

The JT Format packs all the *params2* data for all primitives into a single array using an ordinate dependent order (as shown below) and then encodes each of the lists of ordinate values using a separate Uniform Quantizer per ordinate list.

```
{prim1 params2[0], prim2 params2[0],…primN params2[0],
 prim1 params2[1], prim2 params2[1],…primN params2[1],
 prim1 params2[2], prim2 params2[2],…primN params2[2]}
```

The result of the Uniform Quantizer encoding is a range min and max floating point value pairs for each ordinate value collection, and an integer array of *params2* quantization codes that corresponds to the above described "ordinate dependent order" packed array of *params2* data.

The storage format of Compressed params2 is exactly the same as that documented in Figure for Compressed params1 data collection.

Compressed Colours

Compressed Colours is the compressed representation of the *colour* data for all the primitives in the Primitive Set. This data collection is only present if previously read Colour Binding (see Primitive Set Shape Element) is not equal to "0".

The *colour* data for all primitives in the Primitive Set is compressed/encoded on a per ordinate basis using a separate Uniform Quantizer (with Bits Per Colour number of quantization bits) for each collection of ordinate values. Since *colour* is of type "RGB", it has three ordinate values (three F32 values), and thus three Uniform Quantizers (where a Uniform Quantizer is a scalar quantizer/encoder whose range is divided into levels of equal spacing). See Data Compression and Encoding for more complete description of Uniform Quantizer.

The JT Format packs all the *colour* data for all primitives into a single array using an ordinate dependent order (as shown below) and then encodes each of the lists of ordinate values using a separate Uniform Quantizer per ordinate list.

```
{prim1 colour[0], prim2 colour[0],…primN colour[0],
 prim1 colour[1], prim2 colour[1],…primN colour[1],
 prim1 colour[2], prim2 colour[2],…primN colour[2]}
```

The result of the Uniform Quantizer encoding is a range min and max floating point value pairs for each ordinate value collection, and an integer array of *colour* quantization codes that corresponds to the above described "ordinate dependent order" packed array of *colour* data.

The storage format of Compressed Colours is exactly the same as that documented in Figure for Compressed params1 data collection.

9 Geometry Segments

The Geometry Segments in JT contain Element that define a range of geometry definitions that can be included with a JT file. Complete descriptions for each geometry segment can be foune in the Geometry Annexes.

9.1 XT B-Rep Element

XT B-Rep Segment contains an Element that defines the precise geometric Boundary Representation data for a particular Part in XT boundary representation format.

XT B-Rep Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element).

The XT B-Rep Segment type supports LZMA compression on all element data, so all elements in XT B-Rep Segment use the Logical Element Header LZMA form of element header data.

Object Type ID: 0x873a70e0, 0x2ac9, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

See Annex **F** XT B-rep Segment for a full description of the XT B-Rep Segment.

9.2 JT ULP Segment

The JT ULP Segment contains an Element that defines the lightweight geometric Boundary Representation data for a particular Part in the JT ULP format.

JT ULP Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT ULP Segment type supports compression on all element data, so all elements in JT ULP Segment use the Logical Element Header Compressed form of element header data.

Object Type ID: 0xf338a4af, 0xd7d2, 0x41c5, 0xbc, 0xf2, 0xc5, 0x5a, 0x88, 0xb2, 0x1e, 0x73

See Annex G JT ULP Segment for a full description of the legacy ULP segment.

9.3 JT Smart Topology Table (STT) Segment

JT Smart Topology Table (hereafter referred to as STT) Segment contains an Element that defines the lightweight B-Rep description for a particular Part.

JT STT Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT STT Segment type supports compression on all element data, so all elements in JT STT Segment use the Logical Element Header Compressed form of element header data.

Object Type ID: 0xca7e6f89, 0x97c8, 0x47f0, 0x9f, 0xca, 0x16, 0x99, 0xc, 0xfb, 0xe2, 0x17

See Annex H JT Smart Topology Table (STT) Segment for a full description of the STT Segment.

9.4 JT LWPA Segment

JT LWPA Segment contains an Element that defines light weight precise analytic data for a particular part. More specifically LWPA contains the collection of analytic surfaces in the B-Rep definition of the part.

JT LWPA Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT LWPA

Segment type supports LZMA compression on all element data, so all elements in JT LWPA Segment use the Logical Element Header Compressed form of element header data.

Object Type ID: 0xd67f8ea8, 0xf524, 0x4879, 0x92, 0x8c, 0x4c, 0x3a, 0x56, 0x1f, 0xb9, 0x3a

See Annex I JT LWPA Segment for a full description of the JT LWPA Segment.

9.5 Wireframe Segment

The Wireframe Segment contains an Element that defines the precise 3D wireframe data for a particular Part.

A Wireframe Segment is typically referenced by a Part Node Element using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The Wireframe Segment type supports LZMA compression on all element data, so all elements in Wireframe Segment use the Logical Element Header Compressed form of element header data.

Object Type ID: 0x873a70d0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

See Annex J Wireframe Segment for a full description of the JT Wireframe Segment.

9.6 JT B-Rep Element (deprecated)

JT B-Rep Element represents a particular Part's precise data in JT boundary representation format.

JT B-Rep Segments are typically referenced by Part Node Elements using Late Loaded Property Atom Elements.

The JT B-Rep Segment type supports LZMA compression on all element data, so all elements in JT B-Rep Segment use the Logical Element Header LZMA form of element header data.

Object Type ID: 0x873a70c0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

See Annex K JT B-rep Segment for a full description of the JT B-Rep Segment.

NOTE: JT B-Rep is deprecated and should be considered read only for application creation.

10 Meta Data Segment

Meta Data Segments are used to store large collections of meta-data in separate addressable segments of the JT File. Storing meta-data in a separate addressable segment allows references (from within the JT file) to these segments to be constructed such that the meta-data can be late-loaded (i.e. JT file reader can be structured to support the "best practice" of delaying the loading/reading of the referenced meta-data segment until it is actually needed).

Meta Data Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element).

The Meta Data Segment type supports compression on all element data, so all elements in Meta Data Segment use the Logical Element Header Compressed form of element header data.

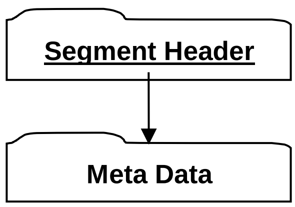


Figure 101 — Meta Data Segment data collection

Complete description for Segment Header can be found in Segment Header.

The following sub-sections document the various Meta Data Element types.

10.1 Property Proxy Meta Data Element

Object Type ID: 0xce357247, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1

A Property Proxy Meta Data Element serves as a “proxy” for all meta-data properties associated with a particular Meta Data Node Element (see Meta Data Node Element). The proxy is in the form of a list of key/value property pairs where the *key* identifies the type and meaning of the *value*. Although the property *key* is always in the form of a String data type, the *value* can be one of several data types.

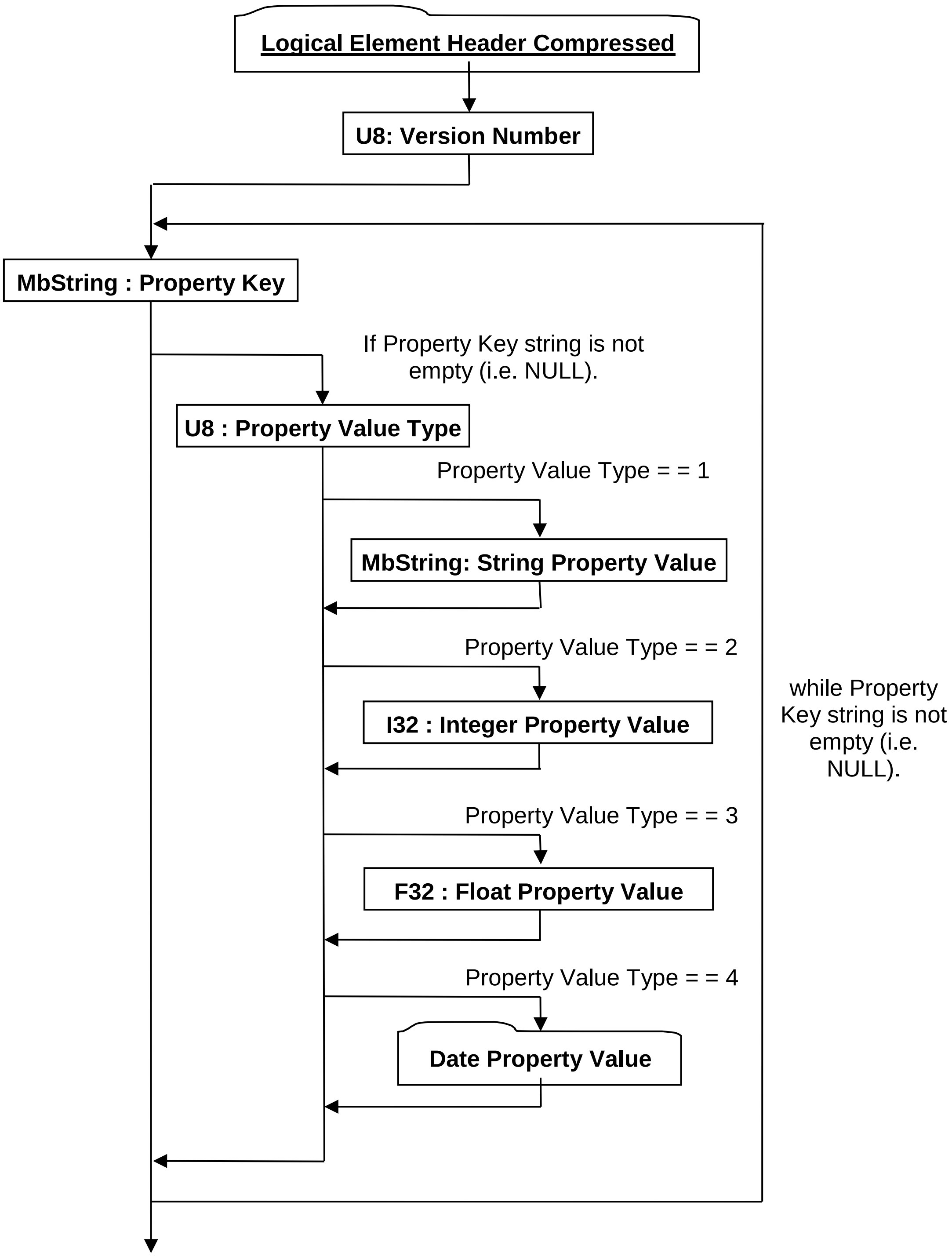


Figure 102 — Property Proxy Meta Data Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

MbString: Property Key

Property Key specifies the *key* string for the property.

U8: Property Value Type

Property Value Type specifies the data type for the Property Value. If the type equals “0” then no Property Value is written. Valid types include the following:

Table 53 — Property Proxy Meta Data Property Value Type values

= 0	Unknown
= 1	MbString data type value
= 2	I32 data type value
= 3	F32 data type value
= 4	Date value

MbString: String Property Value

String Property Value represents the property value when Property Value Type = = 1.

I32: Integer Property Value

Integer Property Value represents the property value when Property Value Type = = 2.

F32: Float Property Value

Float Property Value represents the property value when Property Value Type = = 3.

Date Property Value

Date Property Value represents the property value when Property Value Type = = 4. Date Property Value data collection represents a date as a combination of year, month, day, hour, minute, and second data fields.

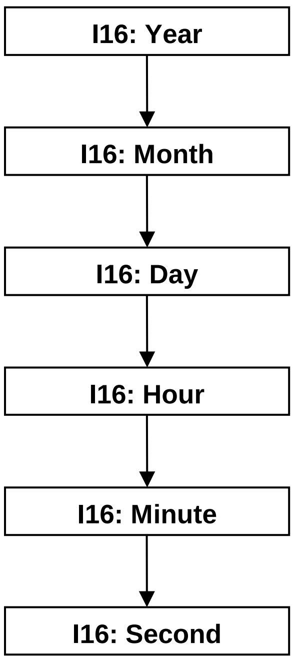


Figure 103 — Date Property Value data collection

I16: Year

Year specifies the date year value.

I16: Month

Month specifies the date month value.

I16: Day

Day specifies the date day value.

I16: Hour

Hour specifies the date hour value.

I16: Minute

Minute specifies the date minute value.

I16: Second

Second specifies the date Second value.

10.2 PMI Manager Meta Data Element

Object Type ID: 0xce357249, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1

The PMI Manager Meta Data Element data collection is a type of Meta Data Element which contains the Product and Manufacturing Information for a part/assembly.

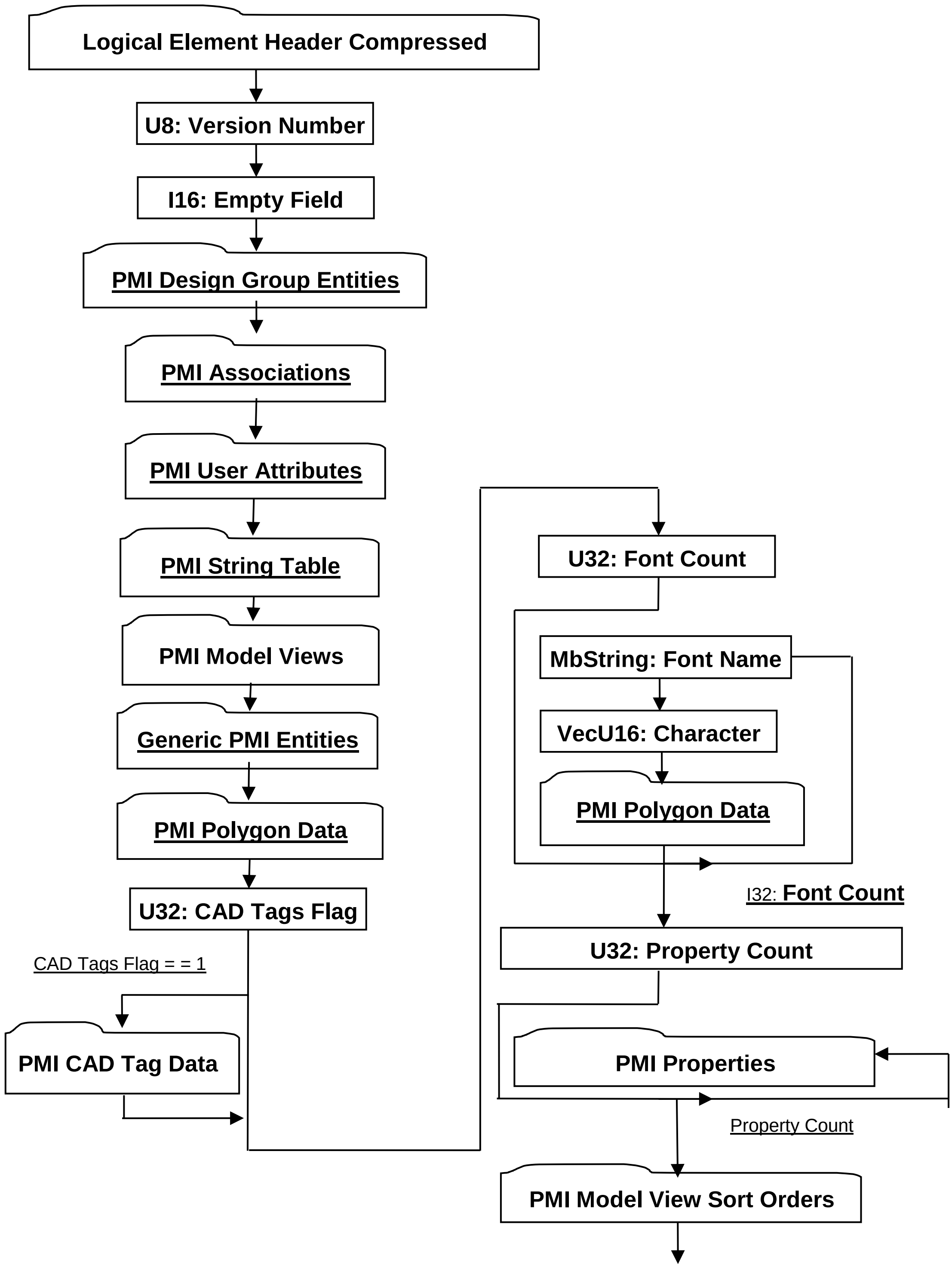


Figure 104 — PMI Manager Meta Data Element data collection

A complete description of Logical Element Header Compressed can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

U8: Version Number

Version Number is the version identifier for this PMI Manager Element. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I16: Empty Field

Refer to Common Data Conventions and Constructs Empty Field.

U32: CAD Tags Flag

CAD Tags Flag is a flag indicating whether CAD Tag data exist for the PMI.

U32: Font Count

U32: Font Count specifies the number of sets of glyph definitions. Each set of glyphs represents a single font definition that consists of a name, a character set and polygonal glyph definition for each character in the set.

MbString: Font Name

Font name specifies a representative name for the font set.

VecU16: Character Set

VecU16:Character Set contains the unsigned 16-bit integer identifiers for each character whose symbol is defined in the ensuing PolygonData segment.

U32: Property Count

Property Count specifies the number of PMI Properties.

10.2.1 PMI Design Group Entities

The PMI Design Group Entities data collection defines data for a list of Design Groups. Design Groups are collections of PMI created to organize a model into smaller subsets of information. This organization is achieved via PMI Associations (see PMI Associations), where specific PMI entities are associated as “destinations” to a “source” PMI Design Group.

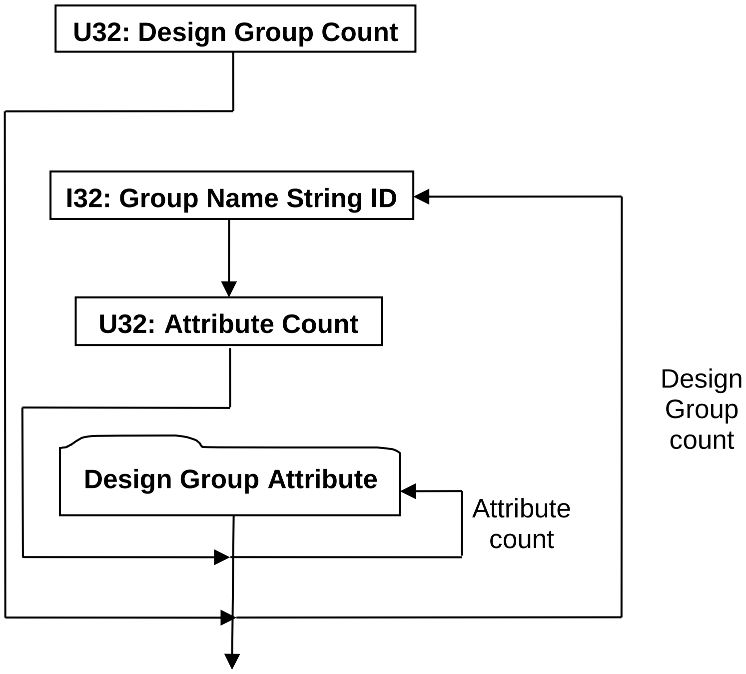


Figure 105 — PMI Design Group Entities data collection

U32: Design Group Count

Design Group Count specifies the number of Design Group entities.

I32: Group Name String ID

Group Name String ID specifies the identifier for the group name character string. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

U32: Attribute Count

Attribute Count specifies the number of Design Group Attribute data collections.

Design Group Attribute

The Design Group Attribute data collection defines a group property/attribute.

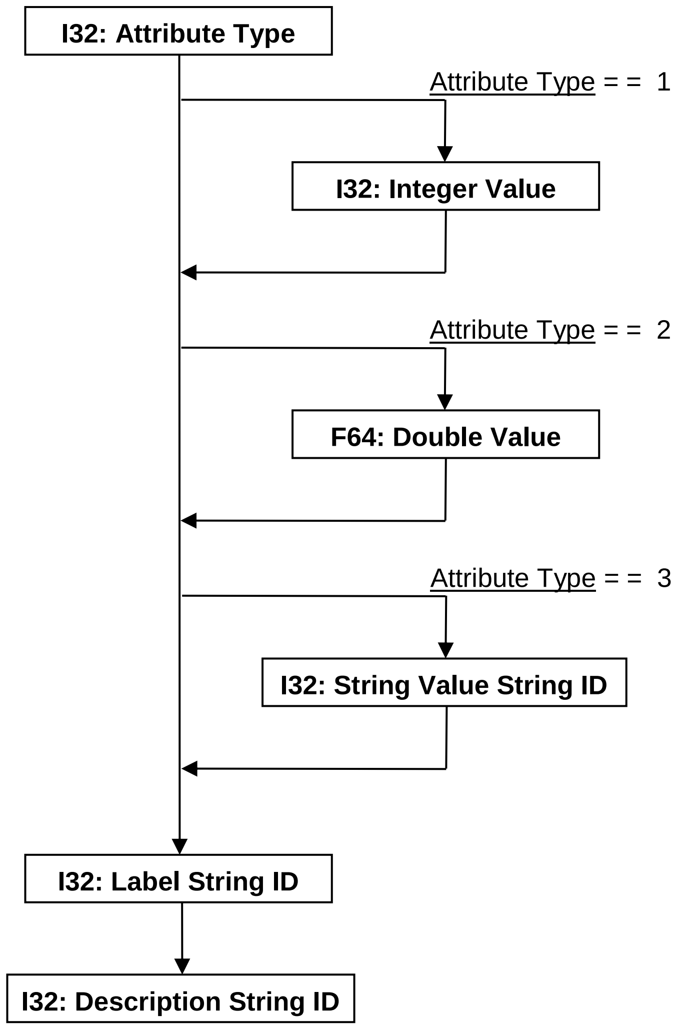


Figure 106 — Design Group Attribute data collection

I32: Attribute Type

Attribute Type specifies the attribute type. Valid types include the following:

Table 54 — PMI Design Group Attribute Type values

= 1	Integer
= 2	Double
= 3	String

I32: Integer Value

Integer Value specifies the value for “integer” Attribute Types.

F64: Double Value

Double Value specifies the value for “double” Attribute Types.

I32: String Value String ID

String Value String ID specifies the string identifier value for “string” Attribute Types. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

I32: Label String ID

Label String ID specifies the string identifier for the attribute label. This identifier is an index to a particular character string in the PMI String Table as defined inPMI String Table. An identifier value of “-1” indicates no string.

I32: Description String ID

Description String ID specifies the string identifier for the attribute description. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

10.2.2 PMI Associations

The PMI Associations data collection defines data for a list of associations. An association defines a link (“relationship”) between two PMI, B-Rep, or Wireframe Rep entities where one entity is defined as the “source” and the other entity is defined as the “destination”.

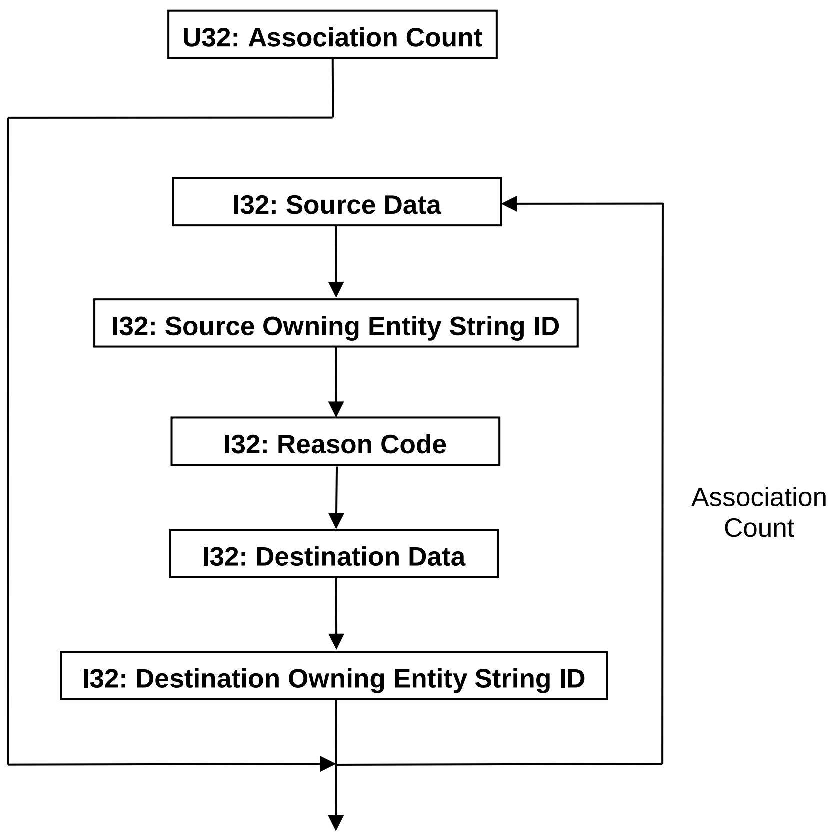


Figure 107 — PMI Associations data collection

U32: Association Count

Association Count specifies the number of associations.

I32: Source Data

Source Data is a collection of source entity information encoded/packed within a single I32 using the following bit allocation. All bits fields that are not defined as in use should be set to “0”.

Table 55 — PMI Associations Source Data values

Bits 0 - 23	Source Entity Identifier. The interpretation of this identifier data is dependent upon the value of Bit 31 documented below.
Bits 24 -30	Source Entity PMI or B-Rep type. Valid types include the following: = 0 PMI - Dimension = 1 PMI - Note = 2 PMI - Datum Feature Symbol = 3 PMI - Datum Target = 4 PMI - Feature Control Frame = 5 PMI - Line Weld = 6 PMI - Spot Weld = 7 PMI - Measurement Point = 8 PMI - Surface Finish = 9 PMI - Locator Designator = 10 PMI - Reference Geometry = 11 PMI - Coordinate System = 12 PMI - Design Group = 13 PMI - User Attribute = 14 B-Rep - Vertex = 15 B-Rep - Edge = 16 B-Rep - Face = 17 PMI - Model View = 18 PMI - Generic = 19 Wireframe Rep - Edge = 20 PMI - Unspecified type = 21 Part Instance = 22 B-rep body = 23 Group
Bit 31	Indirect Identifier Flag = 0 – Value in Bits 0-23 is not the actual CAD identifier, instead Bits 0-23 is an index into the source type’s PMI array or index of the edge/face in B-Rep or Wireframe Rep for the source entity. = 1 – Value in Bits 0-23 is not the actual CAD identifier; instead Bits 0-23 is an index into the list of CAD Tags (as documented in PMI CAD Tag Data) identifying the CAD Tag belonging to the particular source entity.

I32: Source Owning Entity String ID

Source Owning Entity String ID specifies the string identifier for the string which contains the unique CAD identifier of the component (part or assembly) that owns the source PMI or B-Rep entity. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string and implies that the entity is to be found on the current node‘s PMI/B-Rep/Wireframe-Rep segment. It is valid for the source owning entity to be the same as the destination owning entity (i.e. an association between two PMI or B-Rep entities in the same part/assembly).

I32: Reason Code

Reason Code specifies the “reason” for the association. Valid Reason Codes include the following:

Table 56 — PMI Associations Reason Code values

= 0	Association is to the primary entity being dimensioned.

= 1	Association is to the secondary entity being dimensioned.
= 2	Association is to the dimension plane.
= 5	Association is to the entity used to specify the Z-Axis of a coordinate system.
= 10	Association is to an entity "associated" to or "included in" a PMI symbol.
= 11	Association is to an entity used to "attach" a PMI symbol.
= 12	Association is to first entity used to “attach” a PMI symbol.
= 13	Association is to second entity used to “attach” a PMI symbol.
= 14	Specifying PMI grouping, source is PMI/B-Rep entity and destination is design group.
= 15	Association is to a weld line entity.
= 16	Association is to a “hot spot”.
= 20	Accociation is to a PMI B-rep entity contained in a virtual group. Similar to reason code 14 but for a virtual group
= 17	Association is to a child in a PMI stack.
= 72	Association is for PMI miscellaneous relation.
= 73	Association is for PMI related entity.
= 78	Association is to a product instance to cut by a PMI section. Only required for partial scene sectioning
= 98	Association is to show the PMI when associated Model View is selected. Source is the PMI, and destination is Model View.
= 99	Association is to show/select PMI B, if showing/selecting PMI A. Source is PMI A, and destination is PMI B. This is different from an “attached” PMI, where the convention is to show the PMI visibly linked to one another.
= 100	Association is to show all parts except the associated part instance. Source is the part instance, and destination is Model View.

I32: Destination Data

Destination Data is a collection of destination entity information encoded/packed within a single I32. The encoding schema and interpretation of this data is the same as that documented in Source Data.

I32: Destination Owning Entity String ID

Destination Owning Entity String ID specifies the string identifier for the string which contains the unique CAD identifier of the component (part or assembly) that owns the destination PMI or B-Rep entity. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string and implies that the entity is to be found on the current node‘s PMI/B-Rep/Wireframe-Rep segment. It is valid for the source owning entity to be the same as the destination owning entity (i.e. an association between two PMI or B-Rep entities in the same part/assembly).

10.2.3 PMI User Attributes

The PMI User Attributes collection defines data for a list of user attributes. PMI User Attributes are used to add attribute data to a part/assembly. Each user attribute is composed of key/value pair of strings.

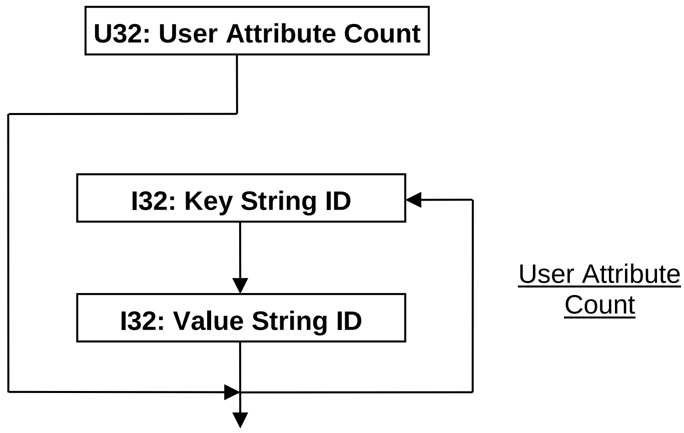


Figure 108 — PMI User Attributes data collection

U32: User Attribute Count

User Attribute Count specifies the number of user attributes.

I32: Key String ID

Key String ID specifies the string identifier for the user attribute key. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

I32: Value String ID

Value String ID specifies the string identifier for the user attribute value. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

10.2.4 PMI String Table

The PMI String Table data collection defines data for a list of character strings and serves as a central repository for all character strings used by other PMI Entities within the same PMI Manager Meta Data Element. PMI Entities reference into this list/array of character strings to define usage of a particular character string using a simple list/array “index” (i.e. String ID).

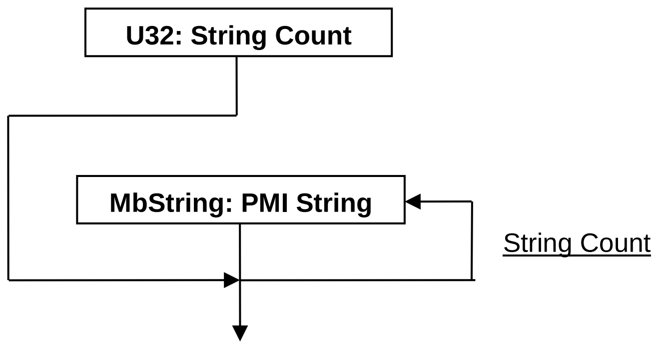


Figure 109 — PMI String Table data collection

U32: String Count

String Count specifies the number of character strings in the string table.

MbString: PMI String

PMI String specifies the character string.

10.2.5 PMI Model Views

The PMI Model Views data collection defines data for a list of Model Views. A fully annotated part/assembly may contain so much PMI information, that it becomes very difficult to interpret the design intent when viewing a 3D Model (with PMI visible) of the part/assembly. Model Views provide a means to capture and organize PMI information about a 3D model so that the design intent can be clearly interpreted and communicated to others in later stages of the Product Lifecycle Management (PLM) process (e.g. manufacturing, inspection, assembly). This organization is achieved via PMI Associations (see PMI Associations), where specific PMI entities are associated as “destinations” to a “source” PMI Model View.

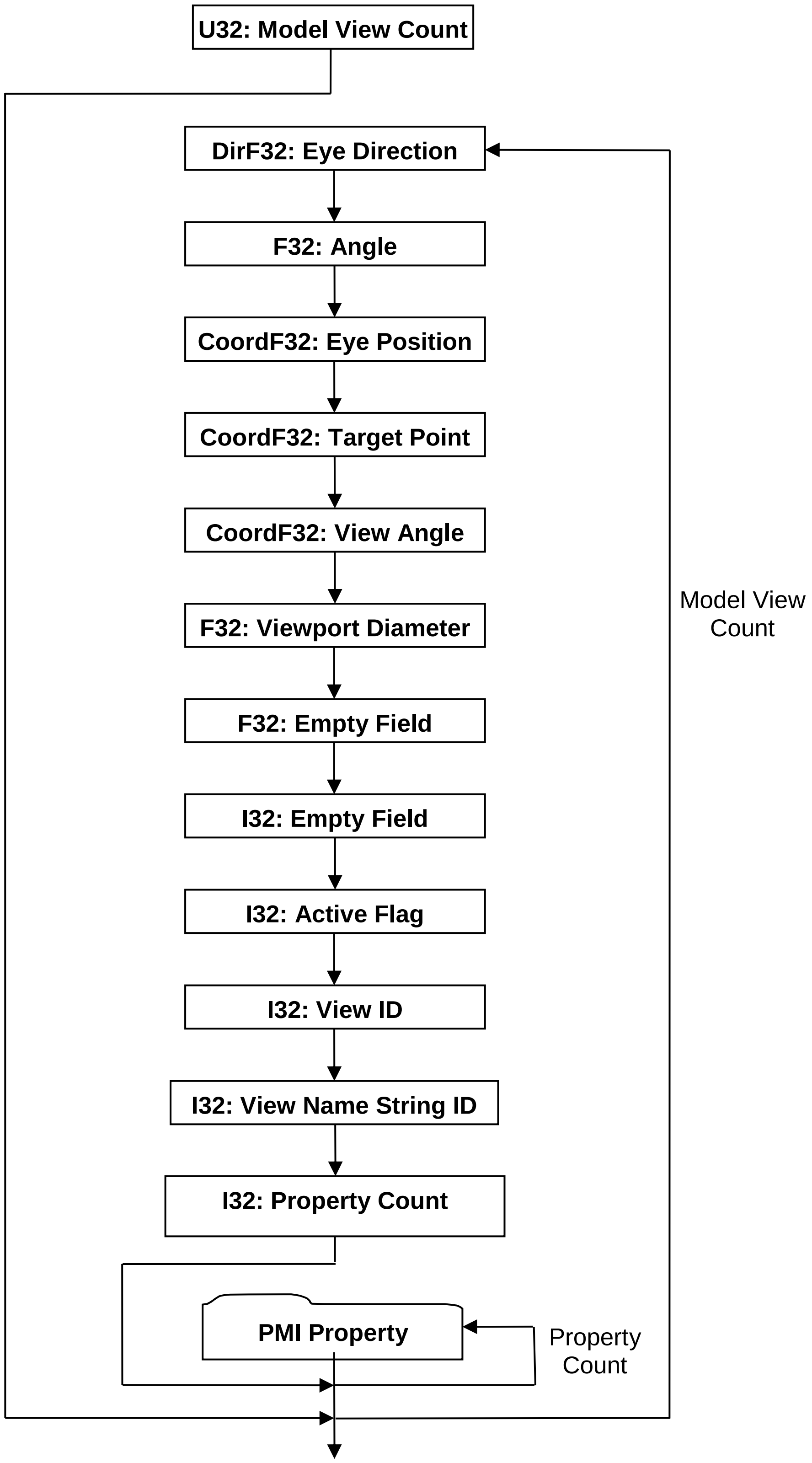


Figure 110 — PMI Model Views data collection

U32: Model View Count

Model View Count specifies the number of Model Views.

DirF32: Eye Direction

Eye Direction specifies the camera direction vector.

F32: Angle

Angle specifies the camera rotation angle (in degrees where positive is counter-clockwise) about the Eye Direction. So this Angle in combination with the Eye Direction is equivalent to specifying a rotation using axis-angle representation.

CoordF32: Eye Position

Eye Position specifies the WCS coordinates of the eye/camera “look from” position.

CoordF32: Target Point

Target Point specifies the WCS coordinates of the eye/camera “look at” position.

CoordF32: View Angle

View angle specifies the X, Y, Z rotation angles (in degrees) of the model’s axis. The rotations are defined with respect to an initial orientation where the model’s axis are aligned with the screen’s axis (i.e. +X axis points to right, +Y axis points up, +Z axis points out at you).

F32: Viewport Diameter

Viewport Diameter specifies the diameter in WCS coordinates of the largest possible circle that could be inscribed within viewport. If a large diameter value is specified, the model appears very small in relation to the viewport; whereas if a small diameter value is specified a close-up (“zoomed-in)” view of the model results.

F32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field.

I32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field.

I32: Active Flag

Active Flag is a flag specifying whether this Model View is the “active” view. Valid values include the following:

Table 57 — PMI Model Views Active Flag values

= 0	Is not the active Model View.
= 1	Is the active Model View

I32: View ID

View ID specifies the Model View unique identifier.

I32: View Name String ID

View Name String ID specifies the string identifier for the Model View’s name. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

PMI Property

A PMI Property data collection consists of a key/value pair and is used to describe attributes of Generic PMI Entity or other specific data.

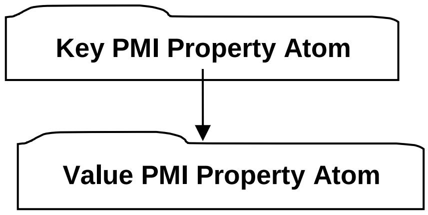


Figure 111 — PMI Property data collection

Both Key and Value have the same format as that documented in Key PMI Property Atom.

For a full description of PMI "Key" strings and "Value" string encoding format descriptions see the PMI Properites Annex in this document.

Key PMI Property Atom

Key PMI Property Atom data collection represents the data format for both the key and value data of a PMI Property key/value pair.

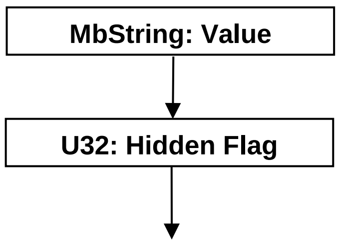


Figure 112 — Key PMI Property Atom data collection

MbString: Value

Value specifies the property atom value encoded into a String. See Table 58 — Common Property Keys and Their Value Encoding formats above for encoding formats of the Value string.

U32: Hidden Flag

Hidden Flag specifies if the property is "hidden" or not. A JT file reader could use this flag to control whether read properties should be exposed to the end user of the application reading the JT file. Valid values include the following:

Table 58 — PMI Property Atom Hidden Flag values

= 0	Property is not hidden.
= 1	Property is hidden.

10.2.6 Generic PMI Entities

The Generic PMI Entity data collection provides a “generic” format for defining various PMI entity types, including user defined types. The generic format defines the data making up the PMI Entity through a combination of the PMI 2D Data collection and a list of PMI Property data collections.

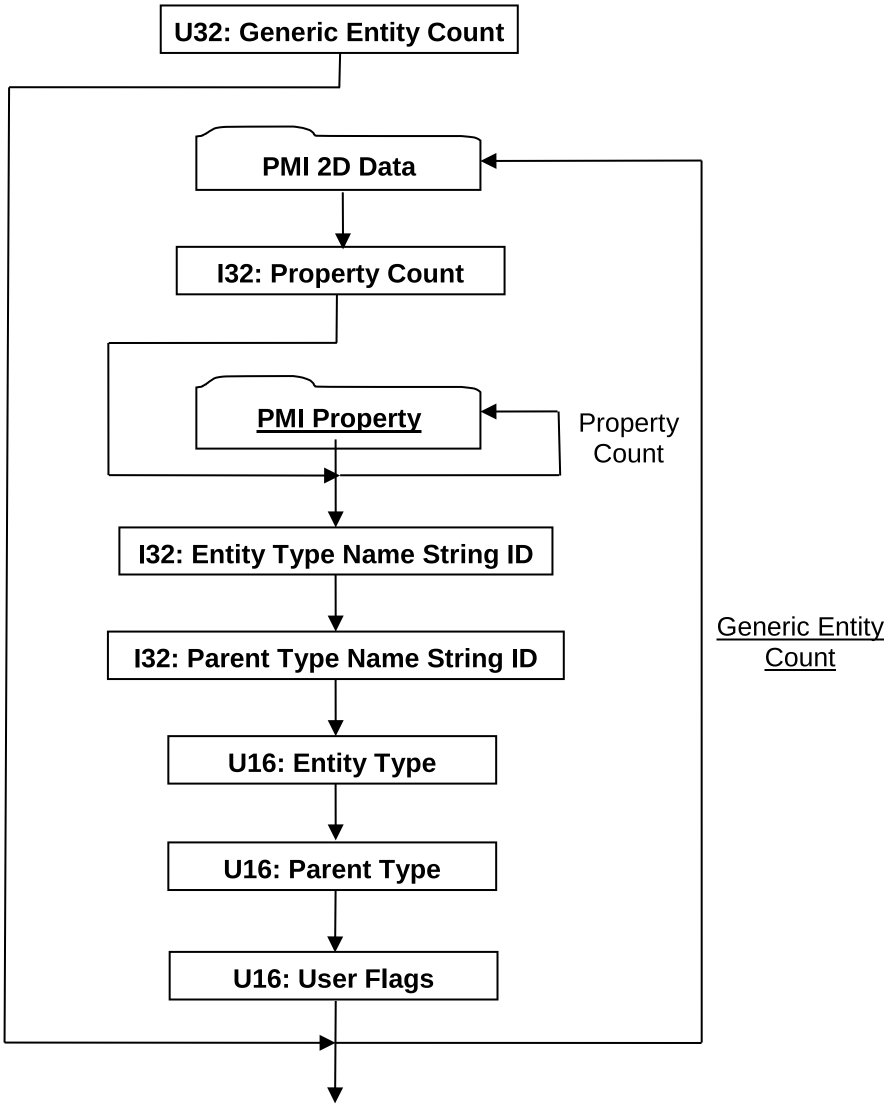


Figure 113 — Generic PMI Entity data collection

Complete description for PMI Property can be found in the logical collection for_PMI Property found in PMI Model View.

U32: Generic Entity Count

Generic Entity Count specifies the number of Generic PMI Entities.

I32: Property Count

Property Count specifies the number of PMI Properties.

I32: Entity Type Name String ID

Entity Type Name String ID specifies the string identifier for the name of the Generic PMI Entity Type. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of "-1" indicates no string.

I32: Parent Type Name String ID

Parent Type Name String ID specifies the string identifier for the name of the parent Generic PMI Entity Type. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of "-1" indicates no string.

U16: Entity Type

Entity Type specifies the Generic PMI Entity Type. The valid Entity Type values (in hexadecimal format) are documented in the following table. Note that for "user defined" Generic PMI Entities a hexadecimal value of "0x0114" (as documented in table below) should be used.

Table 59 — Generic PMI Entity Type values

Entity Type	Description
0x0001	PMI (generally only used as a Parent Type)
0x0002	Weld
0x0004	Spot Weld
0x0008	Line Weld
0x0010	Groove Weld
0x0011	Fillet Weld
0x0012	Slot Weld
0x0014	Edge Weld
0x0018	Arc Spot Weld
0x0020	Resistance Spot Weld
0x0021	Resistance Seam Weld
0x0022	Structural Adhesive Bead Shaped
0x0024	Structural Adhesive Tape Shaped
0x0028	Structural Adhesive Dollop Shaped
0x0040	Mechanical Clinch Connector
0x0041	Surface Finish
0x0042	Measurement Point
0x0044	Datum Locator
0x0048	Certification Point
0x0080	Geometric Dimensioning and Tolerancing
0x0081	Feature Control Frame
0x0082	Dimension
0x0084	Datum Feature Symbol
0x0088	Datum Target
0x0100	Note
0x0101	Face Attribute Note
0x0102	Model View Label Note
0x0104	Coordinate System
0x0108	Reference Geometry
0x0110	Reference Point
0x0111	Reference Axis
0x0112	Reference Plane
0x0114	User Defined
0x0118	Measurement Locator
0x0120	Datum Point
0x0121	Surface Vector Measurement Point
0x0122	Hole Vector Measurement Point

Entity Type	Description
0x0124	Trimmed Sheet Vector Measurement Point
0x0128	Hem Vector Measurement Point
0x0230	Fastener PMI
0x0231	Material specification
0x0232	Process specification
0x0233	Part specification
0x0235	Balloon Note
0x0238	Circle Centre
0x0239	Coordinate Note
0x0240	AttributeNote
0x0241	Bundle or Dressing Note
0x0242	Cutting Plane Symbol
0x0243	Crosshatch
0x0244	E Marking (Note)
0x0245	Organization
0x0246	Region
0x0305	Section
0x0306	Centreline
0x0307	Fit Designation
0x0308	Composite Feature Control Frame
0x0309	Weld Note Type
0x030A	Reference Circle
0x030B	Reference Cylinder
0x030C	Part Transform
0x030D	Callout Dimension Type
0x030E	Parameter Dimension Type (reserved not in use)
0x030F	Chamfer Dimension Type
0x0310	Model View Style
0x0311	PMI Table Type
0x0312	Parameter Fit Designation Type (reserved not in use)
0x0313	Callout Fit Designation
0x8000	Feature Type
0x8001	Feature Thread Type
0x8002	Feature Arc Weld Type
0x8003	Feature Datum Type
0x8004	Feature Discrete Join Type
0x8005	Feature Resistance Weld Type
0x8006	Feature Continuous Join Type
0x8007	Feature Adhesive Fill Type
0x8008	Feature Surface Weld Type
0x8009	Feature Measurement Locator Type
0x800A	Feature Grouped Weld
0x800B	Feature Lazer Weld Type

U16: Parent Type

Parent Type specifies the parent Generic PMI Entity Type. The valid Parent Type values are the same as that documented above for Entity Type. The Parent Type is used to create a class hierarchy of PMI when presenting the PMI contents from a JT file.

U16: User Flags

User Flags is a collection of flags. The flags are combined using the binary OR operator and store various state information for the Generic PMI Entity. All bits fields that are not defined as in use should be set to “0”.

Table 60 — Generic PMI User Flag values

0x0001	Show PMI Entity “flat to screen only” flag = 0 – Allow PMI display plane to rotate with model. = 1 – Display PMI entity in the plane of the screen, so that it does not rotate with model.

PMI 2D Data

The PMI 2D Data collection defines a data format common to all 2D based PMI entities.

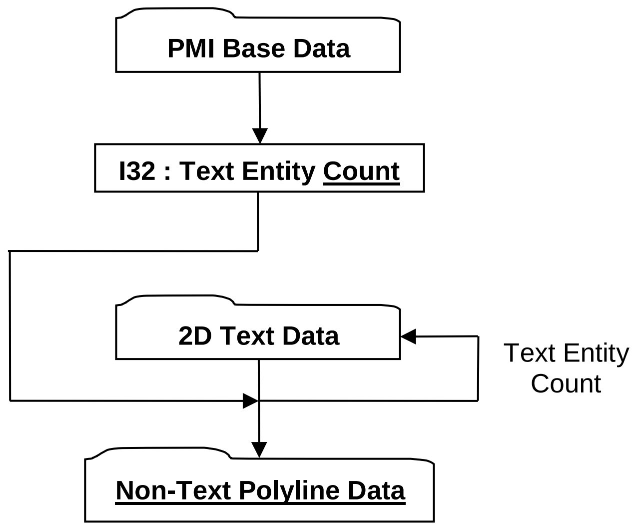


Figure 114 — PMI 2D Data collection

I32: Text Entity Count

Text Entity Count specifies the number of Text entities in the particular PMI entity.

PMI Base Data

The PMI Base Data collection defines the basic/common data that every 2D and 3D PMI entity.

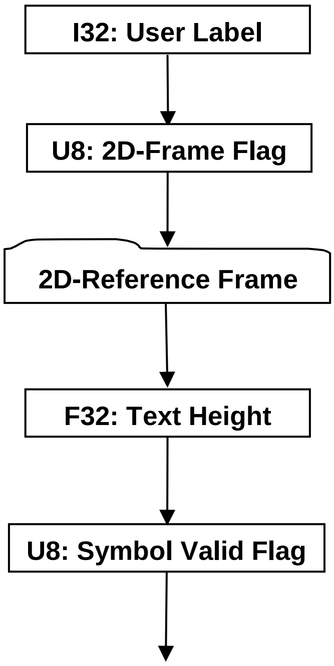


Figure 115 — PMI Base Data collection

I32: User Label

User Label specifies the particular PMI entity identifier.

U8: 2D-Frame Flag

2D-Frame Flag is a flag specifying whether 2D-Reference Frame data is stored. If 2D-Frame Flag has a non-zero value then 2D-Reference Frame data is included. If 2D-Frame Flag has a value of “2”, then dummy (i.e. all zeros) 2D-Reference Frame data is written. The “2D-Frame Flag = = 2” case is used by Generic PMI Entities because for Generic PMI Entities all the Non-Text Polyline Data is already in 3D form (i.e. XYZ coordinate data).

F32: Text Height

Text Height specifies the PMI text height in WCS.

U8: Symbol Valid Flag

Symbol Valid Flag is a flag specifying whether the particular PMI entity is valid. If Symbol Valid Flag has a non-zero value then PMI entity is valid.

2D-Reference Frame

The 2D-Reference Frame data collection defines a reference frame (2D coordinate system) where the PMI entity is displayed in 3D space. All the PMI entity’s 2D and 3D polyline data is assumed to lie on the defined plane.

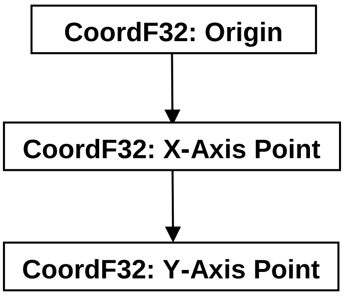


Figure 116 — 2D-Reference Frame data collection

CoordF32: Origin

Origin defines the origin (min-corner) of the 2D coordinate system.

CoordF32: X-Axis Point

X-Axis Point defines a point along the X-Axis of the 2D coordinate system.

CoordF32: Y-Axis Point

Y-Axis Point defines a point along the Y-Axis of the 2D coordinate system.

2D Text Data

The 2D Text Data collection defines a 2D text entity/primitive.

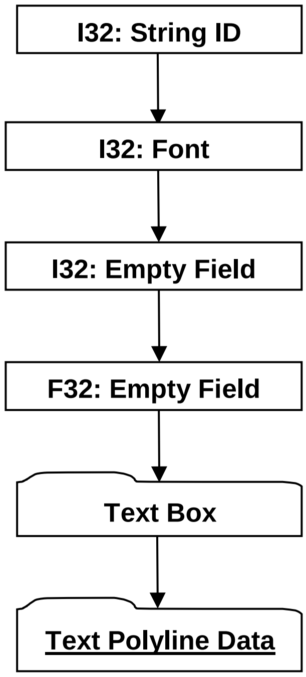


Figure 117 — 2D Text Data collection

I32: String ID

String ID specifies the identifier for the character string. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

I32: Font

Font identifies the font to be used for this text. Valid values include the following:

Table 61 — PMI 2D Base Data Font values

= 1	Simplex
= 2	Din
= 3	Military
= 4	ISO
= 5	Lightline
= 6	IGES 1001
= 7	Century
= 8	IGES 1002
= 9	IGES 1003
= 101	Japanese JISX 0208 coded character set
= 102	Japanese Extended Unix Codes JISX 0208 coded character set
= 103	Chinese GB 2312.1980 Simplified coded character set
= 104	Korean KSC 5601 coded character set
= 105	Chinese Big5 Traditional coded character set

I32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field.

F32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field.

Text Box

The Text Box data collection specifies a 2D box that particular text fits within. All values are with respect to 2D-Reference Frame documented in 2D-Reference Frame.

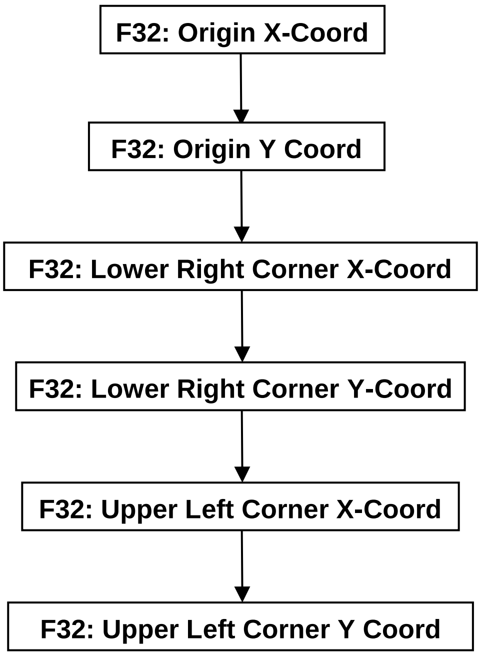


Figure 118 — Text Box data collection

F32: Origin X-Coord

Origin X-Coord defines the 2D X-coordinate of the text origin with respect to 2D-Reference Frame.

F32: Origin Y Coord

Origin Y-Coord defines the 2D Y-coordinate of the text origin with respect to 2D-Reference Frame.

F32: Lower Right Corner X-Coord

Lower Right Corner X-Coord defines the 2D X-coordinate of the lower right corner of the text with respect to 2D-Reference Frame.

F32: Lower Right Corner Y-Coord

Lower Right Corner Y-Coord defines the 2D Y-coordinate of the lower right corner of the text with respect to 2D-Reference Frame.

F32: Upper Left Corner X-Coord

Upper Left Corner X-Coord defines the 2D X-coordinate of the upper left corner of the text with respect to 2D-Reference Frame.

F32: Upper Left Corner Y Coord

Upper Left Corner Y-Coord defines the 2D Y-coordinate of the upper left corner of the text with respect to 2D-Reference Frame.

Text Polyline Data

The Text Polyline Data collection defines any polyline segments which are part of the text representation. This existence of this polyline data is conditional (i.e. not all text has it) and is made up of an array of indices into an array of polyline segments packed as 2D vertex coordinates, specifying where each polyline segment begins and ends. Polylines are constructed from these arrays of data as follows:

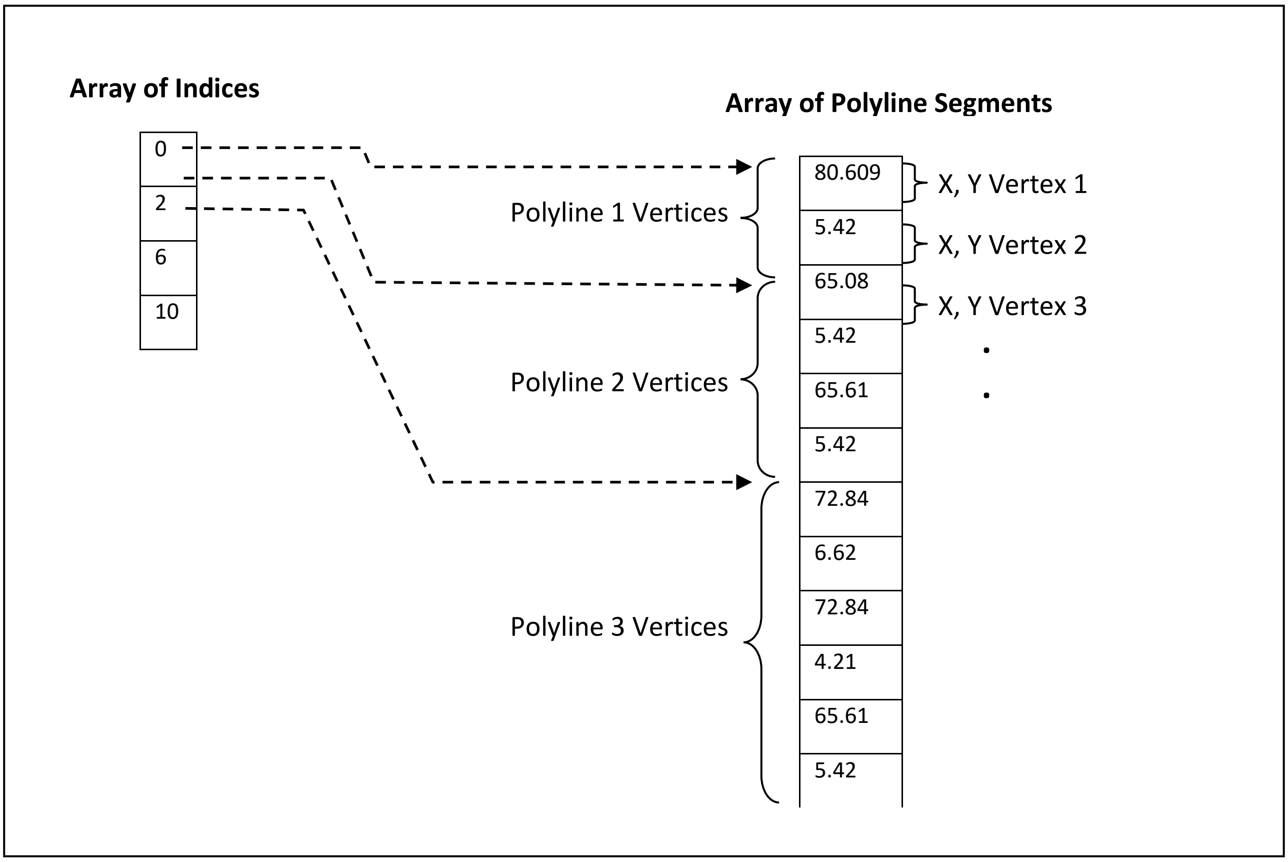


Figure 119 — Constructing Text Polylines from data arrays

This data is represented in JT file in the following format:

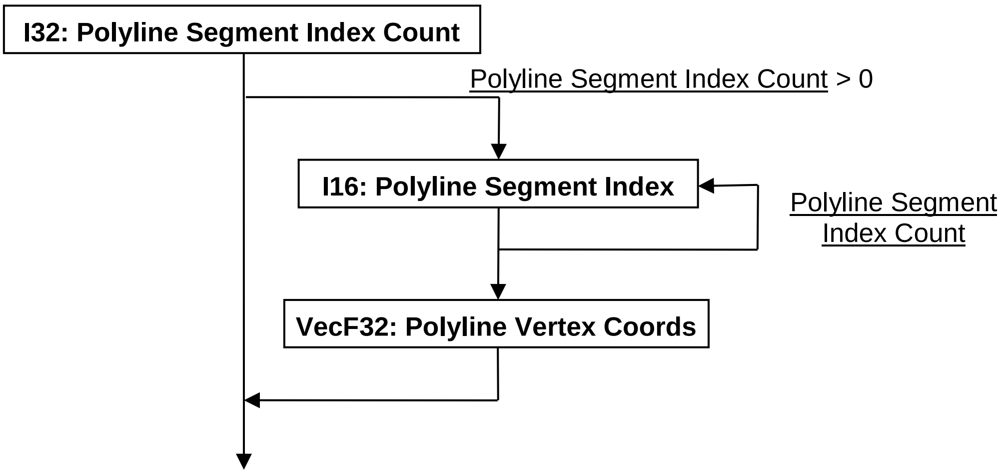


Figure 120 — Text Polyline Data collection

I32: Polyline Segment Index Count

Polyline Segment Index Count specifies the number of polyline segment indices.

I16: Polyline Segment Index

Polyline Segment Index is an index into the Polyline Vertex Coords array specifying where polyline segment begins or ends. This index is a vertex coordinate index so the absolute index into the Polyline Vertex Coords array is computed by multiplying the index value by "2" (i.e. for 2D coordinates).

VecF32: Polyline Vertex Coords

Polyline Vertex Coords is an array of polyline segments packed as 2D point coordinates. These 2D point coordinates are with respect to the 2D-Reference Frame documented in 2D-Reference Frame.

Non-Text Polyline Data

The Non-Text Polyline Data collection contains all the non-text polylines making up the particular PMI entity. Examples of non-text polylines include line attachments, text boxes, symbol box dividers, etc. The Non-Text Polyline Data collection is made up of an array of indices into an array of polyline segments packed as either 2D or 3D vertex coordinates, specifying where each polyline segment begins and ends. Whether vertex coordinates are 2D or 3D is dependent upon the PMI entity type using this data collection. If it is a Generic PMI Entities type then the packed coordinate data is 3D; for all other PMI entity types the packed coordinate data is 2D. Two arrays of values that sequentially specify the polyline type and width in the polyline segments array are included.

Figure 127 below shows how Polylines are constructed from these arrays of data for the packed 2D coordinates case. The packed 3D coordinates case is interpreted the same except that the coordinates array includes a Z component and is thus packed as "[XYZ][XYZ][XYZ]…"

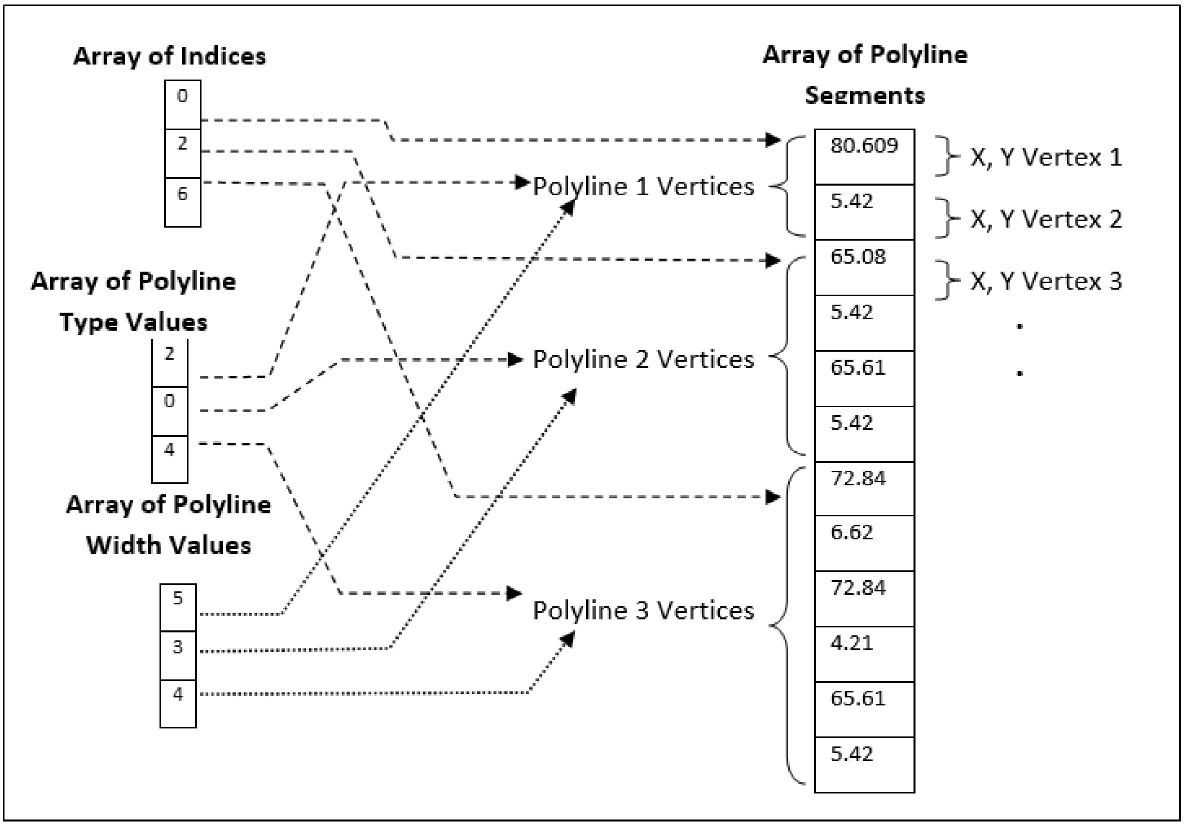


Figure 121 — Constructing Non-Text Polylines from packed 2D data arrays

This data is represented in the JT format as follows:

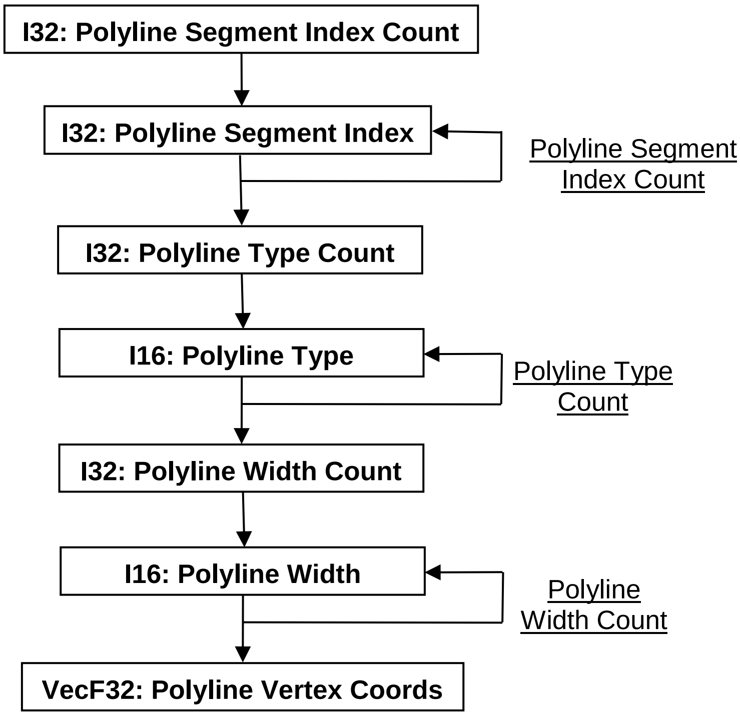


Figure 122 — Non-Text Polyline Data collection

I32: Polyline Segment Index Count

Polyline Segment Index Count specifies the number of polyline segment indices.

I32: Polyline Segment Index

Polyline Segment Index is an index into the Polyline Vertex Coords array specifying where polyline segment begins or ends. This index is a vertex/coordinate index so the absolute index into the Polyline Vertex Coords array is computed by multiplying the index value by "2" (i.e. for 2D coordinates).

I32: Polyline Type Count

Polyline Type Count specifies the number of polyline type values.

I16: Polyline Type

Polyline Type specifies the type of polyline segment in Polyline Vertex Coords array. See Figure 127 — Constructing Non-Text Polylines from packed 2D data arrays for interpretation of this array of type values relative to the defined polylines. Valid values include the following:

Table 62 — PMI 2D Non-Text Polyline Type values

= 0	General line
= 1	General arrow
= 2	General circle
= 3	General arc
= 4	Extended line 1
= 5	Extended line 2
= 6	Extended arc
= 7	Extended circle
= 8	Text line (used in text boxes and symbol box dividers)

= 9	Text string

I32: Polyline Width Count

Polyline Width Count specifies the number of polyline width values.

I16: Polyline Width

Polyline Width specifies the width of polyline segment in Polyline Vertex Coords array. See Figure 127 — Constructing Non-Text Polylines from packed 2D data arrays for interpretation of this array of width values relative to the defined polylines.

VecF32: Polyline Vertex Coords

Polyline Vertex Coords is an array of polyline segments packed as 2D point coordinates. These 2D point coordinates are with respect to the 2D-Reference Frame documented in 2D-Reference Frame.

10.2.7 PMI CAD Tag Data

The PMI CAD Tag Data collection contains the list of persistent IDs, as defined in the CAD System, to uniquely identify individual PMI entities. The existence of this PMI CAD Tag Data collection is dependent upon the value of previously read data field CAD Tags Flag as documented in J.3.2 PMI Manager Meta Data Element.

If the PMI CAD Tag Data collection is present, there will be a CAD Tag for each PMI entity as specified by the below documented CAD Tag Index Count formula.

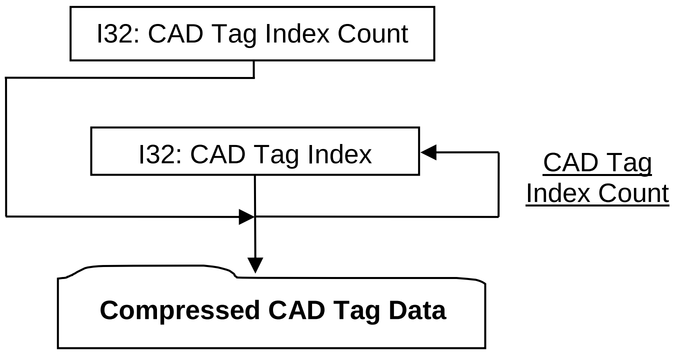


Figure 123 — PMI CAD Tag Data collection

Complete description for Compressed CAD Tag Data can be found in Compressed CAD Tag Data.

I32: CAD Tag Index Count

CAD Tag Index Count specifies the total number of CAD Tag indices. This value shall be equal to the summation of the previously read count values for all the PMI entities supporting CAD Tags. The formula is the sum of the following:

- Line Weld Count
- Spot Weld Count
- SF Count
- MP Count

- Reference Geometry Count
- Datum Target Count
- FCF Count
- Locator Count
- Dimension Count
- DFS Count
- Note Count
- Model View Count
- Design Group Count
- Coord Sys Count
- Generic Entity Count

I32: CAD Tag Index

CAD Tag Index specifies an index into a list of CAD Tags, identifying the CAD Tag belonging to a particular PMI entity. There will be a total of CAD Tag Index Count number of CAD Tag Indices and the order of the indices will be as defined by the above documented CAD Tag Index Count formula (i.e. Line Weld CAD Tag Indices are first, followed by the Spot Weld CAD Tag Indices, followed by the Surface Finish CAD Tag Indices, etc.).

10.2.8 PMI Polygon Data

The PMI Polygon Data collection contains a list of vertices classified as polygonal primitives. Its composition is shown in the figure 162. Each block of PMI PolygonData contains a list of 0 or more PolygonData elements. Empty PolygonData elements are written with 0 vertices and no additional fields.

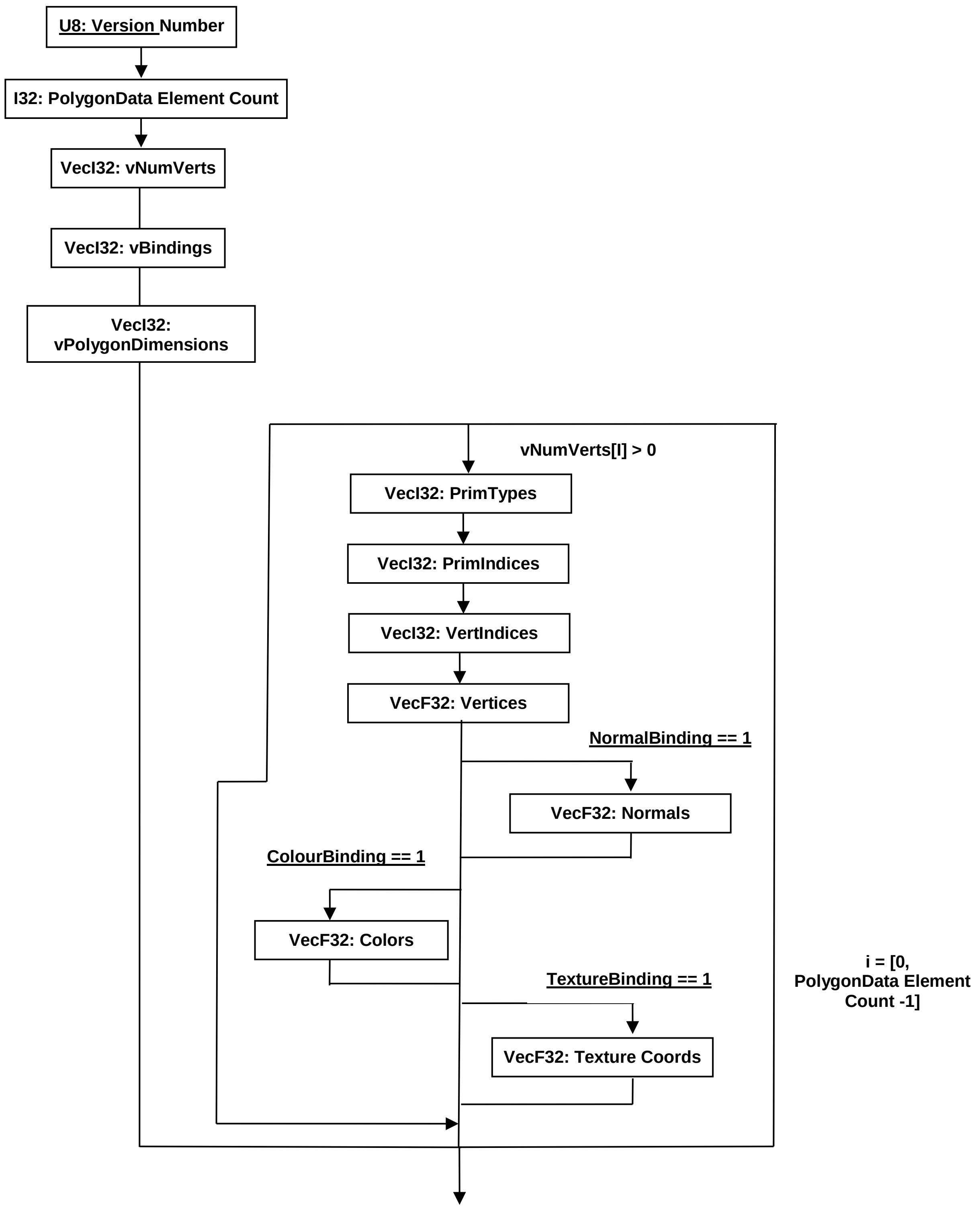


Figure 124 — PMI Polygon Data

U8: Version Number

Version number is the version identifier for this PMI Polygon Data. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

I32: PolygonData Element Count

PolygonData Element Count specifies the number of PolygonData elements.

VecI32: vNumVerts

An integer vector is used to record the number of vertices in each polygon data element. The length of this vector is equal to PolygonData Element Count written in this block of PMI PolygonData. The presence of additional data fields in each PolygonData element is hinged upon that element having more than 0 vertices recorded in this vector.

Retrieve next vertCount from vNumVerts

If the next element in the vNumVerts vector is non-zero, proceed to read other fields that make up a single PMI PolygonData element. Otherwise, skip reading more data for this element and loop back to seek the next element in the vector.

VecI32: vBindings

An integer vector used to record the bindings of all non-zero polygon data elements. This vector has three entries for each such element. The first entry is the color binding, followed by the normal binding, followed by the texture binding. These map to the non-zero polygon data elements in order.

For example:

```
If vNumVerts = {10, 0, 10, 0, 12}
vBindings = {1,0,0, 1,0,0, 1,0,0}
Then
Polygon data element 0:  {Color binding = 1; Normal binding = 0; Texture binding =
0}
Polygon data element 1:  there are no entries in this array since numVerts[1] = 0
Polygon data element 2: {Color binding = 1; Normal binding = 0; Texture binding =
0}
Polygon data element 3:  there are no entries in this array since numVerts[3] = 0
Polygon data element4: {Color binding = 1; Normal binding = 0; Texture binding = 0}
```

VecI32: vPolygonDimensions:

An integer vector used to record the dimensions of all non-zero polygon data elements. The vector has one entry for each such element. These map to the non-zero polygon data elements in order.

For example:

```
If vNumVerts = {10, 0, 10, 0, 12}
vPolygonDimensions = {3, 3, 3}
Then
Polygon data element 0:  {PolygonDimension = 3}
Polygon data element 1:  there are no entries in this array since numVerts[1] = 0
Polygon data element 2: {PolygonDimension = 3}

Polygon data element 3:  there are no entries in this array since numVerts[3] = 0
Polygon data element4: {PolygonDimension = 3}
```

iNumVerts

Number of vertices for the i^{th} PolygonData element.

I32: ColorBinding

A Boolean value that indicates if there are colors present along with the list of coordinates at each vertex.

I32: NormalBinding

A Boolean value that indicates if there are normals present along with the list of coordinates at each vertex.

I32: TextureBinding

A Boolean value that indicates if there are Texture Coordinates present along with the list of coordinates at each vertex.

I32: PolygonDimension

Indicates the dimension of vertex coordinates.

VecI32: PrimTypes

An array indicating the type of each of the primitive stored in the PrimIndices array. Adjacent numbers in the array form tuples of the form [PrimIndex, PrimType]. All primitives to the left of the PrimIndex are of type PrimType unless they are already to the left of an earlier PrimIndex in this array.

VecI32: PrimIndices

Indices of vertices that form a single primitive. The difference between two adjacent values in this array determines the length of the primitive. An extra element is stored at the end of this array to identify the length of the last primitive. Values in this array are indices into the VertIndices array.

VecI32: VertIndices

An array of indices into the Vertices array. This index array eliminates the need to duplicate floating point vertices that are shared by multiple primitives.

VecF32: Vertices

The list of vertex coordinates. Each vertex is made of PolygonDimension coordinates. The length of this list is equal to number of vertices multiplied by PolygonDimension.

VecF32: Normals

An optional list of Normals for each vertex. Presence of this list is indicated by the NormalBinding flag. Each normal consists of PolygonDimension components. The size of this list is equal to number of vertices multiplied by PolygonDimension.

VecF32: Colors

An optional list of Colours for each vertex. Presence of this list is indicated by the ColorBinding flag. Each color consists of PolygonDimension components. The size of this list is equal to number of vertices multiplied by PolygonDimension.

VecF32: Texture Coords

An optional list of Texture coordinates for each vertex. Presence of this list is indicated by the TexCoordBinding flag. Each TexCoord consists of 2 components. The size of this list is equal to number of vertices multiplied by 2.

10.2.9 PMI Properties

The PMI data segment itself can contain a list of PMI Properties to hold special semantic information. See PMI Property for the data description. There are no pre-defined properties for the PMI data segment itself.

10.2.10 PMI Model View Sort Orders

The PMI Model View Sort Orders collection defines data for a list of model view sort orders. Each model view sort order is composed of key/value pair of strings.

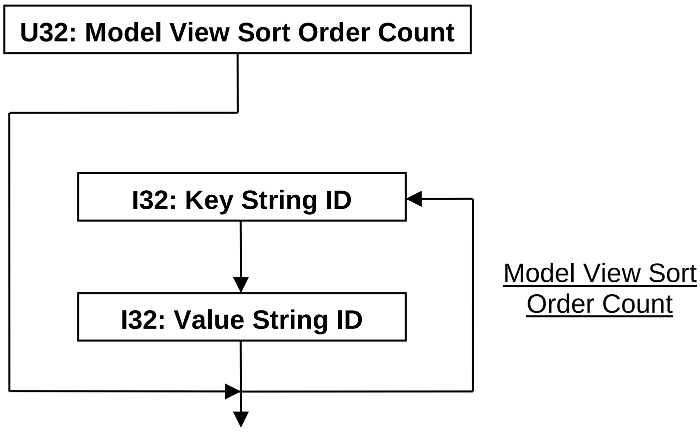


Figure 125 — PMI Model View Sort Orders data collection

U32: Model View Sort Order Count

Model View Sort Order Count specifies the number of model view sort orders.

I32: Key String ID

Key String ID specifies the string identifier for the key of model view sort order. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

I32: Value String ID

Value String ID specifies the string identifier for the value of model view sort order. This identifier is an index to a particular character string in the PMI String Table as defined in PMI String Table. An identifier value of “-1” indicates no string.

11 Info Segment

The info segment is made up of text strings that contain information on the system(s) used to author the JT file it exists in. It is intended to provide a richer list of information than what is stored in the 80 character version string in the File Header.

The information contained in the Info segment is meta-information about the provenance of the JT file itself – not information pertaining to, modifying, or supplementing the content of the JT file proper.

The Info segment is must not be used as a means of expansion to include new kinds of data in a JT file.

The Info segment must not refer to or be referred to by the JT file's content proper. The Info segment stands alone and apart from the content proper, and does not share a swizzling namespace with the remainder of the JT file..

The info segment uses the Logical Element Header Compression form of element header data.

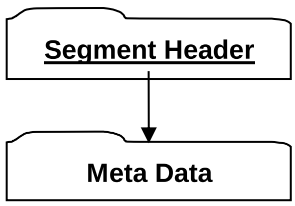


Figure 126 — Info Segment data collection

Complete description for Segment Header can be found in this document.

11.1 Info Segment Data Element

Object Type ID: 0x84c2112a, 0x0001, 0x11e7, 0x80, 0x00, 0xa4, 0x24, 0x9a, 0x27, 0x47, 0x70

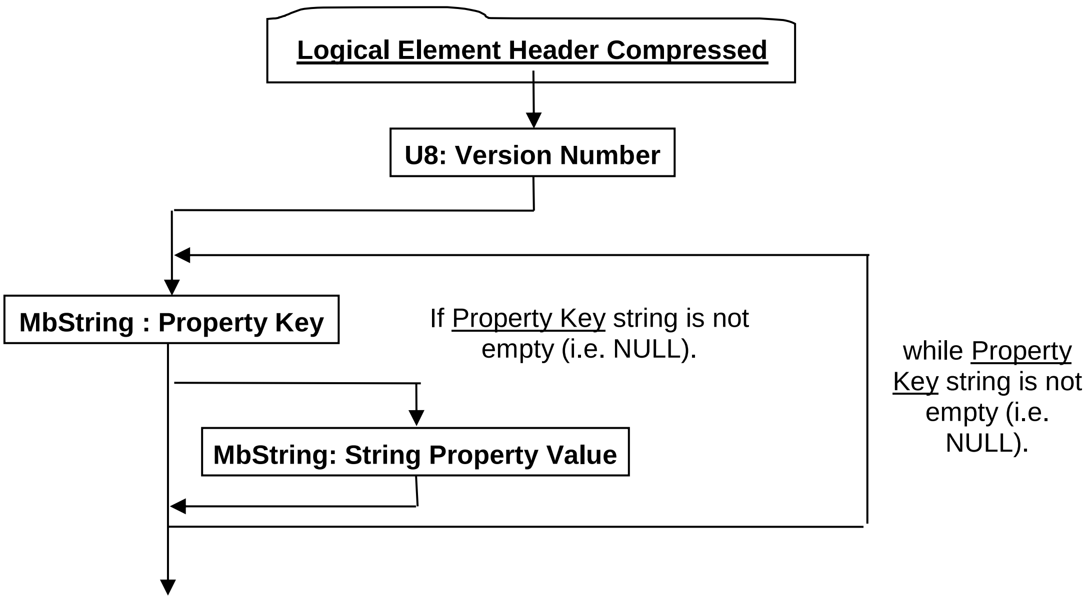


Figure 127 — Property Proxy Meta Data Element data collection

A complete description for Logical Element Header Compressed cab be found in this document.

U8: Version Number

Version Number is the version identifier for this data collection. For information on local version numbers see Common Data Conventions and Constructs Local version numbers.

MbString: Property Key

Property Key specifies the *key* string for the property.

MbString: String Property Value

String Property Value represents the property value when Property Value Type = = 1.

12 Data Compression and Encoding

The JT File format utilizes best-in-class compression and encoding algorithms to produce compact and efficient representations of data. The types of compression algorithms supported by the JT format vary from standard data type agnostic LZMA dictionary compression to entropy coding algorithms that exploit knowledge of the characteristics of the data types they are compressing. Some of the JT format data collections are always stored in a compressed format, whereas other data collections support multiple compression storage formats that qualitatively vary from “lossless” compression to more aggressive strategies that employ “lossy” compression. This support by the JT format of varying qualitative levels of compression allows producers of JT data to fine tune the trade-off between compression ratio and fidelity of the data.

In some instances, data may be encoded/compressed using multiple techniques applied on top of one another in a serial fashion (i.e. encoding applied to the output of another encoder). One common example of this multiple encoding is when an array/vector of floating point data is first quantized into some integer codes and then these resulting integer codes are further compressed/encoded using an Arithmetic or BitLength CODEC (see Encoding Algorithms).

Beyond the data collection specific compression/encoding, some JT format Data Segment types (see) also support having LZMA compression conditionally applied to all the bytes of information persisted within the segment. So individual fields or collections of data may first have data type specific encoding/compression algorithms applied to them, and then if their Data Segment type supports it, the resulting data may be additionally compressed using LZMA.

Whether, and at what qualitative level, a particular Data Segment’s data is compressed/encoded is indicated through compression related data values stored as part of the particular Data Segment storage format. In general, aggressive application of advanced compression/encoding techniques is reserved for the heavy-weight renderable geometric data (e.g. triangles and wireframe lines) which can exist in a JT File.

The following sections document the format of the data compression/encoding within the JT file. Along with documenting the format, a technical description of the various compression/encoding algorithms is included and an example implementation of the decoding portion of the algorithms can be found within Annex B.

12.1 Common Compression Data Collection Formats

For convenience and brevity in documenting the JT format, this section of the reference documents the format for several common “data compression/encoding” related data collections that can exist in the JT format. You will find references to these common compression data collections in the Data Segments section of the document.

12.1.1 Int32 Compressed Data Packet

The Int32CDP (i.e. Int32 Compressed Data Packet) represents a third-generation format used to encode/compress a collection of data into a series of Int32 based symbols. This version of the Int32CDP supersedes the two similarly-named ones from the Version 9 JT Specification, and should not be confused with either of its predecessors. Note that the Int32 Compressed Data Packet collection can in itself contain another nested Int32 Compressed Data Packet collection in some cases.

Four distinct CODECs are available for use within the Int32 Compressed Data Packet, depending on the nature of the data to be compressed.

The Arithmetic CODEC is a so-called “entropy coder” because it can exploit the statistics present in the relative frequencies of the values being encoded. Basically, the more often a value is present the fewer bits it takes to represent that value in the compressed code text. Values that occur too infrequently to take advantage of this property are written *aside* into the “out-of-band data” array to be encoded separately. An “escape” symbol is encoded in their place as a placeholder in the primal CODEC (note, see “Symbol” data field definition in Int32 Compressed Data Packet for further details on the representation of “escape” symbol).

Essentially the “out-of-band data” is the high-entropy residue left over after the CODEC has squeezed all the advantage out of the original data stream that it can. However, this “out-of-band data” is sent back around for another pass because sometimes there are *new* or *different* statistics to be exploited.

The *Chopper* pseudo-CODEC’s is used to identify fields of bits in a sequence of otherwise incompressible data that may be hiding low-entropy statistics that can be profitably exploited. In other words, it "chops" the input data up into bit fields, and then encodes them separately using the other CODECs, or in some cases, another round of chopping. The Chopper also removes *value bias* from the original input data array. Some input data arrays may contain values that are clustered around a certain central value. In these cases, it is profitable to first subtract out a *bias value* from the original input data. In some cases, this simple expedient may dramatically reduce the apparent field width necessary to code the variation in the original sequence.

In some cases, all values may be written as "out of band" when the Codec cannot perform *any* useful compression. In this case, the encoded CodeText Length field will be 0, and the I32**: Out-of-Band Value Count** will be equal to I32: Value Count. The implied action in this case is to merely copy the Out-Of-Band value data into the output Value Element array instead of invoking the Codec.

The Move-to-Front pseudo-CODEC is useful for data that exhibits spatial coherence (i.e. if a given value is likely to be used again in the near future). It decomposes the incoming data stream into two streams called “values” and “offsets”. Each time a new value is observed in the data stream, it is added to a small “cache” or “window” of the most recently seen few values (16 in this case), the value emitted to the “values” array and an “escape” emitted into the “offsets” array. When a value is seen that is already in the cache, then only its offset into the window is emitted, and the value is moved to the front of the window (hence the name). Runs and clusters are thus more efficiently represented by the values/offsets arrays. These arrays in turn are subjected to a different CODEC to finish the job – most likely the Arithmetic or Bitlength.

When all other coding options have been exhausted, the Bitlength CODEC is invoked. The Bitlength CODEC directly encodes all values given to it, does not require a probability context, and hence never produces additional “out-of-band data”. The byte stops there, in other words.

Note that in the diagram below, encoding can loop back recursively for Out-Of-Band data and chopper fields. *For JT files compliant with this specification, the maximum recursion depth may not exceed eight*.

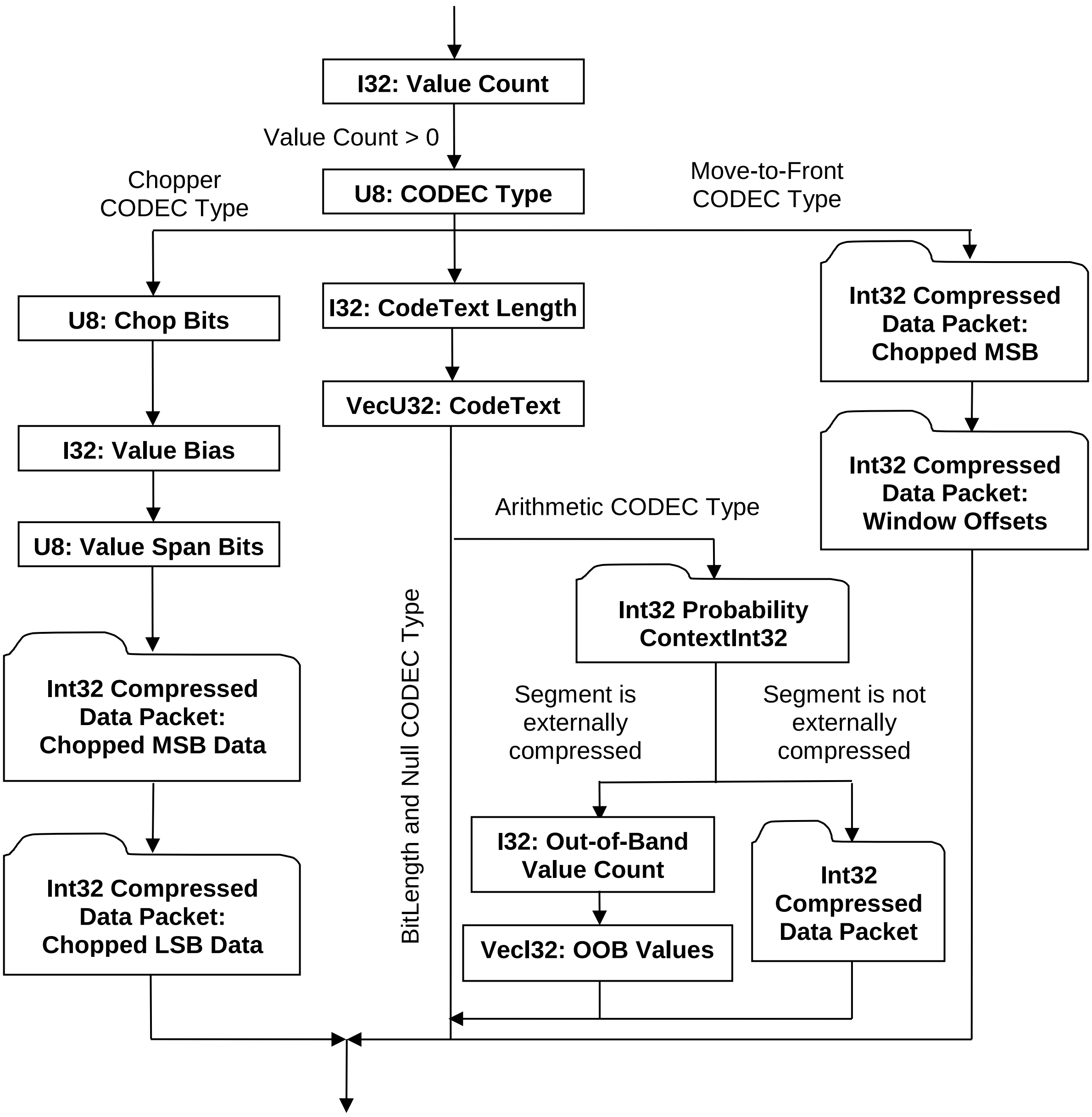


Figure 128 — Int32 Compressed Data Packet data collection

I32: Value Count

Value Count specifies the number of values that the CODEC is expected to decode (i.e. it’s like the “length” field written if you’re just writing out a vector of integers). Upon completion of decoding the CodeText data field below, the number of decoded Values should be equal to Value Count. When only a single Probability Context Table is used, Value Count will also be equal to the number of Symbols decoded upon completion of decoding.

U8: CODEC Type

CODEC Type specifies the algorithm used to encode/decode the data. See 12.2 Encoding Algorithms for complete explanation of each of the encoding algorithms.

Table 63 — Int32 Probability Contexts CODEC Type values

= 0	Null CODEC
= 1	Bitlength CODEC
= 2	Illegal value
= 3	Arithmetic CODEC
= 4	Chopper CODEC
= 5	Move-to-front CODEC

I32: CodeText Length

CodeText Length specifies the total number of bits of CodeText data.

VecU32: CodeText

CodeText is the array/vector of encoded symbols. For CODEC Type not equal to “Null CODEC”, the total number of bits of encoded data in this array is indicated by the previously described CodeText Length data field.

U8: Chop Bits

Chop Bits specifies the number of high-order bits "chopped off" from the *biased* input data array and coded separately from the low-order bits. Repeated applications of the Chopper pseudo-CODEC can expose low-entropy bit fields that would be inaccessible by directly coding the data array. Chop Bits is the number of bits coded into the Chopped MSB Data field. The number of Chop Bits is always greater than 0, and less than 32.

I32: Value Bias

Value Bias is the (signed) number that is subtracted from the original input data array elements *before* computing Value Span Bits and Chop Bits. See Chopped LSB Data below for a full explanation of how to reconstitute the original data values using Value Bias and the two chopped fields.

U8: Value Span Bits

Value Span Bits specifies the total bit width of the *biased* input data array. Note that Value Span Bits minus Chop Bits is the number of low-order bits present in the Chopped LSB Data field.

Int32 Compressed Data Packet: Chopped MSB Data

This field contains the separately compressed most significant bits of the *biased* input data array, whose elements contain Value Span Bits bits of significance. In other words, this field contains the bit field from the *biased* data array beginning at bit number ValueSpan-ChopBits and ending at bit number ValueSpan-1 inclusive. This field may contain negative numbers.

Int32 Compressed Data Packet: Chopped LSB Data

This field contains the separately compressed most significant bits of the original input data array, whose elements contain Value Span Bits bits of significance. In other words, this field contains the bit field from the original data array beginning at bit number 0 and ending at bit number ValueSpan-ChopBits-1 inclusive. This field may only contain positive numbers; all bits above this range shall encode to 0. A pseudo-code representation of the re-constituting the original data values is as follows:

```
OrigValue[i] = (LSBValue[i] | (MSBValue[i] << (ValSpanBits - ChopBits))) + ValueBias;
```

I32: Out-of-Band Value Count

This field encodes the number of out-of-band values associated with the Arithmetic CODEC.

VecI32: OOB Values

This field encodes the out-of-band Int32 values associated with the Arithmetic CODEC.

Int32 Compressed Data Packet: Window Values

Used by the move to pseudo codec, reference Move-To-Front pseudo CODEC

Int32 Compressed Data Packet: Window Offsets

Used by the move to pseudo codec, reference Move-To-Front pseudo CODEC

Int32 Probability Context

Int32 Probability Context data collection encodes a Probability Context Table, and is present only for the Arithmetic CODEC Type. A Probability Context Table is a trimmed and scaled histogram of the input values. It tallies the frequencies of the several most frequently occurring values. It is central to the operation of the Arithmetic CODEC.

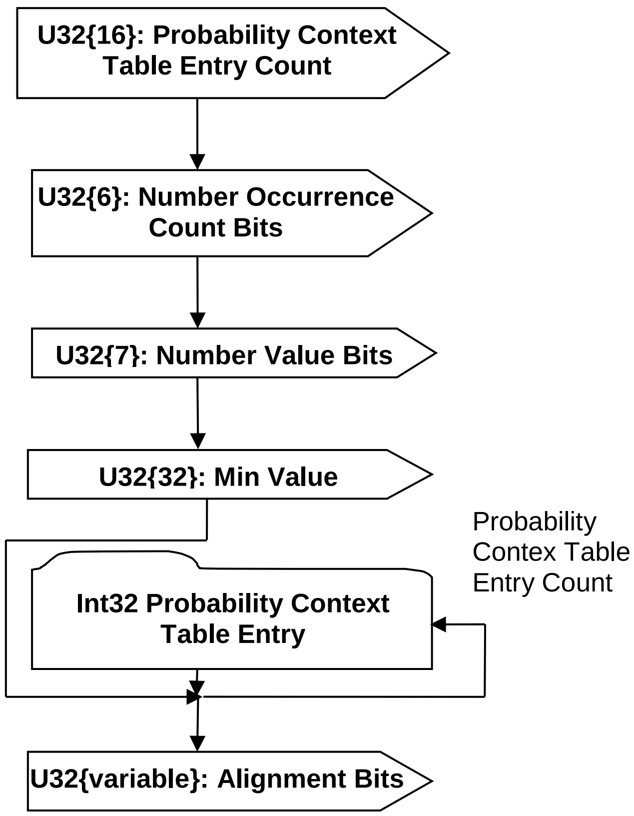


Figure 129 — Int32 Probability Context data collection

U32{16}: Probability Context Table Entry Count

Probability Context Table Entry Count specifies the number of entries in this Probability Context Table.

U32{6}: Number Occurrence Count Bits

Number Occurrence Count Bits specifies the number of bits used to encode the Occurrence Count range.

U32{7}: Number Value Bits

Number Value Bits specifies the number of bits used to encode the Associated Value range. Note that Number Value Bits is only specified in the JT file for the *first* Probability Context Table. If a second Probability Context Table is present, the Number Value Bits from the first should be used for the second as well.

U32{32}: Min Value

Min Value specifies the minimum of all Associated Values (i.e. one per table entry) stored in this Probability Context Table. This value is used to compute the real Associated Value for a Probability Context Table Entry. See Associated Value description in Int32 Probability Context Table Entry.

U32{variable}: Alignment Bits

Alignment Bits represents the number of additional padding bits stored to arrive at the next even multiple of 8 bits. Values of “0” are stored in the alignment bits.

Note: Data written into a JT file is always aligned on bytes. Therefore after reading in a block of bit data such as the probability context tables it is necessary to discard any remaining bits on the last byte that is read in. This is represented by the “Alignment Bits” entry.

Int32 Probability Context Table Entry

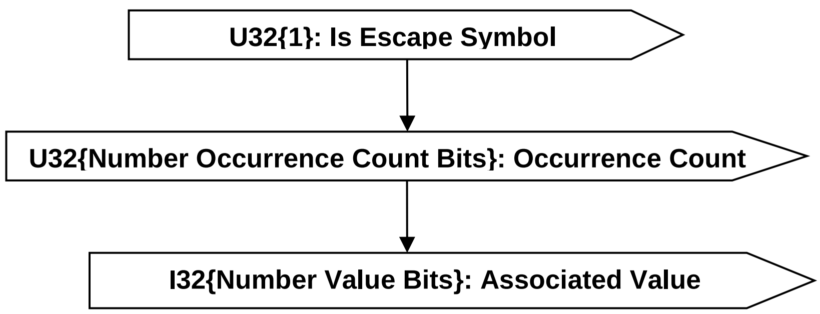


Figure 130 — Int32 Probability Context Table Entry data collection

U32{1}: Is Escape Symbol

This Boolean flag denotes whether the context entry is the escape symbol entry. At most one entry will have this flag set to true in any context.

U32{Number Occurrence Count Bits}: Occurrence Count

Occurrence Count specifies the relative frequency of the value. Complete description for Number Occurrence Count Bits can be found in Int32 Probability Context.

Note: Occurrence Counts for all symbols are normalized (converted to a relative frequency) during the write process in order to ensure the minimum amount of bits possible is used to write them while closely approximating their actual frequency.

This has several implications the reader should be aware of:

The sum of all Occurrence Counts is not guaranteed to equal the number of symbols to be decoded (see I32: Value Count for number of symbols to be decoded).

During Arithmetic decoding

.pDriver->numSymbolsToRead() – Refers to the total number of symbols to be decoded (i.e. I32: Value Count).

pCurrContext->totalCount() – Refers to the sum of the "Occurrence Count" values for all the symbols associated with a Probability Context.

I32{Number Value Bits}: Associated Value

Associated Value is the value (from the input data) that the symbol represents. The CODECs don't directly encode values, they encode symbols. Symbols, then, are associated with specific values, so when the CODEC decodes an array of symbols, you can reconstruct the array of values that was intended by looking up the symbols in the Probability Context Table. This value is stored with "Min Value" subtracted from the value. Complete descriptions for "Min Value" and Number Value Bits can be found in Int32 Probability Context.

Note: The associated value for an escape symbol is undefined and therefore can be any valid U32 number.

12.1.2 Int64 Compressed Data Packet

The Int64CDP (i.e. Int64 Compressed Data Packet) represents a format used to encode/compress a collection of data into a series of Int64 based symbols. Int64CDP shares the same encoding and compression logic as Int32CDP (i.e. Int32 Compressed Data Packet), except the data being compressed consists of an array of Int64 numbers instead of Int32 numbers.

Any scalar field (e.g. the "MinValue" field in an Int64 Probability Context) that is longer than 32 bits is written with the low-order 32 bits first in the stream, then followed by the remaining bits.

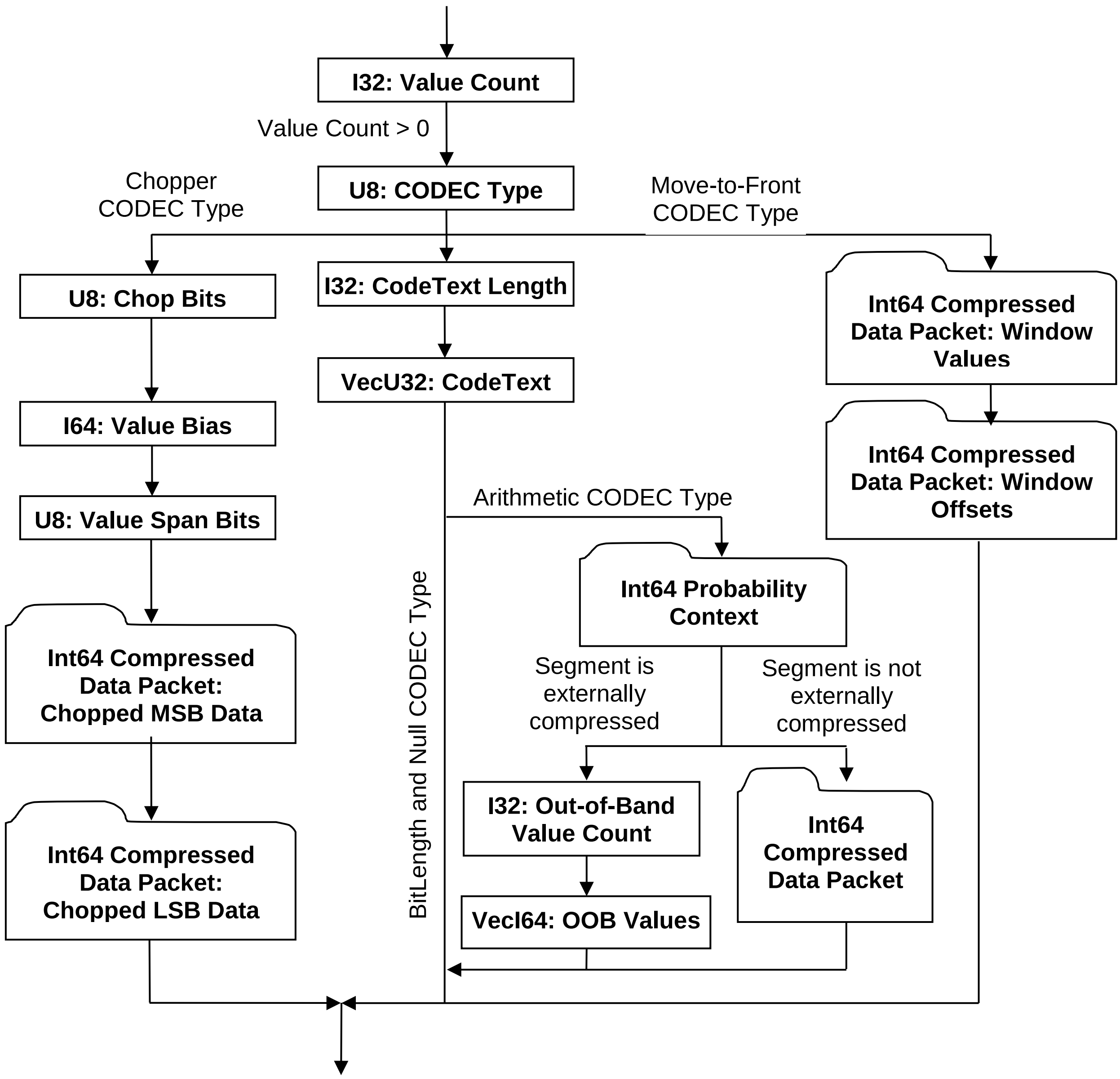


Figure 131 — Int64 Compressed Data Packet data collection

VecI64: OOB Values

This field encodes the out-of-band Int64 values associated with the Arithmetic CODEC.

I64: Value Bias

The meaning of this field is the same as I32: Value Bias except the data type is Int64 instead of Int32.

Int64 Compressed Data Packet: Chopped MSB Data

The meaning of this field is the same as Int32 Compressed Data Packet: Chopped MSB Data except the data type is Int64 instead of Int32.

Int64 Compressed Data Packet: Chopped LSB Data

The meaning of this field is the same as Int32 Compressed Data Packet: Chopped LSB Data except the data type is Int64 instead of Int32.

Int64 Compressed Data Packet: Window Values

The meaning of this field is the same as Int32 Compressed Data Packet: Chopped MSB Data except the data type is Int64 instead of Int32.

Int64 Compressed Data Packet: Window Offsets

The meaning of this field is the same as Int32 Compressed Data Packet: Window Offsets except the data type is Int64 instead of Int32.

Int64 Probability Context

Int64 Probability Context data collection encodes a Probability Context Table, and is present only for the Arithmetic CODEC Type. Int64 Probability Context is the same as Int32 Probability Context, except the data element is of type Int64 instead of Int32.

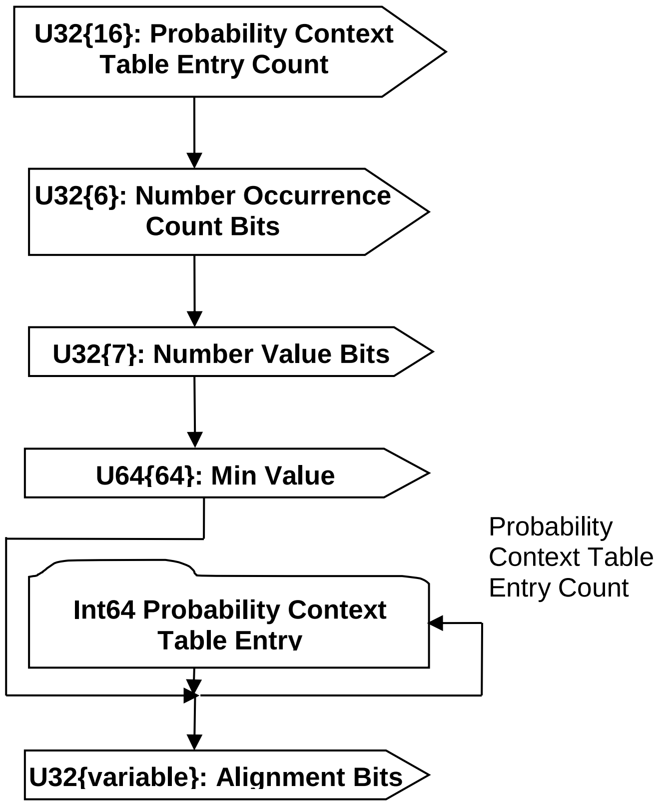


Figure 132 — Int64 Probability Context data collection

U64{64}: Min Value

Min Value specifies the minimum of all Associated Values (i.e. one per table entry) stored in this Probability Context Table. This value is used to compute the real Associated Value for a Probability Context Table Entry. See Associated Value description in Int64 Probability Context Table Entry.

Int64 Probability Context Table Entry

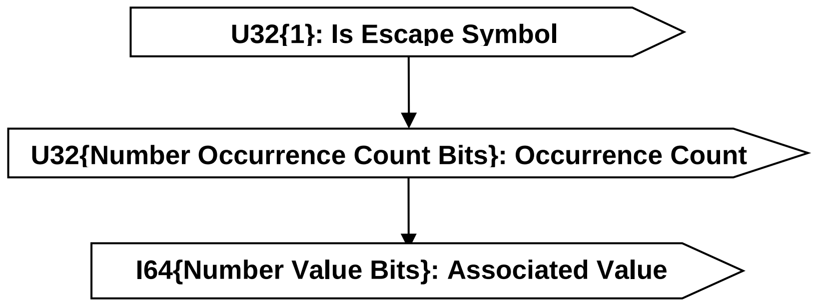


Figure 133 — Int64 Probability Context Table Entry data collection

I64{Number Value Bits}: Associated Value

Similar to I32{Number Value Bits}: Associated Value, I64{Number Value Bits}: Associated Value is the value (from the input data) that the symbol represents. This value is stored with “Min Value” subtracted from the value. Complete descriptions for “Min Value” and Number Value Bits can be found in Int64 Probability Context.

12.1.3 Compressed Vertex Coordinate Array

The Compressed Vertex Coordinate Array data collection contains the quantization data/representation for a set of vertex coordinates.

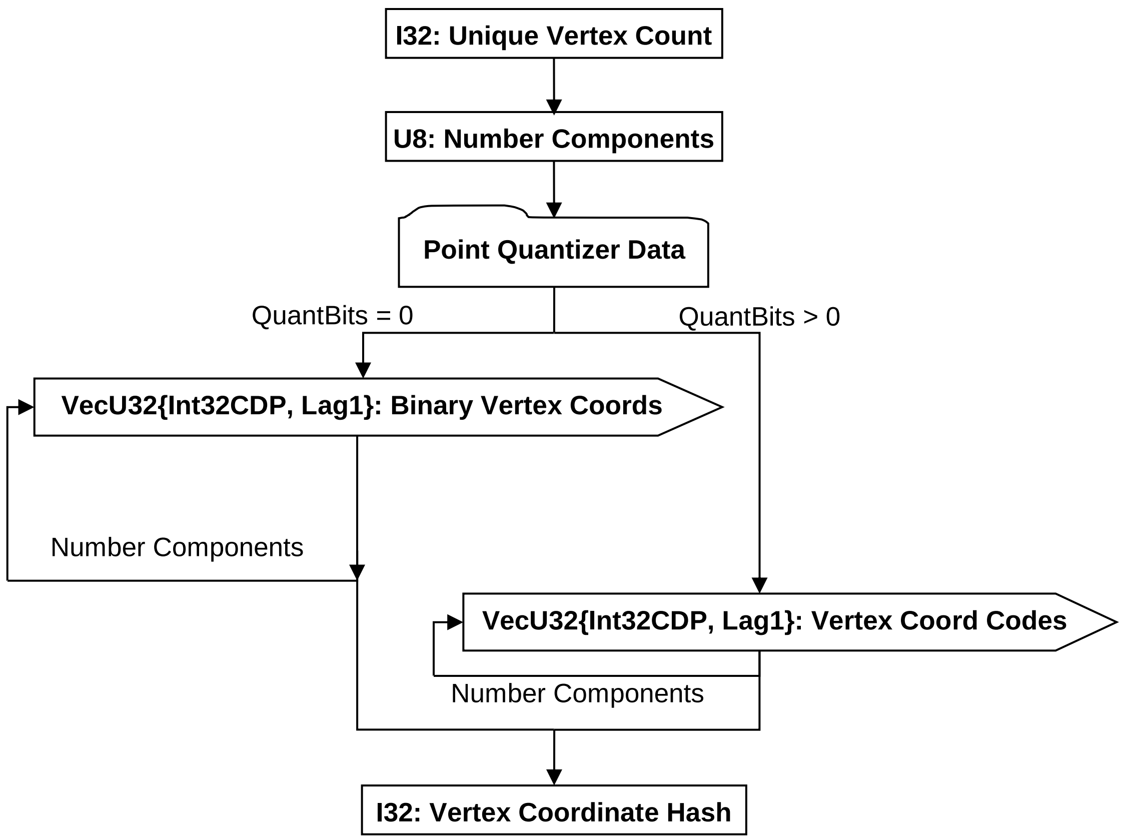


Figure 134 — Compressed Vertex Coordinate Array data collection

Complete description for Point Quantizer Data can be found in Point Quantizer Data.

The above predicates “QuantBits = 0” and “QuantBits > 0” refer to the value of the field U8: Number Of Bits stored in the three components of Point Quantizer Data. All three of these fields are required to be equal.

I32: Unique Vertex Count

Vertex Count specifies the count (number of unique) vertices in the Vertex Codes arrays. Identical values are only stored once therefore it may be necessary to smear out the vertices as described in TopoMesh Compressed Rep Data and TopoMesh Topologically Compressed LOD Data.

U8: Number Components

Number Components specifies the number of vertex components present for each vertex record in the set of vertex records. The only legal value for this field is 3.

VecU32{Int32CDP, Lag1}: Binary Vertex Coords

Binary Vertex Coords is a vector of the ith component values of a set of vertex coordinates *interpreted* as integers. That is to say, the binary IEEE-754 floating point representation of the coordinates is fed *directly* into the Lag1 predictor as if they were integers.

VecU32{Int32CDP, Lag1}: Vertex Coord Codes

Vertex Coord Codes is a vector of quantizer “codes” for all the ith component values of a set of vertex coordinates. Vertex Coord Codes uses the Int32 version of the CODEC to compress and encode data.

I32: Vertex Coordinate Hash

The Vertex Coordinate Hash is the combined hash of the unique vertex coordinate records. If the number of quantization bits is equal to zero the hash value is equal to the combined hash of the vertex coordinate values for each of the component arrays. If the number of quantization bits is greater than 0 the hash value is equal to the combined hash of the vertex coordinates codes for each of the component arrays. Refer to the Hashing Annex for a more detailed description on hashing.

```
UInt32 uHash    = 0;
uInt32 nUniqVtx = 0;
vecF32 vCoord[nUniqVtx][3];
vecU32 vCodes[3];
...
if ( uQuantBits == 0 ) {
  for ( int i=0 ; i<nComp ; i++ ) {
    for ( int j=0 ; j<nUniqVtx ; j++) {
      uHash = hash32( (const UInt32*)(&vCoord[j][i]), 1, uHash );
    }
  }
} else {
  for ( int i=0 ; i<nComp ; i++ ) {
   uHash = hash32( &vCodes[i], nUniqVtx, uHash );
  }
}
```

12.1.4 Compressed Vertex Normal Array

The Compressed Vertex Normal Array data collection contains the compressed data/representation for a set of vertex normals. Compressed Vertex Normal Array data collection is only present if previously read vertex bindings denote normals are present (see Vertex Shape LOD Data U64 : Vertex Bindings for complete explanation of the vertex bindings).

A variation of the CODEC developed by Michael Deering at Sun Microsystems is used to encode the normals when quantization is enabled. The variation being that the “Sextants” are arranged differently than in Deering’s scheme [4], for better delta encoding. See Deering Normal CODEC for a complete explanation on the Deering CODEC used.

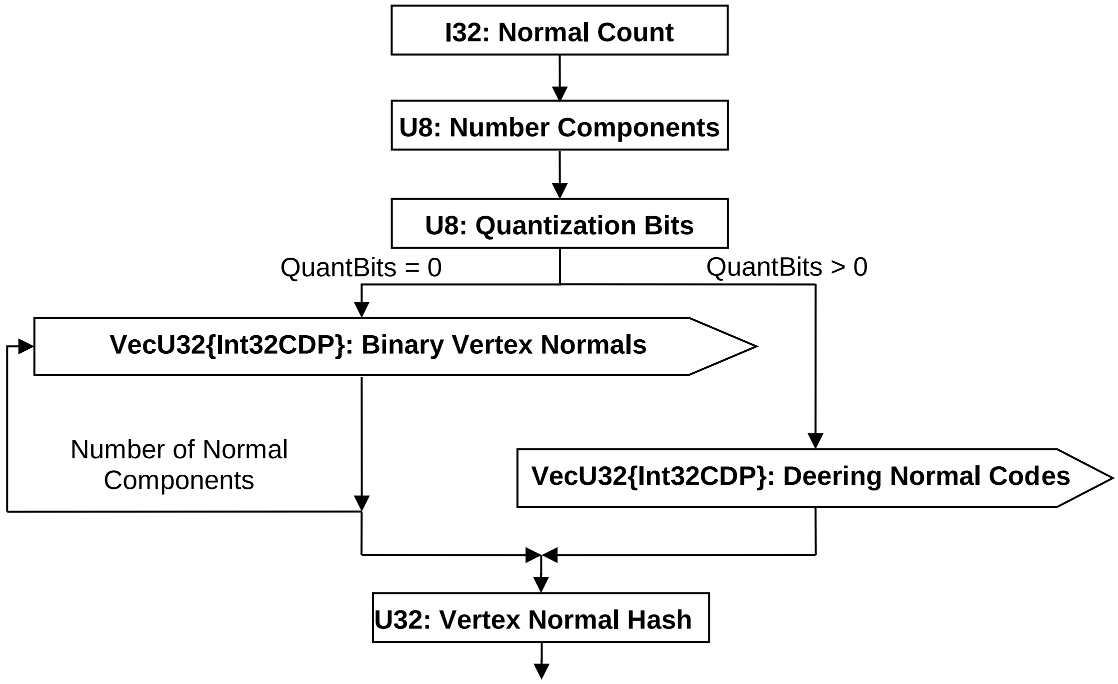


Figure 135 — Compressed Vertex Normal Array data collection

I32: Normal Count

Normal count specifies the number of normals. This number should equal the total number of vertex records.

U8: Number Components

Number Components specifies the number of normal components present for each vertex record in the set of vertex records.

U8: Quantization Bits

The number of bits used when the Deering Normal CODEC if quantization is enabled. A value of 0 denotes that quantization is disabled. The maximum value for this field is 13 (so that the resulting Deering normal codes are of at most 32 bits).

VecU32{Int32CDP}: Binary Vertex Normals

Binary Vertex Normals is a vector of the ith component values of a set of vertex normals *interpreted* as integers. That is to say, the binary IEEE-754 floating point representation of the coordinates is fed *directly* into the Lag1 predictor as if they were integers.

VecU32{Int32CDP}: Deering Normal Codes

Deering Normal Codes is a vector of “codes” (one per normal) for a set of normals produced by the Deering Normal Codec (q.v.). Deering Normal Codes uses the Int32 version of the CODEC to compress and encode data.

U32: Vertex Normal Hash

The Vertex Normal Hash is the combined hash of the vertex normals. If the number of quantization bits is equal to zero the hash value is equal to the combined hash of the vertex normal values for each of the component arrays. If the number of quantization bits is greater than 0 the hash value is equal to the combined hash of the Sextant, Octant, Theta, and Psi Codes for all vertex records.

```
UInt32 uHash   = 0;
uInt32 nVtxRec = 0;
vecF32 vNorm[nVtxRec][3];
vecU32 vDeeringCodes;
...
if ( uQuantBits == 0 ) {
  for ( int i=0 ; i<nComp ; i++ ) {
    for ( int j=0 ; j<nVtxRec ; j++) {
      uHash = hash32( (UInt32*)(&vNorm[j][i]), 1, uHash );
    }
  }
} else {
   uHash = hash32( &vDeeringCodes, nVtxRec, uHash );
}
```

12.1.5 Compressed Vertex Texture Coordinate Array

The Compressed Vertex Texture Coordinate Array data collection contains the quantization data/representation for a set of vertex texture coordinates. Compressed Vertex Texture Coordinate Array data collection is only present if previously read vertex bindings denote texture coordinates are presents (See Vertex Shape LOD Data U64 : Vertex Bindings for complete explanation of the vertex bindings).

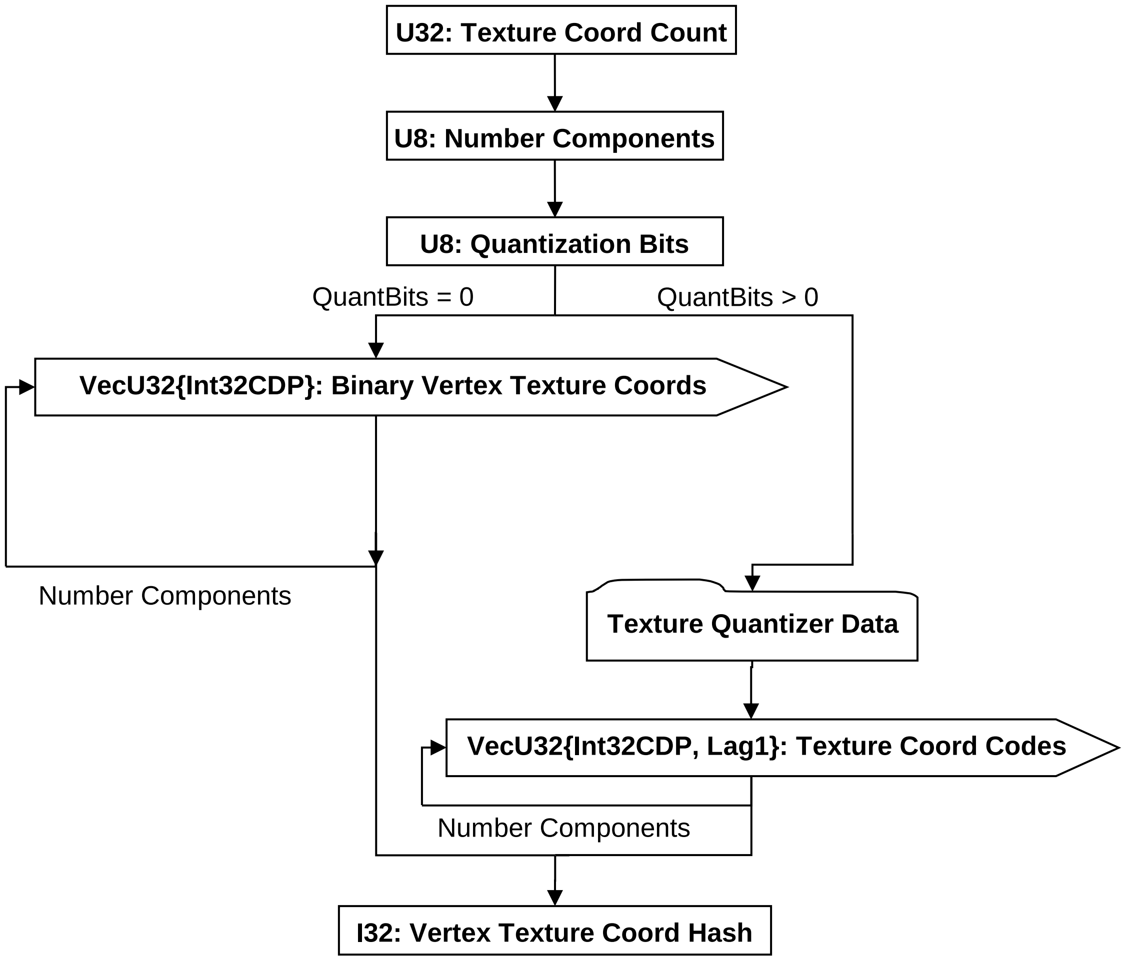


Figure 136 — Compressed Vertex Texture Coordinate Array data collection

Complete description for Texture Quantizer Data can be found in Texture Quantizer Data.

U32: Texture Coord Count

Coord Count specifies the number of Texture Coordinates. This number should equal the total number of vertex records.

U8: Number Components

Number Components specifies the number of Texture Coordinate components present for each vertex record in the set of vertex records.

U8: Quantization Bits

Number of Bits specifies the quantized size (i.e. the number of bits of precision) for each of the components. The actual number of quantization bits used is specified within Texture Quantizer Data. Value shall be within range [0:24] inclusive.

VecU32{Int32CDP}: Binary Vertex Texture Coords

Binary Vertex Texture Coordinates is a vector of the ith component values of a set of texture coordinates *interpreted* as integers. That is to say, the binary IEEE-754 floating point representation of the coordinates is fed *directly* into the Lag1 predictor as if they were integers.

VecU32{Int32CDP, Lag1}: Texture Coord Codes

Texture Coord Codes is a vector of quantizer “codes” for all the nth-component of a set of vertex texture coordinates. Texture Coord Codes uses the Int32 version of the CODEC to compress and encode data.

I32: Vertex Texture Coord Hash

The Vertex Texture Coord Hash is the combined hash of the Vertex Texture Coordinates. If the number of quantization bits is equal to zero the hash value is equal to the combined hash of the vertex texture coordinate values for each of the component arrays. If the number of quantization bits is greater than 0 the hash value is equal to the combined hash of the vertex texture coordinates codes for each of the component arrays.

```
UInt32 uHash    = 0;
uInt32 nVtxRec = 0;
vecF32 vTexCoord[nVtxRec][4];
vecU32 vCodes[4];
...
if ( uQuantBits == 0 ) {
  for ( int i=0 ; i<nComp ; i++ ) {
    for ( int j=0 ; j<nVtxRec ; j++) {
      uHash = hash32( (UInt32*)(&vTexCoord[j][i]), 1, uHash );
    }
  }
} else {
  for ( int i=0 ; i<nComp ; i++ ) {
   uHash = hash32( &vCodes[i], nVtxRec, uHash );
  }
}
```

12.1.6 Compressed Vertex Colour Array

The Compressed Vertex Colour Array data collection contains the quantization data/representation for a set of vertex colours. Compressed Vertex Colour Array data collection is only present if previously read Colour Binding value is not equal to zero (See Vertex Shape LOD Data for complete explanation of Colour Binding data field).

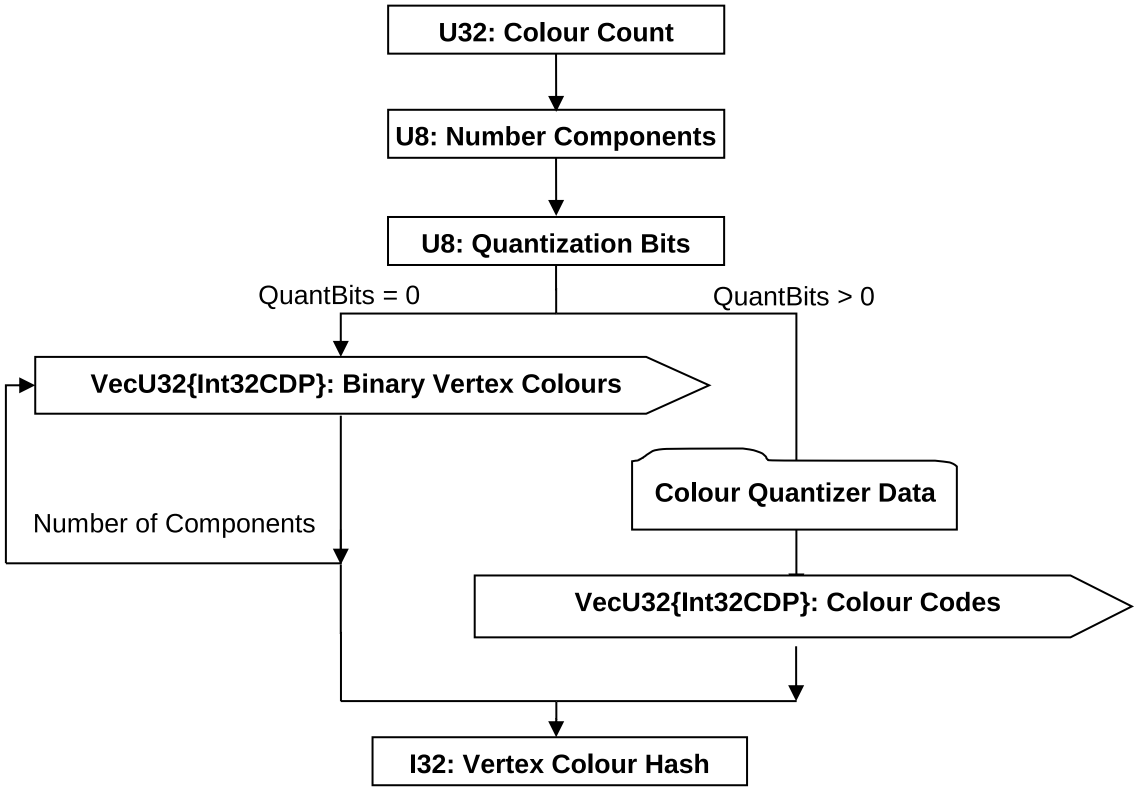


Figure 137 — Compressed Vertex Colour Array data collection

Complete description for Colour Quantizer Data can be found in Colour Quantizer Data.

U32: Colour Count

Colour count specifies the number of colour records. This number should equal the total number of vertex records.

U8: Number Components

Number Components specifies the number of Colour components present for each vertex record in the set of vertex records.

U8: Quantization Bits

Number of Bits specifies the quantized size (i.e. the number of bits of precision) for each of the 3 or 4 colour components. This value shall satisfy the following condition: “0 <= Number Of Bits <= 8”.

VecU32{Int32CDP}: Binary Vertex Colours

Binary Vertex Colours is a vector of the ith component values of a set of texture coordinates *interpreted* as integers. That is to say, the binary IEEE-754 floating point representation of the coordinates is fed *directly* into the Lag1 predictor as if they were integers.

VecU32{Int32CDP}: Colour Codes

Colour Codes is a vector of quantizer “codes” for all the vertex colours. Each Colour Code contains up to four bit fields representing the RGBA or HSVA encoded colour. The width of each field is set by the corresponding data in Colour Quantizer Data. The alpha field lies in the least significant bits, the B/V field lies immediately to the left of the alpha field, the G/S field lies immediately to the left of B/V field, and so on toward the more significant bits.

I32: Vertex Colour Hash

The Vertex Colour Hash is the combined hash of the vertex colours. If the number of quantization bits is equal to zero the hash value is equal to the combined hash of the vertex colour values for each of the component arrays. If the number of quantization bits is greater than 0 the hash value is equal to the hash of the Colour Codes vector.

```
UInt32 uHash    = 0;
uInt32 nVtxRec = 0;
vecF32 vCol[nVtxRec][3];
vecU32 vColorCodes;
...
if ( uQuantBits == 0 ) {
  for ( int i=0 ; i<nComp ; i++ ) {
    for ( int j=0 ; j<nVtxRec ; j++) {
      uHash = hash32( (UInt32*)(&vCol[j][i]), nVtxRec, uHash );
    }
  }
} else {
   uHash = hash32( &vColorCodes, nVtxRec, uHash );
}
```

12.1.7 Compressed Vertex Flag Array

The Compressed Vertex Flag Array data collection contains the quantization data/representation for per vertex flags. Compressed Vertex Flag Array data collection is only present if previously read Vertex Flag Binding value is not equal to zero.

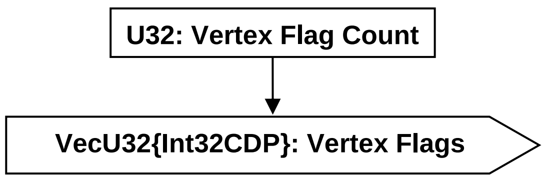


Figure 138 — Compressed Vertex Flag Array data collection

U32: Vertex Flag Count

Vertex flag count specifies the number of vertex flags. This number should be equal to the total number of vertex records.

VecU32{Int32CDP}: Vertex Flags

Vertex Flags is a vector of per vertex bit flags encoded as integers with valid values of either 0 (false) or 1 (true). Vertex Flags uses the Int32 version of the CODEC to compress and encode data.

12.1.8 Compressed Auxiliary Fields Array

Compressed Auxiliary Fields Array data contains additional geometric shape data (auxiliary vertex fields) that may be associated with each vertex record defined in TopoMesh Compressed LOD Data. Each Auxiliary field contains data that is parallel to the existing vertex record fields in order capture additional information about each vertex (e.g. Vertex Identifiers, Weights, or other information). Importantly, each datum in the Auxiliary field may have a single value (of the specified type), or *may have many values* (called „steps“). Again, each data collection may have multiple Auxiliary fields, each of which contains one datum per vertex record in the TopoMesh Compressed LOD Data, each datum containing 1 or more values (steps).

Each Auxiliary field has a GUID tag (unique to the fields in the current TopoMesh Compressed LOD Data record), and a field data type that allows the user to store a variety of different data types. It is not intended that Auxiliary fields be required to directly participate in rendering – that is the province of the vertex attributes defined in TopoMesh Compressed LOD Data.

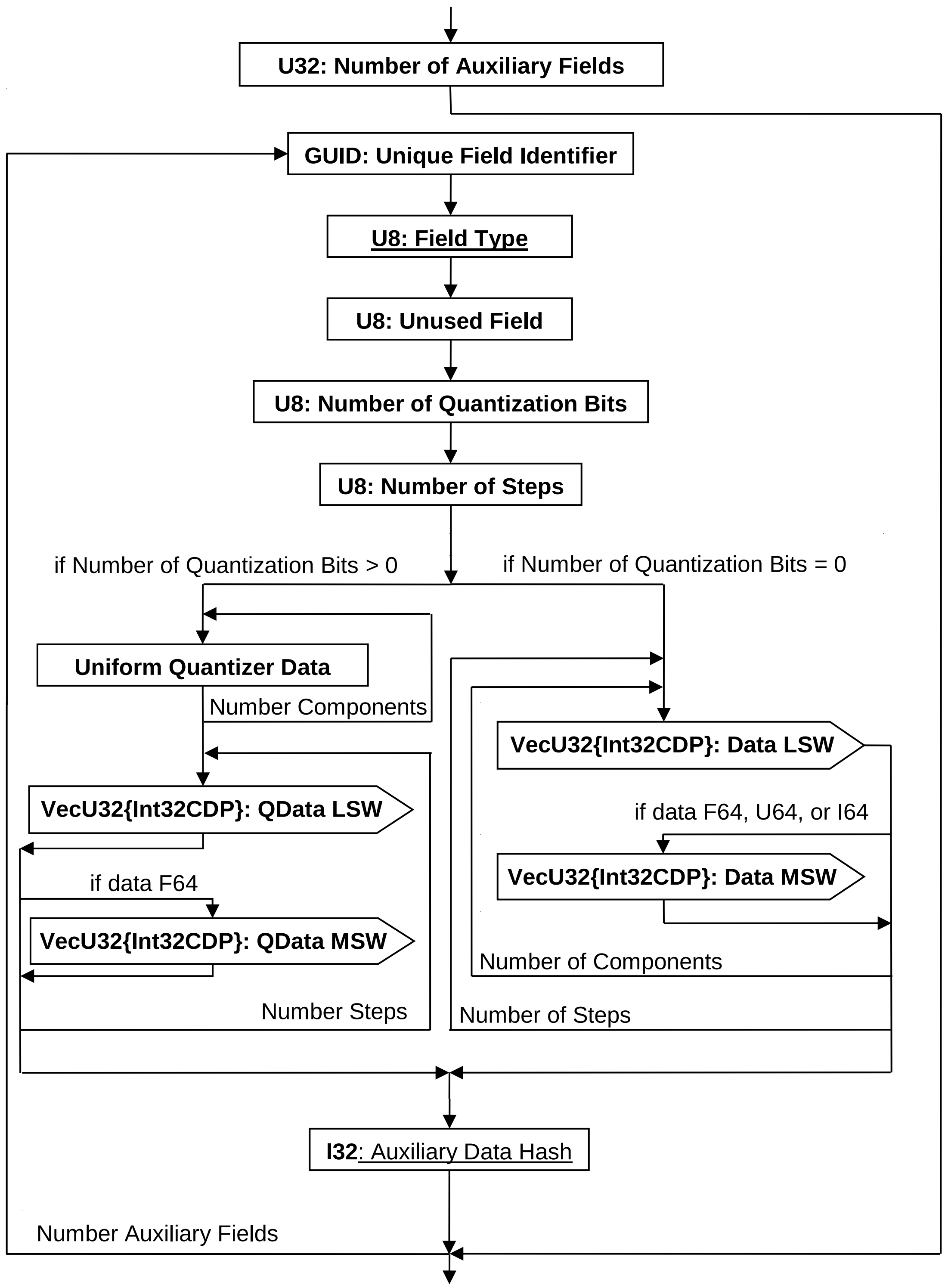


Figure 139 — Compressed Auxiliary Fields Array data collection

U32: Number of Auxiliary Fields

The number of auxiliary vertex fields included in the data collection.

GUID: Unique Field Identifier

Each Auxiliary Vertex Field is associated with Unique Field Identifier to denote the usage of the contained data. This field is intended to be user defined. Any valid GUID, as defined in Table 4, can be used as a Unique Field Identifier.

U8: Field Type

Defines the number of components and type of data contained within the auxiliary field based upon the below table.

Table 64 — TopoMesh Compressed Rep Data V2 Field Type values

Type	Data	Components		Type	Data	Components
1	U8	1		24	I32	4
2	U8	2		25	U64	1
3	U8	3		26	U64	2
4	U8	4		27	U64	3
5	I8	1		28	U64	4
6	I8	2		29	I64	1
7	I8	3		30	I64	2
8	I8	4		31	I64	3
9	U16	1		32	I64	4
10	U16	2		33	F32	1
11	U16	3		34	F32	2
12	U16	4		35	F32	3
13	I16	1		36	F32	4
14	I16	2		37	F32	2x2
15	I16	3		38	F32	3x3
16	I16	4		39	F32	4x4
17	U32	1		40	F64	1
18	U32	2		41	F64	2
19	U32	3		42	F64	3
20	U32	4		43	F64	4
21	I32	1		44	F64	2x2
22	I32	2		45	F64	3x3
23	I32	3		46	F64	4x4

U8: Unused Field

This field is unused.

U8: Number of Quantization Bits

Each Auxiliary field can be lossily or losslessly compressed. A value of 0 in this field means that that field data are to be losslessly compressed. This field must be 0 for all integer field types (i.e. lossy compression is not defined for integer field types). Only floating-point field types may be lossily compressed. A value between 1 and 32, or between 1 and 64 for double precision floating point typed fields, indicates that the data are quantized to the indicated number of bits of significance.

U8: Number of Steps

This field represents the number of “steps” present with each vertex record for the given Auxiliary field. The field must be a number greater than 0. All vertex records have the same number of “steps.” One or more steps, called “suppressed steps”, may contain no auxiliary data. At least one step must be not suppressed for any auxiliary field.

VecU32{Int32CDP}: Data LSW

Data LSW is an array of the low order 32 bits of the vector formed from all steps, components, and vertex records of a single Auxiliary Data field. The data is laid out by cycling through vertex records first, then components, and steps last so that adjacent vertex records for the same step and component are contiguous (i.e. in "stepwise-major, component semi-major" order). For U8, I8, U16, I16, U32, I32 and lossless F32 data types this contains all bits. For U64, I64, and lossless F64 data types it contains bits 0 through 31. Data LSW uses the Int32 version of the CODEC to compress and encode data. In the case when the data is a zero length vector, it means that no auxiliary data exists on any vertex record for this component in this step. In addition, if no auxiliary data exists for one component in a particular step then it is guaranteed that no auxiliary data exists for all the other components in the same step.

VecU32{Int32CDP}: Data MSW

Data MSW is an array of the low order 32 bits of the vector formed from all steps, components, and vertex records of a single Auxiliary Data field. The data is laid out by cycling through vertex records first, then components, and steps last so that adjacent vertex records for the same step and component are contiguous (i.e. in "stepwise-major, component semi-major" order). For U64, I64, and lossless F64 data types it contains bits 32 through 63. Data MSW uses the Int32 version of the CODEC to compress and encode data. In the case when the data is a zero length vector, it means that no auxiliary data exists on any vertex record for this component in this step. In addition, if no auxiliary data exists for one component in a particular step then it is guaranteed that no auxiliary data exists for all the other components in the same step.

VecU32{Int32CDP}: QData LSW

QData LSW is an array of the low order 32 bits of the vector formed from all components and vertex records of a single Auxiliary Data field. The data is laid out by cycling through vertex records first, then components so that adjacent vertex records for the same step are contiguous (i.e. in "stepwise-major" order) For the F32 data type this field contains all bits. For F64 the data type it contains bits 0 through 31. QData LSW uses the Int32 version of the CODEC to compress and encode data. Note that there is one QData LSW packet for each step in the Auxfield rather than a single unified packet as with Data LSW. In the case when the data is a zero length vector, it means that no auxiliary data exists on any vertex record for this component in this step. In addition, if no auxiliary data exists for one component in a particular step then it is guaranteed that no auxiliary data exists for all the other components in the same step.

VecU32{Int32CDP}: QData MSW

QData MSW is an array of the low order 32 bits of the vector formed from all steps, components, and vertex records of a single Auxiliary Data field. The data is laid out by cycling through vertex records first, then components, and steps last so that adjacent vertex records for the same step and component are contiguous (i.e. in "stepwise-major, component semi-major" order). For U64, I64, and lossless F64 data types it contains bits 32 through 63. QData MSW uses the Int32 version of the CODEC to compress and encode data. Note that there is one QData MSW packet for each step in the Auxfield rather than a single unified packet as with Data MSW. In the case when the data is a zero length vector, it means that no auxiliary data exists on any vertex record for this component in this step. In addition, if no auxiliary data exists for one component in a particular step then it is guaranteed that no auxiliary data exists for all the other components in the same step.

I32: Auxiliary Data Hash

The Auxiliary Data Hash is the combined hash of auxiliary field data arrays.

```
UInt32 uHash      = 0;
UInt32 nVtxRec    = 0,  // Number of vertex records
       nComp      = 0,  // Number of components in current Auxfield
       nSteps     = 0;  // Number of steps in current Auxfield
vecU32 vDataLSW [nSteps][nComp],
```

```
       vDataMSW [nSteps][nComp],
       vQDataLSW[nSteps],
       vQDataMSW[nSteps];
...
if (nQuantBits == 0) {
  if ( bU8 || bI8 || bU16 || bI16 || bU32 || bI32 || bF32) {
    for ( int i=0 ; i<nSteps ; i++ )
      for ( int j=0 ; j<nComp ; j++ )
        uHash = hash32( vDataLSW[i][j], nVtxRec, uHash );
  }
  else {   // bU64 || bI64
    for ( int i=0 ; i<nSteps ; i++ ) {
      for ( int j=0 ; j<nComp ; j++ ) {
        uHash = hash32( vDataLSW[i][j], nVtxRec, uHash );
        uHash = hash32( vDataMSW[i][j], nVtxRec, uHash );
      }
    }
  }
} else {
  if (bF32) {
    for ( int i=0 ; i<nSteps ; i++ )
      uHash = hash32( vQDataLSW[i], nVtxRec * nComp, uHash );
  }
  else {   // bF64
    for ( int i=0 ; i<nSteps ; i++ ) {
      uHash = hash32( vQDataLSW[i], nVtxRec * nComp, uHash );
      uHash = hash32( vQDataMSW[i], nVtxRec * nComp, uHash );
    }
  }
}
```

12.1.9 Point Quantizer Data

A Point Quantizer Data collection is made up of three Uniform Quantizer Data collections; there is a separate Uniform Quantizer Data collection for the X, Y, and Z values of point coordinates.

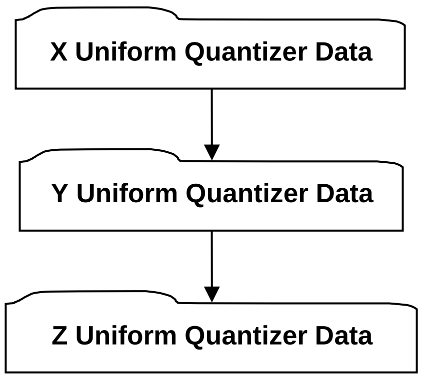


Figure 140 — Point Quantizer Data collection

Complete description for X Uniform Quantizer Data, Y Uniform Quantizer Data and Z Uniform Quantizer Data can be found in Uniform Quantizer Data.

12.1.10 Texture Quantizer Data

A Texture Quantizer Data collection is made up of n Uniform Quantizer Data collections; there is a separate Uniform Quantizer Data collection for each component of the texture coordinates. The number of components is not specified within the quantizer, but rather is determined by the number of texture components present in the underlying data (See Compressed Vertex Texture Coordinate Arrays U8: Number Components).

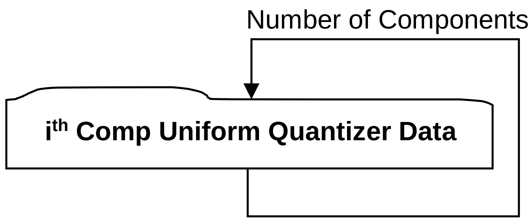


Figure 141 — Texture Quantizer Data collection

Complete description for U Uniform Quantizer Data and V Uniform Quantizer Data can be found in Uniform Quantizer Data.

12.1.11 Colour Quantizer Data

A Colour Quantizer Data collection contains the quantizer information for each of the colour components. The Colour Quantizer utilizes a separate Uniform Quantizer Data collection for each of the 4 colour components, but if the HSV colour model is being used, then it is not necessary to store a complete Uniform Quantizer Data Collection.

For the HSV model, since the range values for each colour component are constant, only the Number of Bits of precision for each colour component's Uniform Quantizer is stored. The Uniform Quantizer range values for the HSV colour components should always be assumed to be the following:

Table 65 — Colour Quantizer values

Component	Quantizer Range	
	Min	Max
Hue	0.0	6.0
Saturation	0.0	1.0
Value	0.0	1.0
Alpha	0.0	1.0

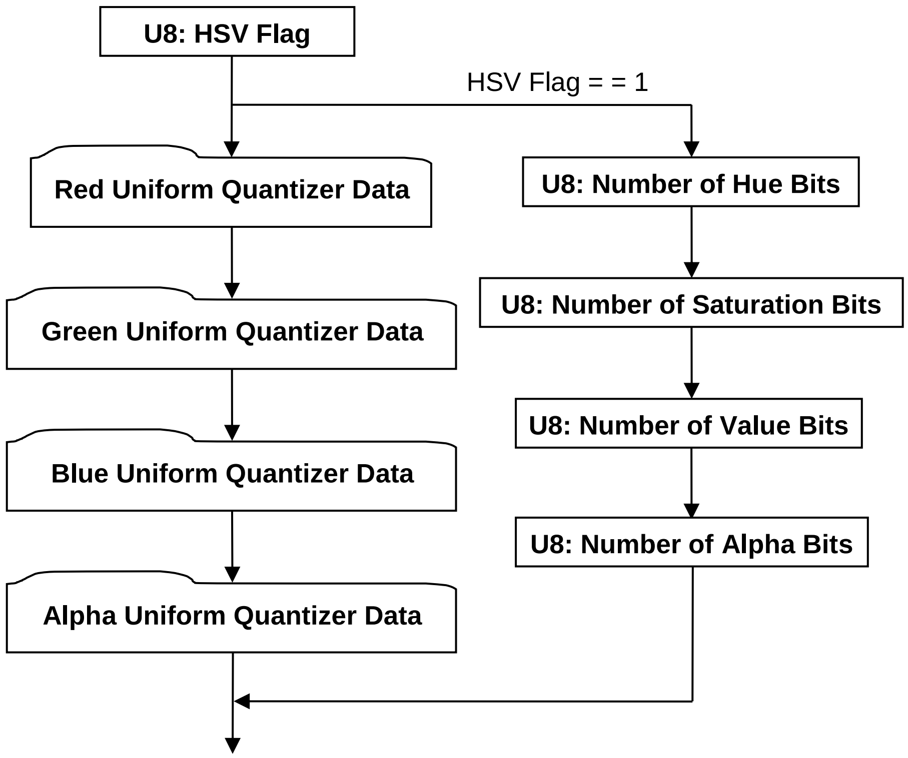


Figure 142 — Colour Quantizer Data collection

Complete descriptions for Red Uniform Quantizer Data, Green Uniform Quantizer Data, Blue Uniform Quantizer Data and Alpha Uniform Quantizer Data can be found in Uniform Quantizer Data. These four Uniform Quantizer Data collections are only present when data field HSV Flag = = 0.

U8: HSV Flag

HSV Flag is a flag indicating whether colour component data is stored in HSV colour model form.

Table 66 — Colour Quantizer HSV Flag values

= 0	Colour component data stored in RGB colour model form.
= 1	Colour component data stored in HSV colour model form.

U8: Number of Hue Bits

Number of Hue Bits specifies the quantized size (i.e. the number of bits of precision) for the Hue component of the colour. Number of Hue Bits data is only present when data field HSV Flag = = 1.

U8: Number of Saturation Bits

Number of Saturation Bits specifies the quantized size (i.e. the number of bits of precision) for the Saturation component of the colour. Number of Saturation Bits data is only present when data field HSV Flag = = 1.

U8: Number of Value Bits

Number of Value Bits specifies the quantized size (i.e. the number of bits of precision) for the Value component of the colour. Number of Value Bits data is only present when data field HSV Flag = = 1.

U8: Number of Alpha Bits

Number of Alpha Bits specifies the quantized size (i.e. the number of bits of precision) for the Alpha component of the colour. Number of Alpha Bits data is only present when data field HSV Flag = = 1.

12.1.12 Uniform Quantizer Data

The Uniform Quantizer Data collection contains information that defines a scalar quantizer/dequantizer (encoder/decoder) whose range is divided into levels of equal spacing.

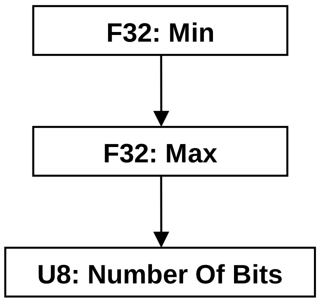


Figure 143 — Uniform Quantizer Data collection

F32: Min

Min specifies the minimum of the quantized range.

F32: Max

Max specifies the maximum of the quantized range.

U8: Number Of Bits

Number of Bits specifies the quantized size (i.e. the number of bits of precision). In general, this value shall satisfy the following condition: “0 <= Number Of Bits <= 32”.

12.1.13 Compressed Entity List for Non-Trivial Knot Vector

Compressed Entity List for Non-Trivial Knot Vector data collection specifies index identifiers (i.e. indices to particular entities within a list of entities) for a set of entities that contain Non-Trivial Knot Vectors. The entity types which can contain non-trivial knot vectors include:

JT B-Rep NURBS Surfaces

JT B-Rep PCS NURBS Curves

JT B-Rep MCS NURBS Curves

Wireframe MCS NURBS Curves

Note that any one occurrence of Compressed Entity List for Non-Trivial Knot Vector data collection will only contain index identifiers for one particular type of the above listed entities. The entity type is inferred based on the data collection which includes/references the Compressed Entity List for Non-Trivial Knot Vector.

A trivial knot vector is one which completely satisfies all conditions of at least one of the following cases:

Case-1 for trivial knot vector:

- Number of knots is an even number.
- Knot vector has a [0:1] knot range.
- There are no interior knots (i.e. NumberKnots = = 2 * (NurbsEntityDegree + 1).

Case-2 for trivial knot vector:

- Number of knots is an even number.
- Knot vector has a [0:1] knot range.
- NurbsEntityDegree < 3.
- Difference between successive non-repeating knots (i.e. KnotDelta) is:
  - KnotDelta = 2.0 / (NumberKnots – (2.0 * NurbsEntityDegree)).

Any knot vector which does not satisfy one of the above cases for “trivial knot vector” is classified as a “non-trivial knot vector.”

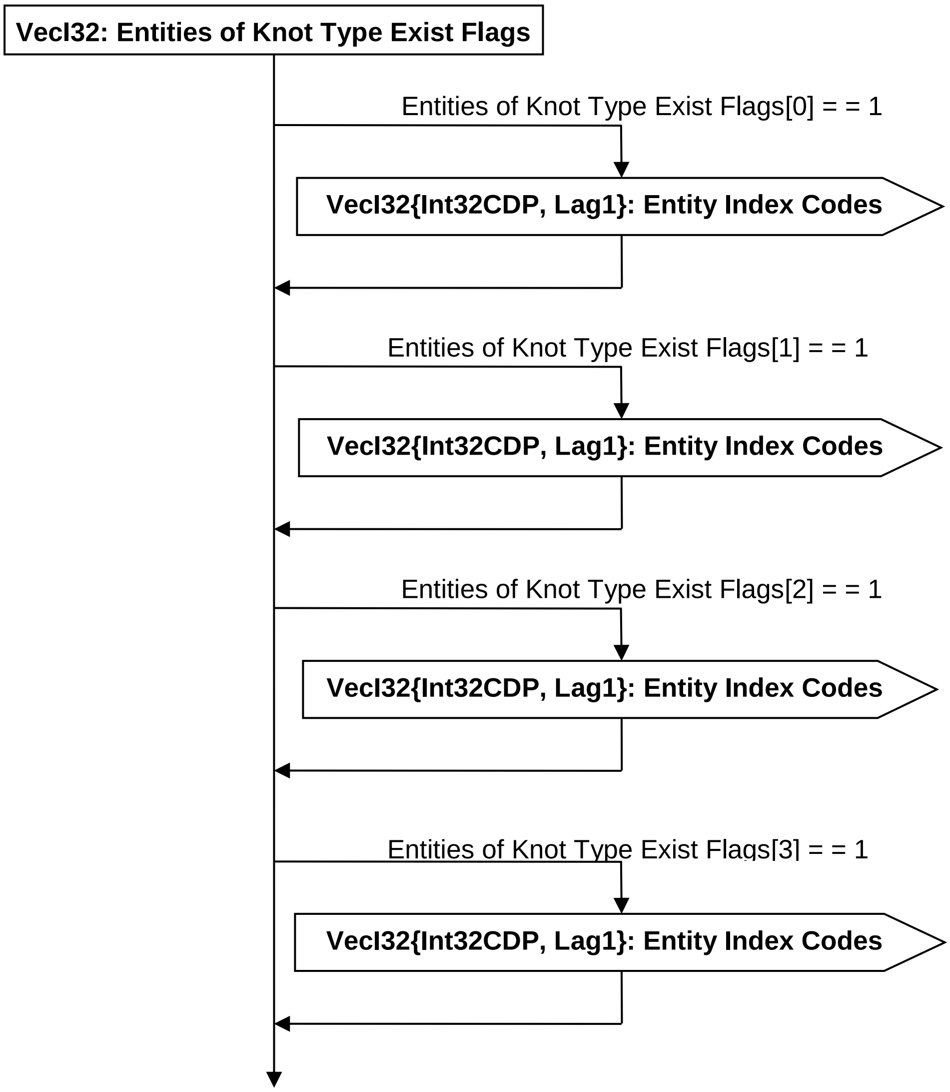


Figure 144 — Compressed Entity List for Non-Trivial Knot Vector data collection

VecI32: Entities of Knot Type Exist Flags

Entities of Knot Type Exist Flags, is a vector of flags indicating for each knot vector type whether Entity Index ID data collections exist/follow for that knot vector type. Knot Vectors are categorized into types based on the following characteristics: whether internal knots occur in *adjacent pairs* and whether the knot range is [0:1] or some other [x_1:x_2] range.

Currently there are four knot vector types, so this Entities of Knot Type Exist Flags vector should be of length four. The four flags have the following meaning:

Table 67 — Knot Type Exist Flag values

[0]	Flag indicating whether Entity IDs data collection exists for "Even Count [0:1] Range" knot type. Knots in this category have their knot range on [0:1], internal knots occur in *adjacent pairs*, *except* when there are no internal knots, in which case Type = 1 instead. = 0 – No Entity IDs data collection exists. = 1 – Entity IDs data collection exists.
[1]	Flag indicating whether Entity IDs data collection exists for "Even Count [x_1:x_2]

	Range” knot type. Knots in this category have their knot range on [x_1:x_2], and internal knots occur in *adjacent pairs*. = 0 – No Entity IDs data collection exists. = 1 – Entity IDs data collection exists.
[2]	Flag indicating whether Entity IDs data collection exists for “Odd Count [0:1] Range” knot type. Knots of this type have their knot range on [0:1], and are not Type 0. = 0 – No Entity IDs data collection exists. = 1 – Entity IDs data collection exists.
[3]	Flag indicating whether Entity IDs data collection exists for “Odd Count [x_1:x_2] Range” knot type. Knots of this type have their knot range on [x_1:x_2], and are not Type 1. = 0 – No Entity IDs data collection exists. = 1 – Entity IDs data collection exists.

Examples of knot vectors of Type 0:

```
0 0 X X 1 1
0 0 X X Y Y 1 1
0 0 X X Y Y Z Z 1 1
```

Examples of knot vectors of Type 1:

```
0 0 1 1         (Note: This is the exception to Type 0)
X X Y Y
X X Y Y Z Z
X X Y Y Z Z W W
```

Examples of knot vectors of Type 2:

```
0 0 X 1 1
0 0 X Y 1 1
0 0 X Y Z 1 1
0 0 X X X 1 1
0 0 X X Y Z Z 1 1
```

Examples of knot vectors of Type 3:

```
X X Y Z Z
X X Y Z W W
```

With this information in hand, the reader is able to reconstruct complete knot vectors in the following manner. When reconstructing the knot vector, you only take just enough values from the decoded knot value array. This may be as few as one. All the other values are inferred. Here's a sketch of the reconstruction algorithm:

```
// Number of knots in the knot vector
cNumKnots = numCtlPts + degree + 1;
// Necessary knot multiplicity at both ends of the knot vector
cClamping = degree + 1;
switch (knotType) {
    // Clamping is 0..1, internal knots occur in ADJACENT PAIRS
    // *EXCEPT* when there are no internal knots, in which case
    // Type = 1 instead.
    case 0: numVals = (cNumKnots - 2 * cClamping)/2;
    // Clamping is X1..X2, internal knots occur in ADJACENT PAIRS
    case 1: numVals = (cNumKnots - 2 * cClamping)/2 + 2;
    // Clamping is 0..1, and not Type 0
    case 2: numVals = (cNumKnots - 2 * cClamping);
    // Clamping is X1..X2, and not Type 1
    case 3: numVals = (cNumKnots - 2 * cClamping) + 2;
}
// numVals is the number of non-inferrable knot values needed
// Let vVals be the knot vector value array
```

```
// vKnot will be the final output knot vector
if (knotType is either 0 or 2)
    Set vKnot[0 .. cClamping-1] to 0
    Set vKnot[cNumKnots-cClamping .. cNumKnots-1] to 1
else
    Set vKnot[0 .. cClamping-1] to vVals[0]
    Set vKnot[cNumKnots-cClamping .. cNumKnots-1] to vVals[numVals-1]
Set vKnot[cClamping .. cNumKnots-cClamping-1] from vVals[1 .. numVals-2]
```

VecI32{Int32CDP, Lag1}: Entity Index Codes

Entity Index Codes is a vector of quantizer “codes” representing entity index identifiers for a set of entities (i.e. indices to particular entities within a list of entities). Entity Index Codes uses the Int32 version of the CODEC to compress and encode data.

12.1.14 Compressed Control Point Weights Data

Compressed Control Point Weights Data collection is the compressed and/or encoded representation of weight data for some set of Control Points. All NURBS based geometry use this data collection to compress/encode Control Point Weight data.

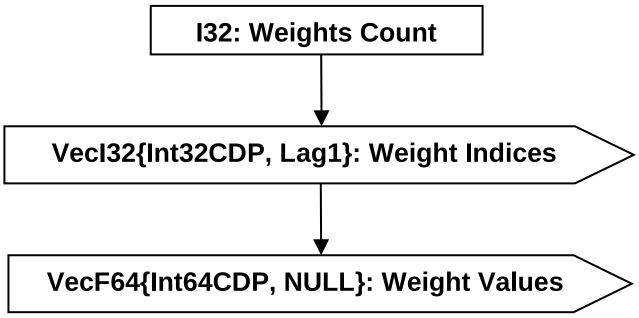


Figure 145 — Compressed Control Point Weights Data collection

I32: Weights Count

Weights Count specifies the total number of Weights. This count can differ from the Control Point count (see A.1.4.1.3 NURBS Surface Control Point Counts) because if the Control Point Dimensionality is non-rational (see data field), then no Weight values are stored for the particular Control Point. Weights Count value also does not necessarily equate to the actual number of Weights stored, since if a particular Control Point’s Weight values is “1”, then no actual Weight value is stored (i.e. JT file loaders/readers can infer that the Weight Value is “1” for Control Points that don’t have a Weight value stored).

VecI32{Int32CDP, Lag1}: Weight Indices

Weight Indices is a vector of indices representing the index identifiers for the conditional set of weights for which an actual Weight Values is stored in Weight Values. Weight Indices uses the Int32 version of the CODEC to compress and encode data.

VecF64{Int64CDP, NULL}: Weight Values

Weight Values is a vector of weight values for the conditional set of weights. Weight Values uses the Int64 version of the CODEC to compress and encode data. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

12.1.15 Compressed Curve Data

Compressed Curve Data collection contains JT B-Rep or Wireframe Rep compressed/encoded geometric Curve data. Currently only NURBS Curve types are supported as part of this data collection. Complete documentation for JT B-Rep and Wireframe Rep can be found in the sections on JT B-Rep Element and Wireframe Rep Element respectively.

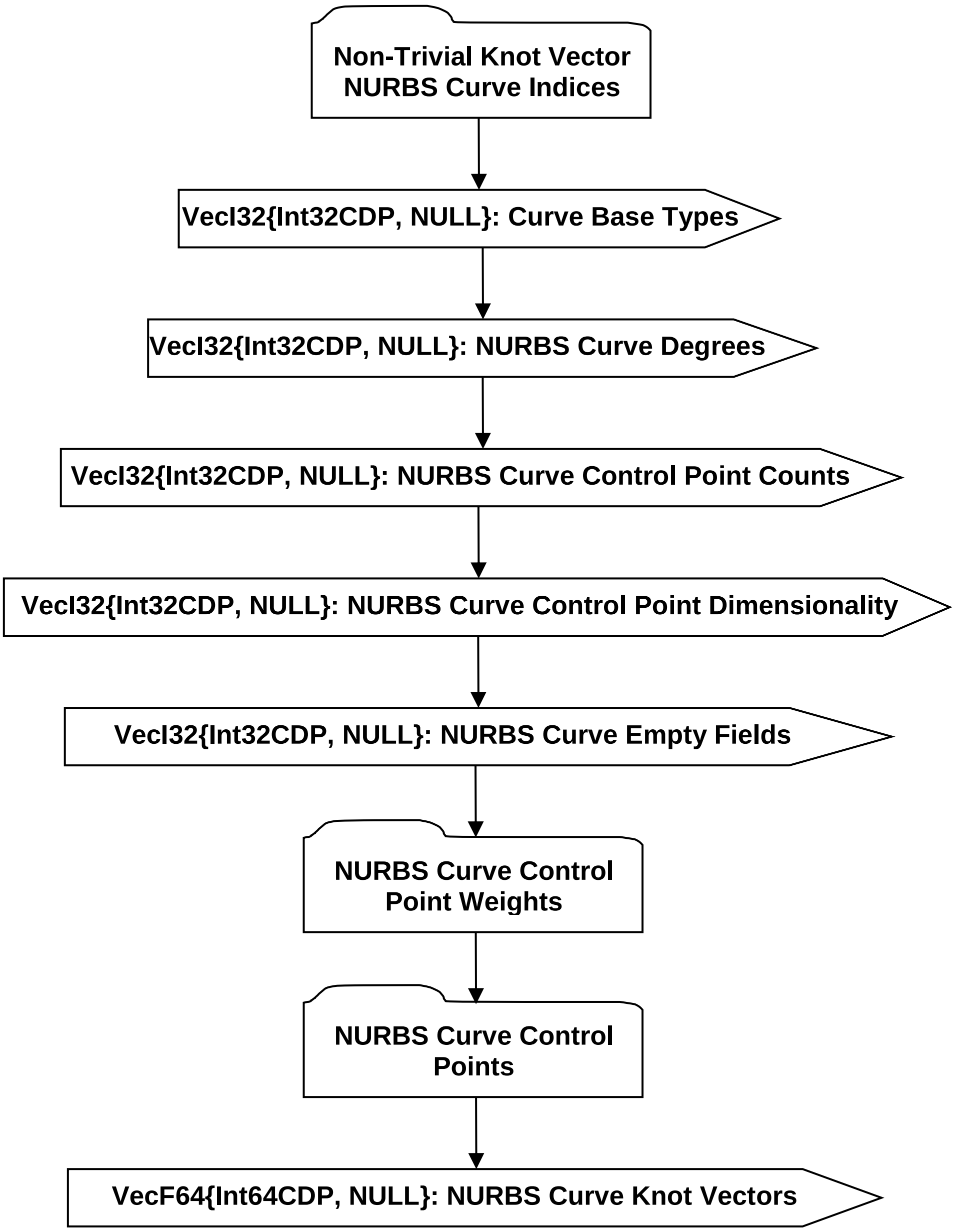


Figure 146 — Compressed Curve Data collection

VecI32{Int32CDP, NULL}: Curve Base Types

Each Curve is assigned a base type identifier. Curve Base Types is a vector of base type identifiers for each Curve in a list of Curves. Currently only NURBS Curve Base Type is supported, but a type identifier is still included in the specification to allow for future expansion of the JT Format to support other curve types.

In an uncompressed/decoded form the Curves base type identifier values have the following meaning:

Table 68 — Compressed Curve Base Type values

= 1	Curve is a NURBS curve

Curve Base Types uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: NURBS Curve Degrees

NURBS Curve Degrees is a vector of Curve degree values for each NURBS Curve in a list of Curves (there is a stored value for each NURBS Curve in the list). NURBS Curve Degrees uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: NURBS Curve Control Point Counts

NURBS Curve Control Point Counts is a vector of control point counts for each NURBS Curve in a list of curves (there is a stored value for each NURBS Curve in the list). NURBS Curve Control Point Counts uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: NURBS Curve Control Point Dimensionality

NURBS Curve Control Point Dimensionality is a vector of control point dimensionality values for each NURBS Curve in a list of Curves (i.e. there is a stored values for each NURBS Curve in the list).

In an uncompressed/decoded form the control point dimensionality values meaning is dependent upon the NURBS Entity type.

For NURBS UV Curve entities the dimensionality value has the following definition:

Table 69 — NURB UV Curve entity dimensionality values

= 2	Non-Rational (each control point has 2 coordinates)
= 3	Rational (each control point has 3 coordinates)

For NURBS XYZ Curve entities the dimensionality value has the following definition:

Table 70 — NURB XYZ Curve entity dimensionality values

= 3	Non-Rational (each control point has 3 coordinates)
= 4	Rational (each control point has 4 coordinates)

NURBS Curve Control Point Dimensionality uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: NURBS Curve Empty Fields

NURBS Curve Empty Fields is a vector of data. Each NURBS Curve in a list of Curves has one reserved data field entry in this NURBS Curve Empty Fields vector. NURBS Curve Empty Fields uses the Int32 version of the CODEC to compress and encode data. Refer to Common Data Conventions and Constructs, Empty Field.

VecF64{Int64CDP, NULL}: NURBS Curve Knot Vectors

NURBS Curve Knot Vectors is a list of knot vector values for each NURBS Curve having non-trivial knot vectors in a list of Curves (i.e. there are stored values for each non-trivial knot vector NURBS Curve in the list). All these NURBS Curve non-trivial knot vectors are accumulated into this single list in the same order as the Curve appears in the Curve list (i.e. Curve-N Non-Trivial Knot Vector, Curve-M Non-Trivial Knot Vector, etc.). The NURBS Curves for which knot vectors are stored (i.e. those containing non-trivial knot vectors) are identified in data collection Non-Trivial Knot Vector NURBS Curve Indices documented in Non-Trivial Knot Vector NURBS Curve Indices. NURBS Curve Knot Vectors uses the Int64 version of the CODEC to compress and encode data. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

Non-Trivial Knot Vector NURBS Curve Indices

Non-Trivial Knot Vector NURBS Curve Indices data collection specifies the Curve index identifiers (i.e. indices to particular NURBS Curves within a list of Curves) for all NURBS Curves containing non-trivial knot vectors. A description/definition for “non-trivial knot vector” can be found in Compressed Entity List for Non-Trivial Knot Vector.

This Curve index data is stored in a compressed format.

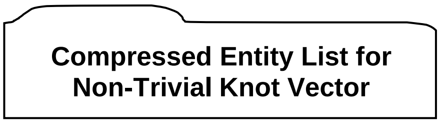


Figure 147 — Non-Trivial Knot Vector NURBS Curve Indices data collection

Complete description for Compressed Entity List for Non-Trivial Knot Vector can be found in Compressed Entity List for Non-Trivial Knot Vector.

NURBS Curve Control Point Weights

NURBS Curve Control Point Weights data collection defines the Weight values for a conditional set of Control Points for a list of NURBS Curves. The storing of the Weight value for a particular Control Point is conditional, because if NURBS Curve Control Point Dimension is “non-rational” or the actual Control Point’s Weight value is “1”, then no Weight value is stored for the Control Point (i.e. Weight value can be inferred to be “1”).

The NURBS Curve Control Point Weights data is stored in a compressed format.

Figure 148 — NURBS Curve Control Point Weights data collection

Complete description for Compressed Control Point Weights Data can be found in Compressed Control Point Weights Data.

NURBS Curve Control PointsNURBS Curve Control Points is the compressed and/or encoded representation of the Control Point coordinates for each NURBS Curve in a list of Curves (i.e. there are stored values for each NURBS Curve in the list). Note that these are non-homogeneous coordinates (i.e. Control Point coordinates have been divided by the corresponding Control Point Weight values).

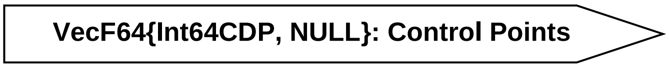


Figure 149 — NURBS Curve Control Points data collection

VecF64{Int64CDP, NULL}: Control Points

Control Points is a vector of Control Point coordinates for all the NURBS Curves in a list of Curves. All the NURBS Curve Control Point coordinates are accumulated into this single vector in the same order as the Curve appears in the Curve list (i.e. Curve-1 Control Points, Curve-2 Control Points, etc.). Control Points uses the Int64 version of the CODEC to compress and encode data in a “lossless” manner. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

12.1.16 Compressed CAD Tag Data

The Compressed CAD Tag Data collection contains the persistent IDs, as defined in the CAD System, to uniquely identify individual CAD entities (e.g. Faces and Edges of a JT B-Rep, PMI, etc.). Exactly what CAD entity types have CAD Tags and what order they are stored in Compressed CAD Tag Data is defined by users of this data collection.

What constitutes a CAD Tag is outside the scope of the JT File format and is indeed part of the CAD system. The JT File format simply provides a way to store any kind of CAD Tag as provided by the CAD system which produced the CAD entity.

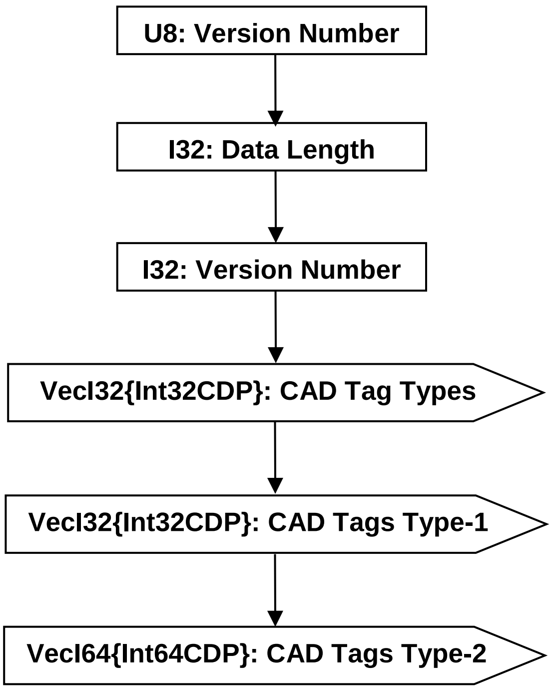


Figure 150 — Compressed CAD Tag Data collection

U8: Version Number

Version Number is the version identifier for the CADTag element. For information on local version numbers see Common Data Conventions and Constructs, Local version numbers.

I32: Data Length

Data Length specifies the length in bytes of the Compressed CAD Tag Data collection. A JT file loader/reader may use this information to compute the end position of the Compressed CAD Tag Data within the JT file and thus skip reading the remaining Compressed CAD Tag Data.

I32: Version Number

Version Number is the local version identifier for the Compressed CAD Tag Data. For information on local version numbers see Common Data Conventions and Constructs, Local version numbers.

VecI32{Int32CDP}: CAD Tag Types

CAD Tag Types is a vector of type identifiers for a list of CAD Tags (where each CAD Tag in the list has a type identifier value).

In an uncompressed/decoded form the CAD Tag type identifier values have the following meaning:

Table 71 — Compressed CAD Tag Type values

= 1	32 Bit Integer CAD Tag Type
= 2	64 Bit Integer CAD Tag Type

CAD Tag Types uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP}: CAD Tags Type-1

CAD Tags Type-1 is a vector of the Type-1 (i.e. 32 Bit Integer Type) CAD Tags for a list of CAD Tags. CAD Tags Type-1 uses the Int32 version of the CODEC to compress and encode data. CAD Tags Type-1 is only present if there are Type-1 CAD Tags in the CAD Tag Types vector.

VecI64{Int64CDP}: CAD Tags Type-2

CAD Tags Type-2 is a vector of the Type-2 (i.e. 64 Bit integer Type) CAD Tag data for a list of CAD Tags. CAD Tags Type-2 uses the Int64 version of the CODEC to compress and encode data. CAD Tags Type-2 is only present if there are Type-2 CAD Tags in the CAD Tag Types vector.

12.2 Encoding Algorithms

The following sections give a brief technical overview/descriptions of the various encoding algorithms used in the JT format. A sample implementation of the encoding and decoding portion of each algorithm can be found in this document Annex.

12.2.1 Uniform Data Quantization

Uniform Data Quantization is a lossy encoding technique in which a continuous set of input values (floating point data) is approximated with integral multiples (i.e. integers) of a common factor. How close the quantization output approximates the original input data is dependent upon the quantization data range and the number of bits specified to hold the resulting integer value.

The quantization is considered “uniform” because the algorithm divides the data input range into levels of equal spacing (i.e. a uniform scale). The form of Uniform Data Quantization used by the JT format is also considered scalar in nature, in that each input value is treated separately in producing the output integer value.

Given the following definitions:

```
inputVal:     Input floating point data to quantize
outputval:       Resulting quantized output integer value
minInputRange:   Specified minimum value of input data range
maxInputRange:   Specified maximum value of input data  range
nBits: Specified number of bits of precision (quantized size)
The basic algorithm (using C++ style syntax) for Uniform Data Quantization is as
follows:
UInt32 iMaxCode = (nBits < 32) ? (0x1 << nBits) - 1 : 0xffffffff;
Float64 encodeMultiplier = Float64(iMaxCode) / (maxInputRange – minInputRange);
UInt32 outputVal = UInt32( (inputVal - minInputRange) * encodeMultiplier + 0.5 );
```

Note: For reasons of robustness, “outputVal” shall also be explicitly clamped to the range [0,iMaxCode]. This is because floating-point roundoff error in the calculation of “encodeMultiplier” can otherwise cause “outputVal” to sometimes come out equal to “iMaxCode + 1”.

Note that all compression algorithms in the following sections operate on quantized integer data.

12.2.2 Bitlength CODEC

This is a very simple compression algorithm that runs either a fixed-width or adaptive-width bit field encoding for each value. It is used whenever none of the more sophisticated CODECs are able to extract any compression advantage. In essence, the Bitlength CODEC takes advantage of the fact that most of the values will require less than 32 bits to represent, and so can be written as bitfields narrower than 32 bits. In some cases, the best answer is to choose a fixed field width that can represent all values in the array. In other cases, a little more compression can be had by using an adjustable-width coding scheme.

When using the variable-width scheme, as each input value is encountered, the number of bits needed to represent it is calculated and compared to the current "field width". The current field width is then adjusted upwards or downwards by a constant “step_size” number of bits (i.e. 2 bits for the JT format) to accommodate the input value storage. This increment or decrement of the current field width is indicated for each encoded value by a prefix code stored with each value.

The prefix code will be one of the following two forms:

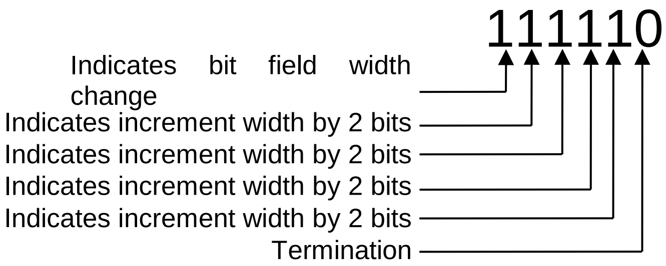
A single '0' bit to denote the same (i.e. current) field width is to be used for the next value.

A '1' bit followed by a series of one or more bits where each bit indicates whether the field width is to be incremented (a '1' bit) or decremented (a '0' bit) by the field step_size, followed by a single terminator bit (which is complement of the previous increment/decrement bit). Note that there can only be increments or decrements in a given prefix code, never both, and that is why the prefix code terminator bit can be recognized as bits are read by simply looking for the complement of the previous increment/decrement bit.

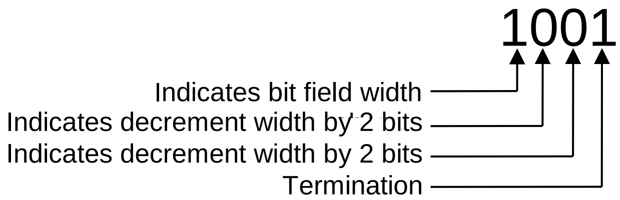
Some examples of prefix codes and their interpretation are as follows:

Example 1: Prefix code to maintain same (current) field width.

Example 2: Prefix code to increment field width four times (8 bits).



Example 3: Prefix code to decrement field width two times.



A pseudo-code sample implementation of bit length decoding is available in the Annex.

12.2.3 Arithmetic CODEC

In 1948, Claude Shannon of Bell Laboratories published his seminal paper “A mathematical theory of communication” that launched the new field of Information Theory. In that same year, two Doctoral students at the Massachusetts Institute of Technology (MIT) made breakthroughs in the coding of information. The first to press was David Huffman, whose coding scheme we now know as Huffman Coding. In that same class with Huffman was Peter Elias who reportedly developed the first articulation of arithmetic coding, but it lay unpublished until 1976, when Jorma Rissanen and Richard Pasco, of IBM, refined it into a practically useful algorithm.

Arithmetic encoding is a so-called “entropy coding” algorithm that replaces an input stream of symbols or bytes with a single fixed point output number (i.e. only the mantissa bits to the right of the binary point are output from MSB to LSB). The total number of bits needed in the output number is dependent upon the length and statistical properties of the input message (i.e. the longer the input message the more bits needed in the output number). This single fixed point number output from an arithmetic encoding process shall be uniquely decodable to create the exact stream of input symbols that were used to create it.

Initially all symbols being encoded have a probability value assigned to them based on the likelihood that the symbol will occur next in the input stream (i.e. the frequency of the symbol in the input stream). Given probability value assignments, each individual symbol is then assigned an interval range along a nominal 0 to 1 “probability line”, where the size of each range corresponds to the symbol’s probability value. Note that a particular symbol owns all values within its assigned range up to, but not including, the range high value, and that it does not matter which symbols are assigned which segment of the range as long it is done in the same manner by both the encoder and the decoder.

Given the above described input stream probability and interval range assignments, a high level description of the arithmetic encoding process is as follows:

Begin with a “current interval” initialized to [0,1). Note, that in interval range notation (i.e. “[0,1)”), the “[“ symbol indicates inclusive of the interval low limit and “)” symbol indicates exclusive of the interval high limit.

Sequentially for each symbol of the input stream, perform two steps.

Subdivide the current interval into subintervals based on the input stream symbol probability values as described above.

Select the subinterval corresponding to the current input stream symbol being sequentially processed and make it the new “current interval”.

After all input stream symbols have been sequentially processed; output enough bits to distinguish the final “current interval” from all other possible final intervals.

In pseudo code form, the algorithm to accomplish the above described arithmetic encoding for an input stream message of any length could look as follows:

```
Set low to 0.0
Set high to 1.0
While there are still input symbols do
    cur_symbol = get next input symbol
    range = high - low
    high = low + range * high_range(cur_symbol)
    low = low + range * low_range(cur_symbol)
End of While
Output low
```

So the arithmetic encoding process is simply one in which we narrow the range of possible numbers with every new sequentially processed input symbol; where the new narrowed range is proportional to the predefined probability values assigned to each symbol in the input stream.

The arithmetic decoding process is the inverse procedure; where the range is expanded in proportion to the probability of each symbol as it is extracted. For the arithmetic decoding process we find the first symbol in the message by seeing which symbol owns the interval range that our encoded message falls in. Then, since we know the low and high range limit values of the first symbol we can remove their effects by reversing the process that put them in.

In pseudo code form, the algorithm for decoding the incoming number could look as follows:

```
Get encoded_number
Do
    find symbol whose range straddles the encoded_number
    output the symbol
    range = symbol_high_value - symbol_low_value
    encoded_number = encoded_number - symbol_low_value
    encoded_number = encoded_number / range
until no more symbols
```

Example

Following is an example to demonstrate in practice the basic principles of arithmetic coding.

Suppose you want to compress, using arithmetic coding, the following sequence/array of integer data:

{2, 9, 12, 12, 0, 7, 1, 20, 5, 19}

For this input stream of data, the assigned probability values will be as follows:

Table 72 — Example assigned probability values

Number	Probability
0	1/10
1	1/10
2	1/10
5	1/10
7	1/10
9	1/10
12	2/10

Number	Probability
19	1/10
20	1/10

Then based on each input numbers probability value, an interval range along a 0 to 1 “probability line” can be assigned to each input number as follows:

Table 73 — Example “probability line” values

Number	Probability	Range
0	1/10	[0.00, 0.10)
1	1/10	[0.10, 0.20)
2	1/10	[0.20, 0.30)
5	1/10	[0.30, 0.40)
7	1/10	[0.40, 0.50)
9	1/10	[0.50, 0.60)
12	2/10	[0.60, 0.80)
19	1/10	[0.80, 0.90)
20	1/10	[0.90, 1.00)

Now proceeding with encoding the example input integer sequence {2, 9, 12, 12, 0, 7, 1, 20, 5, 19}, the first number to be encoded is “2”; so the final encoded value will be a number that is greater than or equal to 0.20 and less than 0.30. Now as each subsequent number in the input stream is sequentially processed for encoding, the possible range of the output number is further restricted. In our example the next number to be encoded is “9” which owns the range [0.50, 0.60) within the new sub-range of [0.20, 0.30); which now further restricts our output number to the range [0.25, 0.26). If we continue this logic for the complete input integer sequence we end up with the following:

Table 74 — Example input integer sequence values

New integer number	Low value	High value
	0.0	1.0
2	0.2	0.3
9	0.25	0.26
12	0.256	0.258
12	0.2572	0.2576
0	0.25720	0.25724
7	0.257216	0.257220
1	0.2572164	0.2572168
20	0.25721676	0.2572168
5	0.257216772	0.257216776
19	**0.2572167752**	0.2572167756

From the above table, are final low values is “0.2572167752” which is the output number that uniquely encodes the integer number sequence {2, 9, 12, 12, 0, 7, 1, 20, 5, 19}.

Given this encoding scheme, the decoding would simply follow the process previously described. We find the first number in the sequence by looking up in the probability range for the value, whose range, our encoded number “0.2572167752” falls within. In our example this equates to the value “2” and so our first decoded value shall be “2”. Then we apply the previously described decoding subtraction and division steps to arrive at a new encoded value of “0.572167752”. Using this new “0.572167752” encoded value and the same logic of the first step, the second decoded value will be “9”. We continue this process until there are no more numbers to decode.

In practice, due to floating point size (i.e. number of bits) restrictions and possible differences in floating point formats on machines, arithmetic encoding is best implemented using 16 bit or 32 bit integer math. Using 16 bit or 32 bit integer math, an incremental transmission scheme can be implemented, where fixed size integer state variables receive new bits in at the low end and shift them out the high end, forming a single number that can be as many bits long as are available on the computer’s storage medium.

Using our example as a guide, define the starting range [0.0, 1.0) to instead be 0 to 0.999 (which is .111 in binary). Then in order to use integer registers to store these numbers, justify the values so that the implied decimal point is at the left hand side of the word. Now load the initial range values based on the word size we are using. In the case of a 16 bit implementation the initial range values will be low equals 0x0000 and high equals 0xFFFF. Since we know these values will go on forever (e.g. 0.999… will continue with FFs) we can shift those extra bits in as needed with no detrimental effects.

Going back to our example and using a 5 digit register, we start with the range:

High: 99999

Low: 00000

Applying the previously described encoding algorithm we first calculate the range between the low and high values; which in this case is 100000 (not 9999 since we assume the high value has an infinite number of 9’s). Next, we calculate the new high value which in this example will be 30000. But before we store the new high value we shall decrement it to account for the implied digits appended to it; so new high value will be 29999. Applying similar logic to computing the new low value results in a new range of:

High: 29999 (999…)

Low: 20000 (000…)

In looking at the newly computed high and low range values, it can be seen that the most significant digits of high and low match. A property of arithmetic coding is that as this encoding process continues, the high and low values will continue to get closer, but will never match exactly. Given this property, once the most significant digit of high and low match, it will never change, and thus we can output this most significant digit as the first number in the coded word and continue working with just 16 bit high and low values. This output process is accomplished by shifting both the high and low values left by one digit and shifting in a “9” in the least significant digit of the high value.

Applying the previously described encoding algorithm and continuing the above described process of shifting out most significant digit into the coded word as high and low continually grow closer together looks as follows for encoding our example integer number sequence {2, 9, 12, 12, 0, 7, 1, 20, 5, 19}:

Table 75 — Example integer number sequence values

	High	Low	Range	Cumulative output
Initial State	99999	00000	100000	
Encode “2” [0.2, 0.3)	29999	20000		
Shift out 2	99999	00000	100000	.2
Encode “9” [0.5, 0.6)	59999	50000		.2
Shift out 5	99999	00000	100000	.25
Encode “12” [0.6, 0.8)	79999	60000	20000	.25
Encode “12” [0.6, 0.8)	75999	72000		.25
Shift out 7	59999	20000	40000	.257
Encode “0” [0.0, 0.1)	23999	20000		.257

	High	Low	Range	Cumulative output
Shift out 2	39999	00000	40000	.2572
Encode “7” [0.4, 0.5)	19999	16000		.2572
Shift out 1	99999	60000	40000	.25721
Encode “1” [0.1, 0.2)	67999	64000		.25721
Shift out 6	79999	40000	40000	.257216
Encode “20” [0.9, 1.0)	79999	76000		.257216
Shift out 7	99999	60000	40000	.2572167
Encode “5” [0.3, 0.4)	75999	72000		.2572167
Shift out 7	59999	20000	40000	.25721677
Encode “19” [0.8, 0.9)	55999	52000		.25721677
Shift out 5	59999	20000	40000	.257216775
Shift out 2				.2572167752
Shift out 0				.25721677520

As can be seen in the above table, after all values in the input stream have been encoded and any final matching most significant digit has been output, the arithmetic coding algorithm requires that two extra digits be shifted out of either the high or low value to finish up the cumulative output word.

Although the above example incrementally encodes very nicely with the arithmetic coding algorithm, there are certain cases where the computed high and low values get closer, but never actually converge to one value in the most significant digit (e.g. High = 0.300001, Low = 0.29992). Thus after a few iterations the difference between high and low becomes so small that 16 bits is not sufficient to represent any difference between the values (i.e. all calculations return the same values). This conditions is known as “underflow” and special logic shall added to the arithmetic coding algorithm to recognize that “underflow” is occurring and thus head it off before the computations reach an impasse.

The additional logic for recognizing that “underflow” is occurring would be executed after each recalculation of High and Low value set, and in pseudo code form this logic would look as follows:

```
underflow = FALSE
if(  (High and Low value’s significant digits don’t match but are on adjacent
numbers) &&
 (2nd MSDMSD of High is “0” and the 2nd MSDMSD of low is “9”)  )
{
    underflow = TRUE
}
```

When/If it is identified that “underflow” is occurring, the encoding algorithm shall perform the following steps to stop the current “underflow”:

Delete the 2nd most significant digit from both the High and Low value.

Shift the other digits (those to the right of the deleted 2nd digit) to the left to fill up the space (note that the most significant digit stays in place).

Increment a counter to remember that we threw away a digit and don’t know whether it was going to converge to “0” or “9”.

A before and after example of performing the above steps to the High and Low values when ‘underflow” occurs is as follows:

	Before	After
High	40344	43449
Low	39810	38100
Underflow_counter	0	1

Now as the encoding algorithm continues and the most significant digit of High and Low values once again converge to a common value, then that value shall be output to the coded word along with

"Underflow_counter" number of "underflow" digits that were previously deleted. The underflow digits output to the coded word will either be all 9s or 0s, depending on whether the High and Low value converged to the higher or lower value.

A pseudo-code sample implementation of arithmetic decoding is available in the annex.

12.2.4 Deering Normal CODEC

Michael Deering first published his work on geometry compression in 1995 [2] and later helped present a course on the subject at SIGGRAPH'99 [3]. Although Deering's approach to geometric compression involves compression of vertices, colours and normals, the description detailed here will focus solely on compression of normals since this is the only component of Deering's approach used in the JT format.

Through both theoretical examination and empirical testing, Deering found that an angular density of 0.01 radians between normals (about 100,000 normalized normals distributed over unit sphere) gave results that were not visually distinguishable from results obtained from finer normal representations. This observation reduced the problem of having to "exactly" represent any general surface normal, to only having to represent about 100,000 specific normals (i.e. general surface normal replaced by the appropriate one of the 100,000 specific normals).

If there were no run-time memory concerns and no concerns for on disk footprint size, these specific 100,000 normals could be simply represented in a table that is indexed into, to reference a particular normal. Instead, Deering's approach leverages symmetrical properties of the unit sphere to reduce the size of the table and allow any normal to be represented by, at max, an 18 bit index as summarized below:

- All normals are normalized (i.e. can be represented as points on the surface of the unit sphere).
- Unit sphere is divided into eight symmetrical octants based on sign bits of normal's X,Y,Z rectilinear representation (see Figure 155).
- Using three bits to represent the three sign bits of the normals XYZ components reduces the problem space to one eighth of the unit sphere.
- Each octant of the unit sphere is divided into six identical sextants by folding about the planes of symmetry; x=y, x=z, and y=z (see Figure 155)
- . The particular sextant can be encoded using another three bits. So now unit sphere is divided into 48 identically shaped triangle patches reducing the normal look-up table to about 2000 entries (i.e. 100000/48).
- Then, a local rectangular orthogonal two dimensional grid is created on the sextant and all normals within the sextant are represented as two n-bit angular addresses (i.e. a quantization of two angular values along the unit sphere) where "n" is in the range from 0 to 6 bits.
- Resulting in a max grand total of 18 bits (3 + 3 + 6 + 6) to represent any normal on the unit sphere.

In the figure below, the sphere is divided into eight octants and each octant is divided into six sextants. Each sextant is assigned an identifying three bit code.

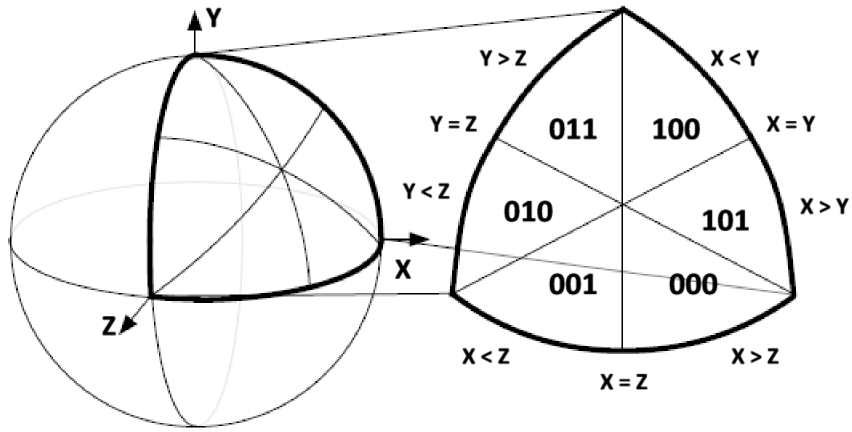


Figure 151 — Sextant Coding on the Sphere

Note that the sextant three bit code assignments used by the JT format (as seen in

Figure 155) are slightly modified from the original assignments as specified by Deering.

The representation of all normals within a sextant by two n-bit angular addresses, as summarized above, is based on the following:

— In spherical coordinates, points on a unit sphere can be parameterized by two angles, θ and φ; where θ is the angle about the y axis and φ is the longitudinal angle from the y=0 plane.

— Mapping between rectangular and spherical coordinates is:

$x = \cos\theta * \cos\varphi \qquad y = \sin\varphi \qquad z = \sin\theta * \cos\varphi.$

— All encoding takes place in the positive octant.

— Angles θ and φ can be quantized into two n-bit integers θ'_n and φ'_n (where "n" is in the range of 0 to 6) and the relationship between these n-bit integers and angles θ and φ for a given "n" is:

$\theta(\theta'_n) = \operatorname{asin} \tan(\varphi_{max} * (n - \theta'_n) / 2n)$
$\varphi(\varphi'_n) = \varphi_{max} * \varphi'_n / 2n.$

Thus to encode (i.e. quantize) a given normal **N** into θ'_n and φ'_n:

— N shall be first represented (see

— Figure 155) in the positive octant and appropriate sextant within that octant, resulting in N'.

— Then N' shall be dotted with all quantized normals in the sextant.

— For a fixed "n", the corresponding θ'n and φ'n values of the quantized sextant normal that result in the largest (nearest unity) dot product defines the proper θ'n and φ'n encoding of N.

With this encoding of normal **N** into θ'_n and φ'_n n-bit integers the complete bit representation of normal **N** can now be defined as follows:

— Uppermost three bits specify the octant.

- Next three bits specify the sextant code as defined in Figure 155.
- Next two n-bit fields specify θ'n and φ'n values respectively.

12.3 LZMA compression

LZMA is a (lossless) dictionary-based data compression algorithm and is essentially the same as that in 7-Zip. Implementers of this specification should consider using XZ Utils for compressing and decompressing JT data. XZ Utils is free general-purpose data compression software. XZ Utils are the successor to LZMA Utils.

The core of the XZ Utils compression code is based on LZMA SDK, but it has been modified quite a lot to be suitable for XZ Utils. The primary compression algorithm is currently LZMA2, which is used inside the .xz container format. Typically, XZ Utils create 30 % smaller output than gzip and 15 % smaller output than bzip2.

Entry points from the liblzma API, used when compressing and decompressing JT files using XZ utils, are listed here;

lzma_code

lzma_easy_encoder

lzma_end

lzma_stream_decoder

The compression and decompression XZ Utils are freely available and in the public domain. For complete description and source code visit http://tukaani.org/xz/ .

13 Common Data Conventions and Constructs

The proceeding sections of this specification specify the mandatory clauses for creating a reference compliant JT file. This section documents format conventions that should be followed to promote consistency in JT.

13.1 Late-Loading Data

The JT format was designed and structured to load entities from a JT file on a deferred or as-needed basis. This concept is referred to within this specification as “late-loading data”. The JT format has many structures in support of this; writers/loaders of JT data may leverage these capabilities.

Initial loading of a JT file shall require the Table of Contents and the LSG segments.

All Meta Data Node Elements, JT B-Rep Elements, XT B-Rep Elements, Wireframe Rep Elements, PMI Manager Meta Data Elements, JT ULP Elements, JT LWPA Elements, and Shape LOD Elements may be ignored until they are actually needed. These Late-Loaded data containers are accessed via a Late Loaded Property Atom Element which appears in a LSG Node's Property list. Contained in this Property is the GUID associated with the segment to be loaded. This GUID can be looked up in the TOC Segment, which will give the location in the JT from which to load the actual Element via the Data Segment convention.

13.2 TOC Segment Location

The TOC Segment should be located within the JT file immediately following the file header.

13.3 Bit Fields

All bits fields that are not defined as in use shall be set to “0”.

13.4 Empty Field

In the File Format section of this reference some data fields may be named/documented as “Empty Field” (e.g LOD Node Data ”Empty Field” field). These fields should be treated as follows:

If you are writing a JT file whose data did not originate from reading a previous JT file, then Empty Fields should be set to a value a “0” when writing the field to a JT file.

If you are writing a JT file whose data originated from reading a previous JT file (i.e. rewriting a JT File), then “Empty Fields” should be written with the same value that was read from the originating JT file.

13.5 Local version numbers

The version numbers seen throughout the data collections are version numbers local to those data types. They provide a simple means by which those data collections can be extended. All version information for 10.0 JT data is included within this document.

For each data collection, data for each local version should be written in sequence. When reading the data, the local version number allows readers to read up to the maximum local version they support and then use the segment length that was read in the Segment Header to skip over additional data. For example when version one and two data is present the user can choose to read only the version one data or additionally read the version two data as required.

Local version numbers are used for conditional branching as depicted in the element figures.

13.5.1 Version numbers

"0x01"

All references to version number in this document shall be this value unless noted otherwise here.

"0x02"

Base Property Atom Element,

String Property Atom Element,

Integer Property Atom Element,

Floating Point Property Atom Element,

JT Object Reference Property Atom Element,

Date Property Atom Element,

Late Loaded Property Atom Element,

Vector4f Property Atom Element

"0x05"

JT B-Rep Element

13.6 Hash Value

Hashing is a means by which a large chunk of values can be represented by single value through the use of a mathematical function that provides a distinctive value for each unique set of ordered values. The hash function used within this format was published by Bob Jenkins in Dr Dobbs Journal in 1997. Its implementation is provided in Annex C. It is the same implementation that was used in JT v9.x.

The hash function takes a pointer to a set of values, the number of values, and a seed hash value. It returns the resulting hash value. Initially the seed value is set to 0, however when hashing multiple data fields together the hash of previous data field is used as the seed hash value of the next data field:

```
UInt32 uHash = 0;
uHash = hash32( pVal0, nVal0, uHash );
uHash = hash32( pVal1, nVal1, uHash );
```

The order that individual fields are hashed is extremely important since readers for this format should assert that the stored hash value matches the calculated hash value of the corresponding fields after reading in all the corresponding data. To this end each hash value stored within this recommendation's format carefully documents which fields it encompasses and the order in which they should be hashed.

13.7 Scene graph construction

The following guidlines apply for scenegraph construction:

1. use a Meta Data Node Element to denote a CAD "Assembly",
2. use a Part Node Element to denote a CAD "part".

Below is an example of a fully-fleshed out small assembly of a three-wheeled motorcycle:

```
Partition                (Partition Node Element)              Root node of JT file
  MetaDataNode   (Meta Data Node Element)           "Three-wheeler" Top-level assembly
    Instance            (Instance Node Element)             "Front wheel" assembly
      Partition           (Partition Node Element)             External reference to JT file for "Wheel" part
        PartNode      (Part Node Element)                   "Wheel" generic part
          RangeLOD   (Range LOD Node Element)         Level-of-detail node
                Group           (Group Node Element)                  High-LOD group node
                  TriStripSet (Tri-Strip Set Shape LOD Element) Rim geometry
                  TriStripSet (Tri-Strip Set Shape LOD Element) Tire geometry
                  TriStripSet (Tri-Strip Set Shape LOD Element) Low-LOD geometry
            MetaDataNode    (Meta Data Node Element)          "Rear Axle" assembly
              Instance             (Instance Node Element)             "Left rear wheel" assembly
                ["Wheel" PartNode above]         This instance node's child is same as the one above.
              Instance             (6.1.1.4 Instance Node Element)             "Right rear wheel" assembly
                ["Wheel" PartNode above]         This instance node's child is same as the one above.
```

Instance Node Elements are used when referring to an *instanced* part but are not fundamentally different from a Group Node Element having only one child.

13.8 Metadata Conventions

Although there are not limits to what Meta Data (i.e. properties) may be attached to nodes in the LSG, the following conventions should be followed in industry when translating CAD data to the JT file format. See the Property Atom Elements section of this document for complete description of the file Elements used to attach this property information to nodes.

13.8.1 Property Key Naming Conventions

Properties in JT are named value pairs constructed of keys and values as defined in Property Proxy Meta Data Element. Properties are used to provide information to downstream applications and processes. In order to enable different applications to read and interpret properties correctly, a common understanding and treatment for naming of keys should be followed.

13.8.1.1 Uniqueness of Property Keys

No duplicate property keys are allowed in the same scene graph node.

It is allowed to create two properties with identical keys if one of them is defined as visible and the other as hidden as described below. This is possible since the visible state changes the property key string by appending a double colon.

This definition of duplicate properties using visible and hidden should be avoided.

13.8.1.2 Hidden Properties

Properties are used for a range of purposes in a JT file; some are relevant to visual interogaton by users and some not. To enable applications with the ability to differantitate properites a convention for naming key strings should be followed. The key string pattern that is used to denote the visibility of a property is a double colon ("::"). The double colon is appended to the end of the name in the property key as shown below.

"property" = "hidden"

"property::" = "visible"

The objective of the "hidden" concept is to indicate to a viewing application that a user should not see the property in an application user interface.

For example:

The property with the key “Name::” will list the value “body_4465” in a viewing application when property display is selected.

Property type="STRING" key="Name::" value="body_4465"

The property SUBNODE will not be listed by a viewing application.

Property type="STRING" key="SUBNODE" value="1"

Defintion of key names with “::” included in the key name shall not affect visibility of that property when parsing of the JT data; i.e., “hidden” shall not be interpreted as “encrypted” or “secure” with respect to JT data.

13.8.1.3 Case-sensitivity of Property Keys

Property keys are case-sensitive; i.e. “ud_CAD_MASS” and “UD_CAD_MASS” are considered two separate properties.

13.8.1.4 Blanks (white spaces) in Property Keys

Property keys are strings, spaces are allowed. Converting blank spaces to underscores should not be followed as a convention or recommended practice.

13.8.1.5 Special Characters in Property Keys and Property Values

All strings in JT files, including property keys, are stored as Unicode (UCS-2) strings. As such, “special characters” supported by Unicode are allowed.

13.8.1.6 Maximum Length of Property Keys and Property Values

Since property keys are strings, the maximum length is theoretically 2^31 (about 2.1 billion) bytes; i.e. 2^30 characters. Keys this large should not be created. Applications writing or reading JT files have a significantly lower limit for the maximum property key length. The definition of overly large property keys may lead to data exchange issues.

13.8.1.7 Properties with an empty Value

Properties in JT are always <key, value>. Since blanks are valid values, the value can be an empty string. Such properties have to be handled properly by the consuming applications and shall not be ignored.

13.8.2 PMI Properties

See PMI Properties Annex for the full list of properties that can be found in JT parts that contain Generic PMI Entities.

Note: Implementers should use the descriptions found in PMI Manager Meta Data Element as opposed to working with PMI Data Segment.

13.8.3 CAD Properties

The CAD Properties table provides a description for properties that CAD data translators should follow when placing CAD information in a JT file as properties on various LSG nodes.

13.8.3.1 Required Properties

The CAD unit properites are required to properly interpret numeric data for analysis operations (e.g. measurement) and support the building of assemblies when reading JT files with disparate units.

The JT_PROP_MEASUREMENT_UNITS property is required to define model dimensions. It is relevant to the interpretation of geometrical values, such as coordinates of B-Rep and LOD data as well as certain properties. To avoid ambiguous interpretation, the property should appear only once per JT Part. If the property exists for two different nodes in the scene graph, the property of the lowest (last) node along a given path takes precedence.

By convention, the property JT_PROP_MEASUREMENT_UNITS is defined as a hidden property.

Table 76 — CAD Properties

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values	Required / Optional
CAD_CENTER_OF_GRAVITY	Center of gravity of solids within part	String	F64	3 space separated numeric values	Optional See Note *
CAD_FORCE_UNITS	Defines the units for forces	String	MbString	MilliNewton Newton PoundForce	Optional
CAD_DENSITY	Density of solids within part	String	F64	numeric	Optional See Note *
CAD_MASS	Mass or weight of solids within part	String	F64	numeric	Optional See Note *
CAD_MASS_UNITS	Defines the Units of mass	String	MbString	micrograms milligrams grams kilograms ounces pounds	Required
CAD_MOMENT_OF_INERTIA	Moment of inertia value	String	F64	6 space separated numeric values ;xx,yy,zz, xy,xz,yz	Optional See Note *
CAD_PART_NAME	Component name from translator	String	MbString		Optional
CAD_PROP_MATERIAL_THICKNESS	Sheet thickness within part	String	F64	numeric	Optional
CAD_PROP_YOUNGS_MODULUS	Youngs Modulus value	String	F64	numeric	Optional
CAD_SOURCE	CAD program the Part originated from	String	MbString		Optional

CAD_SURFACE_AREA	Surface area of solids within part.	String	F64	numeric	Optional See Note *
CAD_VOLUME	Volume of solids within part	String	F64	numeric	Optional See Note *
JT_PROP_MEASURE MENT_UNITS	Defines the Model Units and is therefore relevant for the reading and interpretation of the Geometry and LOD data. Note: Must be present as a hidden property using. If the property is in the scene graph twice on different nodes, then the definition that is the lowest (last along a specific path through the graph) takes precedence.	String	MbString	millimeters centimeters meters inches feet yards micrometers decimeters kilometers mils miles	Required See Note

Note * : These properties contain calculated values which are typically taken from the origin CAD system during conversion.

Note: CAD properties may appear with prefix “UD_”. These values are specified by the JT content harmonization group in order to allow users to pass user defined values independent on some CAD measurements.

Note: ISO 14306 Ed 1 states the JT_PROP_MEASUREMENT_UNIT property value is in lowercase. It may occur that the property is given in mixed case, upper case for the first letter. In this situation, some tools might interpret the JT_PROP_MEASUREMENT_UNITS property as unknown or not defined. Implementors should check for this property value with the first letter in both upper and lower case.

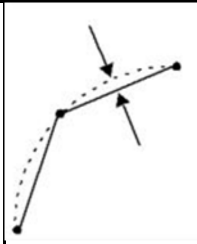
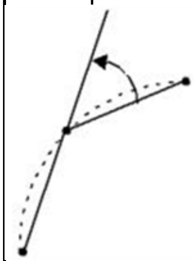
13.8.4 Tessellation Properties

The facetted graphical representations in JT are present as LODs. Three properties may be stored on Part Node Elements to indicate the tessellation tolerances used to generate each LOD. These properties are defined in the table Tesselation Property Values.

Note: Tessellation properties should be defined with the visible specifier included in their property key.

Table 77 — Tessellation Property values

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
Chordal	Chordal deviation tessellation tolerance in MCS units for each LOD. The Measure of maximum allowable distance a linear approximation for a curve/surface may deviate from the true curve/surface.	MbString	space separated F32 values Number of	Numeric

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
	 A floating point value in the range [0.0,1.0] for relative interpretation, arbitrary range for absolute interpretation.		values will be defined by the numer of LODs	
Angular	Angular tessellation tolerance for each LOD in degrees. Two consecutive segments in a linear approximation of a curve/surface form an angle; this value specifies the maximum angle allowed.  A floating point value in the range [0.0,90.0].	MbString	space separated F32 values Number of values will be defined by the numer of LODs	Numeric
SegLength	The maximum absolute length of (tessellated) line segments in a curve approximation. A floating point value of arbitrary range	MbString	F32	Numeric

13.8.5 Miscellaneous Properties

The below table documents some miscellaneous properties often placed on various nodes in the LSG to communicate specific information about the node or its contents.

Table 78 — Miscellaneous Property values

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
PMI_TYPE_TABLE	May be attached to Part Node Element to indicate the list of PMI type values and associated names for all PMI types (basically equivalent to the Entity Type field documented in Generic PMI Entities). The string is a "." and "," delimited string of the following form: "10.Groove Weld,11.Fillet Weld,12.Plug/Slot Weld,14.Edge Weld"	MbString	<string>	

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
JT_PROP_SHAPE_DATA_TYPE	May be attached to Shape Node Elements to indicate what geometry type the shape data represents.	MbString	<string>	“Surface” “Wire”
JT_PROP_ORIGINATING_BREPTYPE	May be attached to Part Node Element to indicate the type of B-Rep associated with the Part.	MbString	<string>	“None” “JtBrep” “XTBrep”

JT_PROP_NAME	May be attached to any form of node or attribute with which one wants to associate a textual name (e.g. Part/Assembly/Instance name, Material name, Light Set name, etc.).
	
	For Part/Assembly/Instance names this string has the following encoded form where “;” is a delimiter and “:’ is a terminator:
	`“AlignmentPin.part;0;`
	Name
	Version
	Instance
	
	For attribute names this string has the following encoded form:
	`“Chrome material”`
	Name
		MbString	<string>	

13.8.6 The SUBNODE property and Reference Sets

A SUBNODE property can be defined on a part or assembly node in a JT file. By convention, the node which has the property defined is considered to be part of the parent as opposed to being a normal node entity in an assembly. SUBNODE properties can be used to represent a number of CAD constructs in JT files including reference sets.

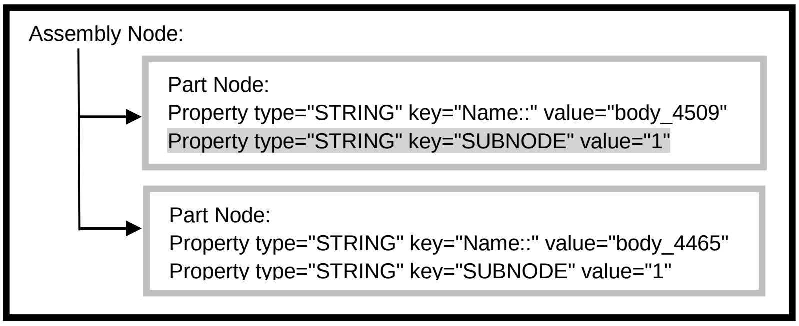


Figure 152 — Assembly node with SUBNODE

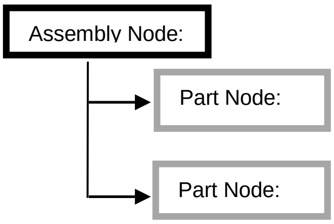


Figure 153 — Assembly node without SUBNODE

JT viewing systems by default do not display the trees structure for a node containing sub node definitions expanded.

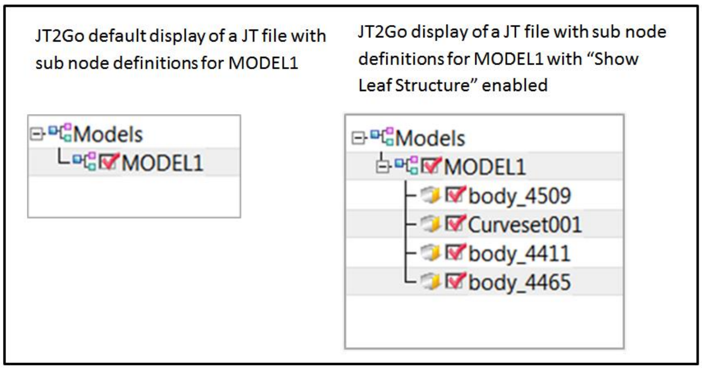


Figure 154 — Displaying Nodes that have SUBNODE properties

The node containing the SUBNODE property can be either a part node or an assembly node.

The value assigned to the SUBNODE property has no specific meaning. Implementers should set the value for the property to a string value of “1”. Parts that contain the SUBNODE property must contain a string property with a name for the part.

Table 79 — SUBNODE Property

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values

JT Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
SUBNODE	Specificies the node as being a sub node of the parent assembly	MbString	<string>	0 or 1

13.8.6.1 Reference Sets and the Reference Set Property

Many CAD systems are able to create a modeled construct whereby a single CAD part can contain a user defined substructure of geometry and PMI. A CAD part constructed this way is said to have reference sets or reference geometry. In JT a CAD part constructed this way, by convention, is referred to as a CAD component.

A JT CAD component is a unique assembly structure whereby the top level part in the assembly is the CAD Component and the part(s) that make up the assembly are sub nodes. When this arrangement exists in a JT file the CAD Component is said to contain Reference Sets. Reference Set definitions are a JT convention made up of assembly nodes, part nodes and properties. The part or assembly nodes defined in the CAD Component assembly are the actual reference set definitions.

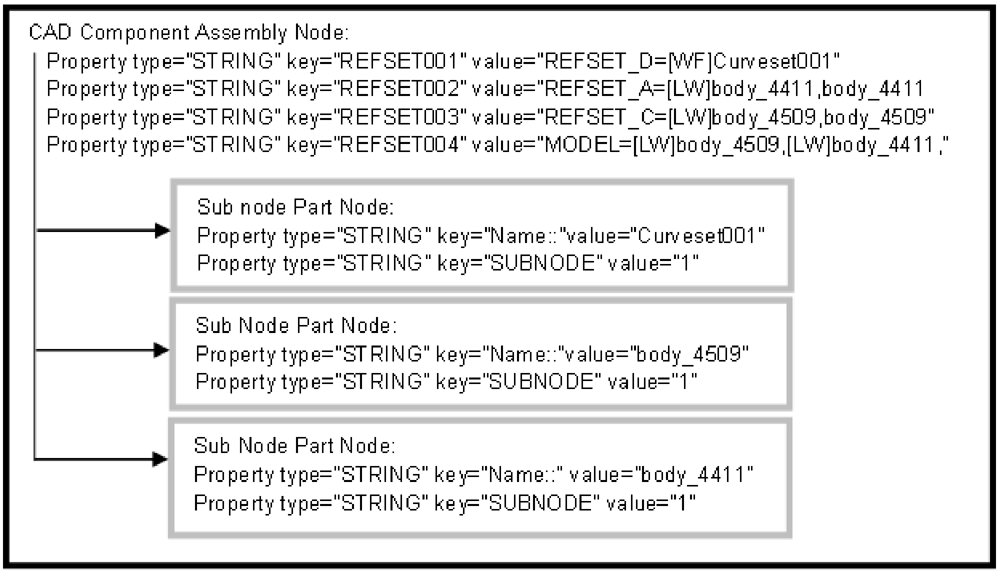


Figure 155 — CAD Component with Reference sets

This representation of the CAD Component structure is achieved through the use of the REFSET<XXX> property in the CAD Component part and the SUBNODE property in the parts that contain the reference set definitions.

The reference set property key has the form REFSETXXX with the XXX being a three digit incrementing number starting at 001. The numbers must be assigned concurrently. Up to 999 references set properties can be assigned for a CAD component. A reference set property is a comma delimited string of part names and hints. These parts can contain geometry such as solid bodies, wireframe or points that represent content relevant to the owning CAD component. To facilitate identification of the reference set content, hint strings can be combined with the part names. There are four reserved hint strings for reference sets; PMI, PT (point cloud), WF (wireframe) and LW (light weight, facet only).

Reference set encoding conventions

- Reference set names are not case sensitive
- The values = , and \ need to be escaped within the values of the property keys if they are in a part name or reference set name

The Reference Set Properties table provides a description of the Reference set property values.

Table 80 — Reference Set Properties

Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
REFSETXXX	Specifies the string of part names that are included in the reference set. The property value defines the reference set name that is displayed in viewing systems i.e.; type="STRING" key="REFSET<XXX>" value="REFSET_A_D=[WF]<ref set name>,[LW]<ref set name>,[PT]<ref set name>,<ref set name>,[PMI]" There are 3 reserved strings that can be added as prefixes to the JT part names included on the reference set string. They are; [PT] : precedes the referenced part name(s) that contain point cloud geometry [WF] : precedes the referenced part name(s) that contain wireframe geometry [LW] : precedes the referenced part name(s) that contain only facet geometry representations. B-Rep is not present. The reference hint string [PMI] is included without part names. When the [PMI] hint is included corresponding PMI entities with the JTTK_MULTICAD_REFSET property may exist. See table 2	MbString	<string>	Comma delimited string of part names and hints

The Properties table, related to the use of Reference Sets, provides a list of additional properties that assist applications with using and displaying reference sets.

Table 81 — Properties related to the use of Reference Sets

Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
REFSET_META	Contains a comma delimited list of aliased reference sets via the following convention; REFSET_META="alias_1=name_1,alias_2=name_2 An aliased reference sets is a reference	MbString	<string>	Comma delimited string

	set definition used to record a group of reference sets that crosses multiple CAD Components.			
JTTK_MULTICAD_REFSET	This property must be set on the PMI. To have PMI visible within a reference it must have a JTTK_MULTICAD_REFSET property defined. The property is a string of Reference Set names that the PMI will be visible in. i.e. type="STRING" key=" JTTK_MULTICAD_REFSET" value="<refset name>, <refset name>"	MbString	<string>	Comma delimited string

The REFSET_CURRENT property is defined in instance nodes that are instances of a CAD Component. The property contains a string value that is the name of a reference set that exists in the CAD Component that has been instanced. Best practice is to use this property to determine the reference set that will become active if the model is set to the "as saved" state.

Table 82 — REFSET_CURRENT property

Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
REFSET_C**U**RRENT	Defines which reference set will be displayed in an instance node i.e. type="STRING" key="REFSET_CURRENT"value =<refset name>	MbString	<string>	Comma delimited string

13.9 LSG Attribute Accumulation Semantics

For applications producing or consuming JT format data, it is important that the JT format semantics of how attributes are meant to be applied and accumulated down the LSG are followed. If not followed, then consistency between the applications in terms of 3D positioning and rendering of LSG model data will not be achieveable.

Although each attribute type defines its own application and accumulation LSG semantics (the details of which can be found in each attribute type sub-section under Attribute Elements), there are some general rules which apply:

Attributes at lower level in the LSG take precedence and replace or accumulate with attributes set at higher levels. When multiple Attributes of the same type are present on a Node, they accumulate in the order they are specified (i.e. from the front of the Attribute list toward the back).

Nodes with no associated attributes inherit those of their parents.

Attributes are inherited only from a node's parents. Thus a given node's attributes do not affect those on the node's siblings.

The root of a partition inherits the attributes in effect at the referring partition node.

Attributes can be marked "final", which terminates accumulation of that attribute type at that marked attribute and propagates the accumulated value at that point to all descendants of the associated node. Descendants can override a "final" attribute using the "force" flag. Note that "force" does not turn OFF "final" – it is simply a one-shot override of "final" for the specific attribute marked as "forcing." Multiple attributes of the same type may be marked as "forcing" and in this case, the last one wins. Both of these flags are OFF by default. An analogy for this "force" and "final" interaction is that "final" is a back-door in the attribute accumulation semantics and that "force" is the doggy-door in the back-door!

13.10 LSG Part Structure

The JT Format Reference does not mandate that a particular node hierarchy be used for modeling physical Parts within a LSG structure. In fact there are many node hierarchies for representing Parts in LSG that will function correctly in most JT enabled applications. Still, there is a convention that most JT translators follow (and some JT enabled applications may assume exists) for modeling Parts within a LSG. The convention is to model each Part within a LSG structure with the following node hierarchy:

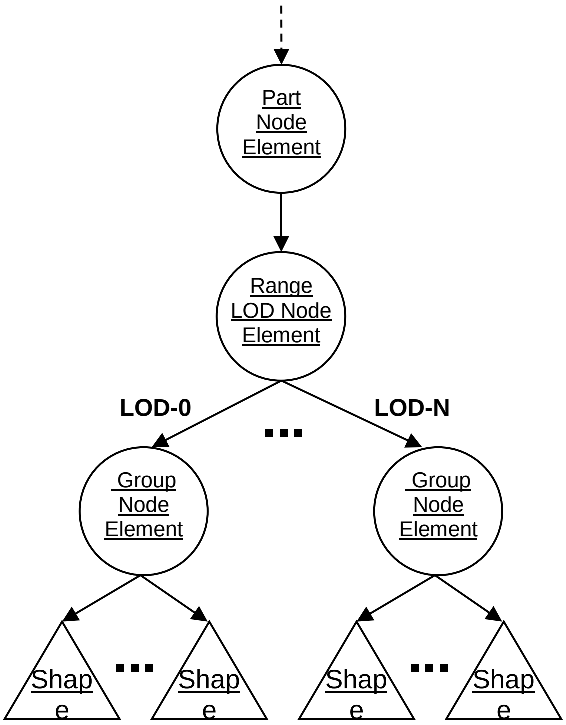


Figure 156 — JT Format Convention for Modeling each Part in LSG

13.11 Range LOD Node Alternative Rep Selection

Best practices suggest that LSG traversers apply the following strategy, at Range LOD Nodes (see Range LOD Node Element), when making alternative representation selection decisions based on Range Limits: The first alternate representation is valid when the world coordinate distance between the centre and the eye point is less than or equal to the first range limit (and when no range limits are specified). The second alternate representation is valid when the distance is greater than the first limit and less than or equal to the second limit, and so on. The last alternate representation is valid for all distances greater than the last specified limit.

13.12 B-Rep Face Group Associations

The original purpose of the face group concept was to provide associativity between B-Rep faces and geometry. Exactly how a B-Rep face associates to a face group number is the topic of this section. An implicit scheme has been chosen for face group associativity, rather than storing some kind of explicit data on either the Vertex Shape LOD Data or the B-Rep. The primary motivation for this implicit scheme is to keep the JT files simple and small; additional association information would not only be redundant, but also wasteful. Tessellators shall exercise this policy when producing Vertex Shape LOD Data from B-Reps, grouping the triangles into face groups according to its rules. Tristrips may not cross face groups. Applications shall be able to count on this policy so that, for example, they can map a picking action back to its corresponding B-Rep face reliably.

JT B-Rep/ULP: In the case of JT B-Rep and ULP reps, the mapping is simple. These Reps have a consistent sequential index origin-0 numbering scheme for their regions, shells, and faces. So the B-Rep faces are simply assigned sequentially to face group by increasing region and shell. For example, suppose we have a JT B-Rep with 2 regions, each with 2 shells, each with 2 faces. The Face Group ⇔ Region/Shell/Face mapping will be as follows:

```
FG0 ⇔ R0 S0 F0
FG1 ⇔ R0 S0 F1
FG2 ⇔ R0 S1 F0
FG3 ⇔ R0 S1 F1
FG4 ⇔ R1 S0 F0
FG5 ⇔ R1 S0 F1
FG6 ⇔ R1 S1 F0
FG7 ⇔ R1 S1 F1
```

XT B-Rep: In the case of XT B-Rep, the mapping is based on an identifier of each XT face that is persisted on disk. The identifier is unique within each XT body, but it is not an index. XT B-Rep maintains a zero-based contiguous index of XT face based on increasing identifier value within the same XT body. In the case when multiple bodies are present in XT B-Rep, face index is assigned sequentially by increasing XT body index. For example, suppose we have a XT B-Rep with 2 bodies, each with 2 faces, then the Face Group to Body/Face mapping will be as follows:

```
FG0 ⇔ B0 F0
FG1 ⇔ B0 F1
FG2 ⇔ B1 F0
FG3 ⇔ B1 F1
```

13.13 Watermark Image

A watermark image can be added to a JT file as a texture. Textures are stored as Attribute Elements in JT. Attribute Elements are placed in the Logical Scene Graph (LSG) as objects associated with nodes. For more information on Attribute Elements, see Attribute Elements.

Use of a JT texture to represent a watermark is implementation de-pendent. When a texture contains specific properties, an application shall display the texture image on top of any other graphics, effectively displaying it as a watermark.

This implementation requires properties that follow a defined convention. These properties alert the application that the texture information should be displayed in a unique way and at which locations on the display screen the texture should appear. Properties on textures are defined as Property Atom Elements. These properties are meta-data objects associated with attributes. Each attribute element in an LSG may hold zero or more property atom elements.

Table 83 — Texture watermark properties

Property Key	Meaning	JT File Data Type	Encoded Data Type	Valid Values
WATERMARK / Watermark (*)	Alerts to the application that the texture containing this property is meant to be displayed as a watermark	String	MbString	Watermar k
LOCATION / Location (*)	Defines a region on the display for the water mark to appear top left = 1, top center = 2, top right = 3, mid left = 4, mid center = 5, mid right = 6, bottom left = 7, bottom center = 8, bottom_right = 9	Integer	Integer	1-9
Y_DPI	DPI for the output image in Y direction	Float	F32 value	
X_DPI	DPI for the output image in X direction	Float	F32 value	

(*) Property key may appear in upper case or camel style.

13.14 State Flags

State Flags are defined within the Base Property Atom Element definition this way;

> State Flags are a collection of flags. The flags are combined using the binary OR operator and store various state information for property atoms. Bits 0 – 7 are freely available for an application to store whatever property atom information desired. All other bits are reserved for future expansion of the file format.

Tests have shown that there are problems with interoperability when Bits 0-7 are freely used. State Flags bits 0-7 shall be defaulted to 0x40000000.

Annex A
Object Type Identifiers

All objects stored in a JT file are classified by type and thus include an object type identifier as part of their persisted data. The data format for these Object Type identifiers is a GUID. These Object Type identifiers are consistent for all objects, of a particular type.

Table A.1 — Object Type Identifiers

GUID	Object Type
0xffffffff, 0xffff, 0xffff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff	Identifier to signal End-Of-Elements.
Types Stored Within LSG Segment (Segment Type = 1)	
0x10dd1035, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Base Node Element
0x10dd101b, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Group Node Element
0x10dd102a, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Instance Node Element
0x10dd102c, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	LOD Node Element
0xce357245, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1	Meta Data Node Element
0xd239e7b6, 0xdd77, 0x4289, 0xa0, 0x7d, 0xb0, 0xee, 0x79, 0xf7, 0x94, 0x94	NULL Shape Node Element
0xce357244, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1	Part Node Element
0x10dd103e, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Partition Node Element
0x10dd104c, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Range LOD Node Element
0x10dd10f3, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Switch Node Element
0x10dd1059, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Base Shape Node Element
0x98134716, 0x0010, 0x0818, 0x19, 0x98, 0x08, 0x00, 0x09, 0x83, 0x5d, 0x5a	Point Set Shape Node Element
0x10dd1048, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Polygon Set Shape Node Element
0x10dd1046, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Polyline Set Shape Node Element
0xe40373c1, 0x1ad9, 0x11d3, 0x9d, 0xaf, 0x0, 0xa0, 0xc9, 0xc7, 0xdd, 0xc2	Primitive Set Shape Node Element
0x10dd1077, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Tri-Strip Set Shape Node Element
0x10dd107f, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Vertex Shape Node Element
0x10dd1001, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Base Attribute Data
0x10dd1014, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Draw Style Attribute Element
0x10dd1083, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Geometric Transform Attribute Element

0x10dd1028, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Infinite Light Attribute Element
0x10dd1096, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Light Set Attribute Element
0x10dd10c4, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Linestyle Attribute Element
0x10dd1030, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Material Attribute Element
0x10dd1045, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Point Light Attribute Element
0x8d57c010, 0xe5cb, 0x11d4, 0x84, 0xe, 0x00, 0xa0, 0xd2, 0x18, 0x2f, 0x9d	Pointstyle Attribute Element
0x10dd1073, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Texture Image Attribute Element
0xaa1b831d, 0x6e47, 0x4fee, 0xa8, 0x65, 0xcd, 0x7e, 0x1f, 0x2f, 0x39, 0xdc	Texture Coordinate Generator Attribute Element
0x10dd1106,0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	PaletteMap Attribute Element
0xa3cfb921, 0xbdeb, 0x48d7, 0xb3, 0x96, 0x8b, 0x8d, 0xe, 0xf4, 0x85, 0xa0	Mapping Plane Element
0x3e70739d, 0x8cb0, 0x41ef, 0x84, 0x5c, 0xa1, 0x98, 0xd4, 0x0, 0x3b, 0x3f	Mapping Cylinder Element
0x72475fd1, 0x2823, 0x4219, 0xa0, 0x6c, 0xd9, 0xe6, 0xe3, 0x9a, 0x45, 0xc1	Mapping Sphere Element
0x92f5b094, 0x6499, 0x4d2d, 0x92, 0xaa, 0x60, 0xd0, 0x5a, 0x44, 0x32, 0xcf	Mapping TriPlanar Element
0x10dd104b, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Base Property Atom Element
0xce357246, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1	Date Property Atom Element
0x10dd102b, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Integer Property Atom Element
0x10dd1019, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Floating Point Property Atom Element
0xe0b05be5, 0xfbbd, 0x11d1, 0xa3, 0xa7, 0x00, 0xaa, 0x00, 0xd1, 0x09, 0x54	Late Loaded Property Atom Element
0x10dd1004, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	JT Object Reference Property Atom Element
0x10dd106e, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	String Property Atom Element
Types Stored Within JT B-Rep Segment (Segment Type = 2)	
0x873a70c0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	JT B-Rep Element
Types Stored Within Meta Data Segment (Segment Type = 4)	
0xce357249, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1	PMI Manager Meta Data Element
0xce357247, 0x38fb, 0x11d1, 0xa5, 0x6, 0x0, 0x60, 0x97, 0xbd, 0xc6, 0xe1	Property Proxy Meta Data Element
Types Stored Within Shape LOD Segment (Segment Type = 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16)	
0x3e637aed, 0x2a89, 0x41f8, 0xa9, 0xfd, 0x55, 0x37, 0x37, 0x3, 0x96, 0x82	Null Shape LOD Element
0x98134716, 0x0011, 0x0818, 0x19, 0x98, 0x08, 0x00, 0x09, 0x83, 0x5d, 0x5a	Point Set Shape LOD Element

0x10dd10a1, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Polyline Set Shape LOD Element
0xe40373c2, 0x1ad9, 0x11d3, 0x9d, 0xaf, 0x0, 0xa0, 0xc9, 0xc7, 0xdd, 0xc2	Primitive Set Shape Element
0x10dd10ab, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Tri-Strip Set Shape LOD Element
0x10dd109f, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Polygon Set LOD Element
0x10dd10b0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Vertex Shape LOD Element
Types Stored Within XT B-Rep Segment (Segment Type = 17)	
0x873a70e0, 0x2ac9, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	XT B-Rep Element
Types Stored Within Wireframe Segment (Segment Type = 18)	
0x873a70d0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97	Wireframe Rep Element
0xf338a4af, 0xd7d2, 0x41c5, 0xbc, 0xf2, 0xc5, 0x5a, 0x88, 0xb2, 0x1e, 0x73	JT ULP Element
Types Stored Within JT LWPA Segment (Segment Type = 24)	
0xd67f8ea8, 0xf524, 0x4879, 0x92, 0x8c, 0x4c, 0x3a, 0x56, 0x1f, 0xb9, 0x3a	JT LWPA Element
Type Stored Within Info Segment (Segment Type = 31)	
0x84c2112a, 0x0001, 0x11e7, 0x80, 0x00, 0xa4, 0x24, 0x9a, 0x27, 0x47, 0x70	JT Info Element
Type Stored Within STT (Segment Type = 32)	
0xca7e6f89, 0x97c8, 0x47f0, 0x9f, 0xca, 0x16, 0x99, 0xc, 0xfb, 0xe2, 0x17	JT STT Element
Type Stored Within STEP-B-Rep Segment (Segment Type = 33)	
0x869c7d53, 0xccb0, 0x451b, 0xb2, 0x3, 0xd1, 0x42, 0x81, 0x56, 0x14, 0x56	STEP B-Rep Element

Annex B
Coding Algorithms – An Implementation

This Appendix provides a sample C++ implementation for the encoding and decoding portion of the various compression CODECs used in the JT format. This sample code is not intended to be fully functional encoder/decoder class implementations, but is instead intended to demonstrate the fundamentals of implementing the encoding/decoding portion of the CODEC algorithms used in the JT format.

B.1 Common classes

The following sub-sections define some general classes used by all the coding algorithms.

B.1.1 CntxEntryBase class

```
//
// Type used to build probability context tables.
// Used by ProbabilityContext class.
//
class CntxEntryBase
{
  public:

    // ---------- Housekeeping ----------
    CntxEntryBase() : _bIsEscape(false), _cCount(-1), _cCumCount(-1) {};
    CntxEntryBase(Bool bIsEsc, Int32 cCount)
        : _bIsEscape(bIsEsc), _cCount(cCount), _cCumCount(-1) {};
    CntxEntryBase(const CntxEntryBase &rhs) { *this = rhs; }
    ~CntxEntryBase() {};
    CntxEntryBase &operator=(const CntxEntryBase &rhs)
      { _bIsEscape = rhs._bIsEscape;
        _cCount    = rhs._cCount;
        _cCumCount = rhs._cCumCount;
        return *this;
      }

    // ---------- Operations Interface ----------
    Bool isEscape() const
      { return _bIsEscape; }
    Int32 operator==(const CntxEntryBase2 &rhs) const
      { return (_iSym == rhs._iSym); };

  public:

    // ---------- Member Variables ----------
    Int32         _cCount;    // Number of occurrences
    Int32         _cCumCount; // Cumulative number of occurrences
    Bool          _bIsEscape; // True if this symbol is the escape symbol
};

template <class ValueType>
class CntxEntry : public CntxEntryBase
{
  public:

    // ---------- Housekeeping ----------
    CntxEntry() : CntxEntryBase(), _val(ValueType()) {};

    CntxEntry( Bool bIsEsc, Int32 cCount, const ValueType &val ):
        CntxEntryBase(bIsEsc, cCount), _val(val) {};
```

```
    CntxEntry( const CntxEntry &rhs ) { *this = rhs; }
    ~CntxEntry() {};
    CntxEntry &operator=(const CntxEntry &rhs)
        { _val = rhs._val;
         CntxEntryBase::operator= (rhs);
         return *this;
        }
    Int32 operator==(const CntxEntry &rhs) const
        { return (_iSym == rhs._iSym); };

  public:

    // ---------- Member Variables ----------
    ValueType       _val;     // Associated value
};
```

B.1.2 ProbContext class

```
//
// Type used to build probability context tables.
// Used by CodecDriver class.
//
template <class ValueType>
class ProbContext
{
  public:
    typedef CntxEntry< ValueType > CntxEntryV;

    // ---------- Housekeeping ----------
    ProbContext();
    ProbContext(const ProbContext &rhs);
    ~ProbContext();
    ProbContext &operator=(const ProbContext &rhs);
    Bool operator==(const ProbContext &rhs) const;
    enum { cMaxCntxCount = 8192 };

    // ---------- Accessor Interface ----------
    Int32 totalCount() const
        { return _cTotalCount; }
    Int32 numEntries() const
        { return _vEntries.length(); }
    Bool getEntry(Int32 iEntry, const CntxEntryBase *&rpEntry) const
        { const CntxEntryV *aEntries = _vEntries.ptr();
          rpEntry = &aEntries[iEntry]; return True; }
    Bool getEntryV(Int32 iEntry, const CntxEntryV *&rpEntry) const
        { const CntxEntryV *aEntries =  _vEntries.ptr();
          rpEntry = &aEntries[iEntry]; return True; }
    Bool getEntryV(Int32 iEntry, CntxEntryV *&rpEntry)
        { CntxEntryV *aEntries = _vEntries.ptr();
          rpEntry = &aEntries[iEntry]; return True; }

    // ---------- Lookup Interface ----------
    Bool lookupValue(const ValueType &rValue, const CntxEntryV *&opCntxEntry)
const;
    Bool lookupEntryByCumCount(Int32 iCount, const CntxEntryV *&opCntxEntry) const;

    // ---------- Reorganizing Interface ----------
    Bool accumulateCounts();
    Bool sortByValue();

  protected:
    Vec< CntxEntryV >  _vEntries;
    Int32              _cTotalCount;
    Int32              _iEscPosCache;
```

```
    static int _compareCounts(const void *pVal1,
                              const void *pVal2,
                              const void *uData);
};

template <class ValueType>
Bool ProbContext::lookupValue( const ValueType   &rValue,
                               const CntxEntryV *&opCntxEntry ) const
{
    // If we do not find the value, then return NULL for the entry
    opCntxEntry = NULL;

    // If the escape position is not cached, sort the context by value
    // and then set it.  Then, we can binary search for values.  We do
    // this because translateValuesToSymbols() will call this method in
    // a tight loop.  Anything is better than linear search!
    ProbContext *pThis = (ProbContext*) this;
    CntxEntryV *pEntries = pThis->_vEntries.ptr();
     Int32 nEntries = _vEntries.length();
    if (_iEscPosCache == -1) {
        // Search for the escape symbol
        Bool bFoundEsc = False;
        for (Int32 i = 0 ; i < nEntries ; i++) {
            if (pEntries[i].isEscape()) {
                // Move the escape symbol to context slot 0
                ::swap(pEntries[0], pEntries[i]);
                bFoundEsc = True;
                break;
            }
        }
        // Sort by value
        if (bFoundEsc) {
            // Sort by value _except_ leave the escape symbol in slot 0
            ::sort(&pEntries[1], nEntries-1, FtorCntxValue<ValueType>());
            pThis->accumulateCounts();
            pThis->_iEscPosCache = 0;
        }
        else {
            pThis->sortByValue();
            pThis->_iEscPosCache = -2;
        }
    }

    // Binary search for rValue!
    Int32 l = (_iEscPosCache == 0),
          h = nEntries - 1,
          m;
    while (l <= h) {
        m = (l + h) >> 1;
        if (pEntries[m]._val == rValue) {
            opCntxEntry = &pEntries[m];
            return True;
        }
        else if (pEntries[m]._val < rValue)
            l = m + 1;
        else
            h = m - 1;
    }

    // If we don't find the value, then we return the position of
    // the escape symbol.
    if (_iEscPosCache >= 0)
        opCntxEntry = &pEntries[_iEscPosCache];

    return True;
}
```

```
template <class ValueType>
Bool ProbContext2::lookupEntryByCumCount(Int32 iCount, const CntxEntryV
*&opCntxEntry ) const
{
    const CntxEntryV *aEntries = _vEntries.ptr();
    const Int32       nEntries = _vEntries.length();

    const Int32       seqSearchLen = 4;
    Int32 ii=0;
    opCntxEntry = NULL;

    // For short lists, do sequential search
    if ( nEntries <= (seqSearchLen*2) ) {
        ii = 0;
        while ((iCount>=(aEntries[ii]._cCumCount + aEntries[ii]._cCount)) &&
               (ii<nEntries))
        {
            ii++;
        }

        if ( ii >= nEntries ) {
            Assert( 0 && "Bad probability table" );
        }
        opCntxEntry = &aEntries[ii];
    }

    // For long lists, do a short sequential searches through most likely
    // elements, then do a binary search through the rest.
    else {
        for (ii=0; ii<seqSearchLen; ii++) {
            if (iCount < (aEntries[ii]._cCumCount + aEntries[ii]._cCount)) {
                opCntxEntry = &aEntries[ii];
                return True;
            }
        }

        Int32 low=ii, high=nEntries-1, mid;
        while(1) {
            if ( high < low ) {
                break;
            }
            mid = low + ((high-low)>>1);

            if ( iCount < aEntries[mid]._cCumCount ) {
                high = mid-1;
                continue;
            }
            if ( iCount >= (aEntries[mid]._cCumCount + aEntries[mid]._cCount) ) {
                low = mid+1;
                continue;
            }

            opCntxEntry = &aEntries[mid];
            return True;
        }
        Assert( 0 && "Bad probability table" );
    }

    return True;
}

template <class ValueType>
Bool ProbContext2::accumulateCounts()
{
    // Check for zero length context
    CntxEntryV *aEntries = _vEntries.ptr();
    Int32       nEntries = _vEntries.length();
```

```
    if ( nEntries == 0 ) {
        _cTotalCount = 0;
        return True;
    }

    // Accumulate counts in _cCumCount for entries 1 and higher
    aEntries[0]._cCumCount = 0;
     Int32 ii;
    for ( ii=1 ; ii<nEntries ; ii++ ) {
        aEntries[ii]._cCumCount = aEntries[ii-1]._cCount + aEntries[ii-
1]._cCumCount;
    }

    // Set the total count for the context
    _cTotalCount = aEntries[ii-1]._cCount + aEntries[ii-1]._cCumCount;

    return True;
}

template <class ValueType>
struct FtorCntxValue
{
   Bool operator () (const CntxEntry<ValueType>& l, const CntxEntry<ValueType>& r)
const
   { return  (l._val < r._val); }
};

template <class ValueType>
Bool ProbContext2::sortByValue()
{
   // Sort the entries in order of values from smallest to largest
   ProbContextV *pThis = (ProbContextV*) this;
   sort( _vEntries.ptr(), (size_t)(_vEntries.length()), FtorCntxValue<ValueType>()
);

   pThis->_iEscPosCache = -1;
   pThis->accumulateCounts();

   return True;
}
```

B.1.3 CodecDriver class

```
//
// A class that deals with the conversions from SYMBOL to VALUE and
// provides end-consumer APIs for using the codecs.
//
template <class ValueType>
class CodecDriver
{
  public:

    // ---------- Internal Types ----------
    typedef enum {
        CodecNull       = 0,    // Null Codec
        CodecBitLength  = 1,    // Bitlength Codec
        CodecArithmetic = 3,    // Arightmetic Codec
        CodecChopper    = 4,    // Chopper Pseudo-codec
        CodecMTF        = 5     // Move-to-front Pseudo-codec
    } CodecType;
    // Type of value predictor
    typedef enum {
            PredLag1       = 0, // Predicts as last values
            PredXor1       = 1, // Predict as last, but use xor instead of subtract
            PredNULL       = 2, // No prediction.
```

```
    } PredictorType;

    static Bool unpackResiduals(const Veci    &rvResidual,
                                Veci          &rvVals,
                                PredictorType  ePredType);
    static Bool unpackResiduals(const Vecu    &rvResidual,
                                Vecu          &rvVals,
                                PredictorType  ePredType);

    static Float64 log2( Float64 x ) { return (log(x) / 0.6931471805599453); }

    //////////////////////////////
    // Convenience Methods
    //////////////////////////////

  protected:
    static Int32 _predictValue( const Int32 *vVal, Int32 iIndex,
                                PredictorType ePredType );
};

Bool CodecDriver::unpackResiduals( const Veci      &rvResidual,
                                   Veci            &rvVals,
                                   PredictorType    ePredType )
{
    const Int32 len = rvResidual.length();
    Int32 iPredicted;
    rvVals.setLength(len);
    Int32 *aVals = rvVals.ptr();
    const Int32 *aResidual   = rvResidual.ptr();
    for ( Int32 i = 0 ; i < len ; i++ ) {
        if (i < 4) {
            // The first four values are just primers
            aVals[i] = aResidual[i];
        } else {
            // Get a predicted value
            iPredicted = _predictValue(rvVals.ptr(), i, ePredType);

            if (ePredType == PredXor1) {
                // Encode the residual as the current value XOR predicted
                aVals[i] = aResidual[i] ^ iPredicted;
            } else  {
                // Encode the residual as the current value plus predicted
                aVals[i] = aResidual[i] + iPredicted;
            }
        }
    }

    return True;
}

Bool
CodecDriver2::unpackResiduals( const Vecu     &rvResidual,
                               Vecu           &rvVals,
                               PredictorType  ePredType )
{
    return unpackResiduals(*((const Veci*)&rvResidual),
                          *((Veci*) &rvVals),
                          ePredType);
}

Int32
CodecDriver2::_predictValue( const Int32*  paVals,
                             Int32         iIndex,
                             PredictorType ePredType )
{
    Int32 iPredicted = 0;
    switch (ePredType) {
```

```
        default:
        case PredLag1:      // Predicts as last value
        case PredXor1:      // Predicts as last value
            iPredicted    = paVals[iIndex-1];
            break;
    }

    return iPredicted;
}
```

B.2 Bitlength decoding class

The following sub-section contains a sample implementation of the decoding portion of the Bitlength CODEC algorithm. A summary technical explanation of the Bitlength CODEC can be found in Bitlength CODEC description.

B.2.1 BitLengthCodec class

```
template <class ValueType>
class BitLengthCodec : public Codec<ValueType>
{
  public:
    typedef Vec< ValueType >               VecValue;
    typedef ProbContext< ValueType >       ProbContextV;
    typedef CodecDataCntx< ValueType >     CodecDataCntxV;

    Bool encode( const VecValue &vValues,
                 VecValue       &ovOOBValues,
                 Vecu           &ovCodeText,
                 Int32          &onBitsCodeText,
                 ProbContextV   *pProbCntx       );
    Bool decode( Int32           nValues,
                 VecValue       &ovOOBValues,
                 const Vecu     &vCodeText,
                 Int32           nBitsCodeText,
                 VecValue       &ovValues,
                 ProbContextV   *pProbCntx       );

  protected:
    Int32 _nBitsInSymbol(Int32 iSymbol) const;
    Bool getNextCodeText (UInt32 &uCodeText, Int32 &nBits);

    Vecu     *_pvCodeText;
    Int32    *_pcCodeTextLen;
    Int32     _iCurCodeText;
    Vecus     _vnValBits;
};

template <class ValueType> void
BitLengthCodec3T<ValueType>::GetSignedBits(Int32 &iOut, UInt32 n)
{
    GetUnsignedBits(*((UInt32*)&iOut),n);
    iOut <<= (32 - n);
    iOut >>= (32 - n);
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::GetUnsignedBits(UInt32 &uOut, UInt32 n)
{
    if (n == 0) uOut = 0;
    else if (_nValBits >= n) {
        uOut = _uVal >> (32 - n);
        _uVal    <<= n;
        _uVal &= (n==32)-1;
```

```
        _nValBits -= n;
        _nBits    += n;
    }
    else {
        Int32 _nLBits = _nValBits;
        uOut = _uVal >> (32 - n);
        _nBits    += _nLBits;
        getNextCodeText (_uVal, _nValBits);
        Int32 _nRBits = (n - _nLBits);
        uOut |= _uVal >> (32 - _nRBits);
        _uVal     <<= _nRBits;
        _uVal &= (_nRBits==32)-1;
        _nValBits -= _nRBits;
        _nBits    += _nRBits;
    }
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::GetUnsignedBits(UInt64 &ulOut, UInt32 n)
{
    UInt32 low32 = 0, high32 = 0;
    GetUnsignedBits(low32, ::min(n, (UInt32)32));
    GetUnsignedBits(high32, ::max((Int32)n-32, (Int32)0));
    UInt64 ulHigh32 = high32;
    ulHigh32 <<= 32;
    ulOut = ulHigh32 | low32;
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::GetSignedBits(Int64 &lOut, UInt32 n)
{
    GetUnsignedBits(*((UInt64*)&lOut),n);
    lOut <<= (64 - n);
    lOut >>= (64 - n);
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::nibblerEmit(UInt32 iVal)
{
    return _nibblerEmit(iVal, ::bitsize(iVal));
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::nibblerEmit(Int32  iVal)
{
    return _nibblerEmit(*(const UInt32*)&(iVal), ::bitsize(iVal));
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::_nibblerEmit(UInt32 uVal, UInt32 nBits)
{
    //uVal &= ((1 << nBits) - 1);     // Eliminate any upper bits
    while (nBits > 0) {
        addCodeText(uVal, cNibbleWidth);
        UInt32 n = min(UInt32(cNibbleWidth), nBits);
        uVal >>= n;
        nBits -= n;
        addCodeText((nBits > 0), 1);    // 1 if more bits, 0 if not
    }
    return;
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::nibblerGet(UInt32 &oiVal)
{
    oiVal = 0;
    UInt32 bMoreBits, uTmp, cNibbles = 0;
```

```
    do {
        GetUnsignedBits(uTmp, cNibbleWidth);
        uTmp <<= cNibbles * UInt32(cNibbleWidth);
        oiVal |= uTmp;
        GetUnsignedBits(bMoreBits, 1);
        cNibbles++;
    } while (bMoreBits);
    return;
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::nibblerGet(Int32 &oiVal)
{
    oiVal = 0;
    UInt32 bMoreBits, uTmp, cNibbles = 0;
    do {
        GetUnsignedBits(uTmp, cNibbleWidth);
        uTmp <<= cNibbles * UInt32(cNibbleWidth);
        oiVal |= uTmp;
        GetUnsignedBits(bMoreBits, 1);
        cNibbles++;
    } while (bMoreBits);
    // Sign-extend the resulting bits
    UInt32 sw = cNibbles * UInt32(cNibbleWidth);
    if (sw < 32) {
        oiVal <<= 32 - sw;
        oiVal >>= 32 - sw;
    }
    return;
}

// Simply write out all the bits for 64 bit
template <class ValueType> void
BitLengthCodec3T<ValueType>::nibblerEmit(Int64  lVal)
{
#if 1
    addCodeText(*(const UInt64*)&lVal, 64);
#else
    _nibblerEmit(*(const UInt64*)&(lVal), ::bitsize(lVal));
#endif
}

template <class ValueType> void
BitLengthCodec3T<ValueType>::nibblerGet(Int64  &olVal)
{
#if 1
    GetUnsignedBits(*(UInt64*)&olVal, 64);
#else
    olVal = 0;
    UInt32 bMoreBits, cNibbles = 0;
    UInt64 uTmp, uTmp64;
    do {
        GetUnsignedBits(uTmp, cNibbleWidth);
        uTmp64 = uTmp;
        uTmp64 <<= cNibbles * UInt32(cNibbleWidth);
        olVal |= uTmp64;
        GetUnsignedBits(bMoreBits, 1);
        cNibbles++;
    } while (bMoreBits);
    // Sign-extend the resulting bits
    UInt32 sw = cNibbles * UInt32(cNibbleWidth);
    if (sw < 64) {
        olVal <<= 64 - sw;
        olVal >>= 64 - sw;
    }
#endif
}
```

```
template <class ValueType> void
BitLengthCodec3T<ValueType>::_nibblerEmit(UInt64 uVal, UInt32 nBits)
{
    while (nBits > 0) {
        addCodeText(uVal, cNibbleWidth);
        UInt32 n = min(UInt32(cNibbleWidth), nBits);
        uVal >>= n;
        nBits -= n;
        addCodeText((nBits > 0), 1);    // 1 if more bits, 0 if not
    }
    return;
}

Template <class ValueType>
Bool BitLengthCodec::encode(const VecValue   &vValues,
                        VecValue         &ovOOBValues,
                        Vecu             &ovCodeText,
                        Int32            &onBitsCodeText,
                        ProbContextV     *)
{
    Int32  i, j, k;
    Int32  iSymbol;               // Symbol value to encode
    Int32  cSymBits = 0,          // Number of bits in iSymbol
           nValues;               // Number of values to encode
    // Initialize output state
    ovOOBValues.setLength(0);
    ovCodeText.setLength(0);
    onBitsCodeText = 0;
    _pvCodeText = &ovCodeText;
    _pcCodeTextLen = &onBitsCodeText;

    // Short circuit for null array of values
    nValues = vValues.length();
    if (nValues <= 0)
        return True;

    _vnValBits.setLength(nValues);
    UInt16 *paiSymBits = _vnValBits.ptr();
    const ValueType *paiValues = vValues.ptr();
    // Find the minimum value and compute how many bits each value takes
    ValueType iMinSymbol = Limits<ValueType>::maxValue();
    ValueType iMaxSymbol = Limits<ValueType>::maxNegValue();
    Float64 fMean = 0.0;
    for (i = 0 ; i < nValues ; i++) {
        iMinSymbol = ::min(iMinSymbol, paiValues[i]);
        iMaxSymbol = ::max(iMaxSymbol, paiValues[i]);
        fMean += Float64(paiValues[i]);
    }
    fMean /= nValues;
    ValueType iMean = Int32(fMean);
    for (i = 0 ; i < nValues ; i++) {
        paiSymBits[i] = bitsize(paiValues[i] - iMean);
    }
    // A "block" is: 3 bits of number of bits (repeats while value is either 3 or -
4 for
larger width changes)
    //              4 bits of block length
    Int32 cBlkLenBits = 4; // Number of bits used to express a block length. 0
means 0.
    Int32 cBlkValBits = 4; // *Delta* number of bits to express the current field
width
    Int32 cBlkHdrBits = cBlkLenBits + cBlkValBits;
    Bool bMerged;
    // Block-forming: Merge Down/Up blocks
    do {
```

```
bMerged = JtFalse;
Int32 iPrevRunPos = 0,
      iCurRunPos  = 0,
      iNextRunPos = 0;
while (iCurRunPos < nValues - 1) {
    // Advance the next run
    iNextRunPos++;
    while (iNextRunPos < nValues - 1 &&
           paiSymBits[iNextRunPos] == paiSymBits[iNextRunPos-1])
    {
        iNextRunPos++;
    }
    if (iNextRunPos >= nValues)
        break;
    Int32  wab = (iCurRunPos - iPrevRunPos),
           wbc = (iNextRunPos - iCurRunPos);
    if (wab == 0) {
        iCurRunPos = iNextRunPos;
        continue;
    }
    else if (wbc == 0) {
        continue;
    }
    // If we've bitten off more than one block's worth of data
    // we must start a new block.  Length 0 is allowed because
    // we may need to insert multiple consecutive block headers
    // in order to change the field width more bits than can be
    // represented in cBlkValBits.
    if (wab > (1 << cBlkLenBits)) {
        iPrevRunPos += (1 << cBlkLenBits);
    }
    else if (wab == (1 << cBlkLenBits)) {
        iPrevRunPos = iCurRunPos;
        iCurRunPos  = iNextRunPos;
        continue;
    }
    UInt16 &ua = paiSymBits[iPrevRunPos],
           &ub = paiSymBits[iCurRunPos],
           &uc = paiSymBits[iNextRunPos];
    // If the runs go "down-up"
    if (ua > ub && ub < uc) {
        // Test if we should increase ub to the lesser of ua and uc
        if (ua < uc) {
            // Test if we should increase ub to ua
            if (wbc * (ua - ub) <= cBlkHdrBits) {
                for (j = iCurRunPos ; j < iNextRunPos ; j++)
                    paiSymBits[j] = ua;
                iCurRunPos = iNextRunPos;
                continue;
            }
        }
        else if (ua > uc) {
            // Test if we should increase ub to uc
            if (wbc * (uc - ub) <= cBlkHdrBits) {
                for (j = iCurRunPos ; j < iNextRunPos ; j++)
                    paiSymBits[j] = uc;
                iNextRunPos = iCurRunPos;
                bMerged = True;
                continue;
            }
        }
        else {  // ua == uc
            // Test if we should increase ub to ua/uc
            if (wbc * (ua - ub) <= 2 * cBlkHdrBits) {
                for (j = iCurRunPos ; j < iNextRunPos ; j++)
                    paiSymBits[j] = ua;
                iCurRunPos  = iPrevRunPos;
```

```
                        iNextRunPos = iPrevRunPos;
                        bMerged = True;
                        continue;
                    }
                }
            }

            // Shift down the runs
            iPrevRunPos = iCurRunPos;
            iCurRunPos  = iNextRunPos;
        }
    } while (bMerged);
    // Block forming: Merge down/down and up/up runs
    do {
        bMerged = JtFalse;
        Int32 iPrevRunPos = 0,
              iCurRunPos  = 0,
              iNextRunPos = 0;
        while (iCurRunPos < nValues - 1) {
            // Advance the next run
            iNextRunPos++;
            while (iNextRunPos < nValues - 1 &&
                   paiSymBits[iNextRunPos] == paiSymBits[iNextRunPos-1])
            {
                iNextRunPos++;
            }
            if (iNextRunPos >= nValues)
                break;
            Int32  wab = (iCurRunPos - iPrevRunPos),
                   wbc = (iNextRunPos - iCurRunPos);
            if (wab == 0) {
                iCurRunPos = iNextRunPos;
                continue;
            }
            else if (wbc == 0) {
                continue;
            }
            // If we've bitten off more than one block's worth of data
            // we must start a new block.  Length 0 is allowed because
            // we may need to insert multiple consecutive block headers
            // in order to change the field width more bits than can be
            // represented in cBlkValBits.
            if (wab > (1 << cBlkLenBits)) {
                iPrevRunPos += (1 << cBlkLenBits);
            }
            else if (wab == (1 << cBlkLenBits)) {
                iPrevRunPos = iCurRunPos;
                iCurRunPos  = iNextRunPos;
                continue;
            }
            UInt16 &ua = paiSymBits[iPrevRunPos],
                   &ub = paiSymBits[iCurRunPos],
                   &uc = paiSymBits[iNextRunPos];
            // If the runs go "up-up"
            if (ua < ub && ub < uc) {
                // Test if we should increase ua to ub
                if (wab * (ub - ua) < cBlkHdrBits) {
                    for (j = iPrevRunPos ; j < iCurRunPos ; j++)
                        paiSymBits[j] = ub;
                    iCurRunPos = iNextRunPos;
                    bMerged = True;
                    continue;
                }
                // Test if we should increase ub to uc
                else if (wbc * (uc - ub) < cBlkHdrBits) {
                    for (j = iCurRunPos ; j < iNextRunPos ; j++)
                        paiSymBits[j] = uc;
```

```
                    iNextRunPos = iCurRunPos;
                    bMerged = True;
                    continue;
                }
            }
            // If the runs go "down-down"
            else if (ua > ub && ub > uc) {
                // Test if we should increase ub to ua
                if (wbc * (ua - ub) < cBlkHdrBits) {
                    for (j = iCurRunPos ; j < iNextRunPos ; j++)
                        paiSymBits[j] = ua;
                    iCurRunPos = iNextRunPos;
                    continue;
                }
            }

            // Shift down the runs
            iPrevRunPos = iCurRunPos;
            iCurRunPos  = iNextRunPos;
        }
    } while (bMerged);

    // Compute the total bits
    UInt32 cMaxBlkLen = (1 << cBlkLenBits) – 1;
    UInt32 iLastRunPos = 0;
    Int32 cTotalBits = paiSymBits[0] + cBlkHdrBits;
    for (UInt32 iCurRunPos = 1 ; iCurRunPos < nValues ; iCurRunPos++) {
        cTotalBits += paiSymBits[iCurRunPos];
        if (paiSymBits[iCurRunPos] != paiSymBits[iCurRunPos -1]) {
            UInt32 cNumBlks = ((iCurRunPos - iLastRunPos) + (cMaxBlkLen-1)) /
cMaxBlkLen;
            cTotalBits += cNumBlks * cBlkHdrBits;
            iLastRunPos = iCurRunPos;
        }
    }
    UInt32 cValSpanBits = bitsize(UInt32(iMaxSymbol – iMinSymbol));
    UInt32 cFixedWidBits = nValues * cValSpanBits + (13+2*(cValSpanBits+1));

    /////
    // If the fixed-width total bits are better, then write out the values
    // in a single fixed-width format.
    /////
    if (cFixedWidBits < cTotalBits) {
        // Write the fixed-width tag
        addCodeText(0, 1);

        // Write the min and max symbols into the stream
        nibblerEmit (iMinSymbol);
        nibblerEmit (iMaxSymbol);

        // Iterate over the remaining symbols
        UInt32 uCodeText = 0;
        for (Int32 i = 0; i < nValues; i++ ) {

            // Get the next symbol
            iSymbol = paiValues[i];

            // Write it
            uCodeText = iSymbol - iMinSymbol;
            addCodeText(uCodeText, cValSpanBits);
        }
    }
    /////
    // Otherwise, encode with variable-length fields
    /////
    else {
        // Write the variable-width tag
```

```
        addCodeText(1, 1);

        // Write out the mean value
        nibblerEmit(iMean);

        // Set the initial field-width
        Int32 cMaxFieldDecr = -(1 << (cBlkValBits - 1)),       // -ve number
              cMaxFieldIncr =  (1 << (cBlkValBits - 1)) - 1;   // +ve number
        Int32 cCurFieldWidth = 0;
        Int32 cTargFieldWidth;
        for (Int32 ii = 0 ; ii < nValues ;) {
            // Adjust the current field width to the target field width
            cTargFieldWidth = paiSymBits[ii];
            {
                if (cCurFieldWidth <= cTargFieldWidth) {
                    while (cTargFieldWidth - cCurFieldWidth >= cMaxFieldIncr) {
                        addCodeText(cMaxFieldIncr, cBlkValBits);
                        cCurFieldWidth += cMaxFieldIncr;
                    }
                }
                else {
                    while (cTargFieldWidth - cCurFieldWidth <= cMaxFieldDecr) {
                        addCodeText(cMaxFieldDecr, cBlkValBits);
                        cCurFieldWidth += cMaxFieldDecr;
                    }
                }
                addCodeText(cTargFieldWidth - cCurFieldWidth, cBlkValBits);
                cCurFieldWidth = cTargFieldWidth;
            }

            // Write out the run length
            for (j = ii+1 ; j < ii + (1 << cBlkLenBits) - 1 && j < nValues ; j++)
                if (paiSymBits[ii] != paiSymBits[j])
                    break;
            addCodeText(j - ii, cBlkLenBits);

            // Write out the data bits for the run
            for (k = ii ; k < j ; k++)
                addCodeText(paiValues[k] - iMean, cCurFieldWidth);

            // Advance to the end of the run
            ii = j;
        }
    }

    return True;
}

template <class ValueType>
Bool BitLengthCodec2::decode( Int32           nValues,
                              const VecValue &,
                              const Vecu     &vCodeText,
                              Int32           nBitsCodeText,
                              VecValue       &ovValues,
                              ProbContextV   *)
{
    Int32  nTotalBits = 0;    // Total number of codetext bits expected
    ValueType iSymbol;        // Decoded symbol value
    Int32  cNumCurBits = 0;   // Current field width in bits
    ValueType iMinSymbol = 0; // The minimum symbol value.  Used as bias.
    ValueType iMaxSymbol = 0; // The maximum symbol value.  Used as bias.
    Int32  nSyms = 0;         // Number of symbols read so far
    ValueType *paiValues;     // Pointer into ovValues where we write decoded
values

    // Get codetext from the driver and loop over it until it's gone!
    ovValues.setLength(nValues);
```

```
    paiValues = ovValues.ptr();

    _iCurCodeText = 0;
    _pvCodeText = (Vecu*) &vCodeText;
    _pcCodeTextLen = &nBitsCodeText;

    /////
    // If the fixed-width total bits are better, then write out the values
    // in a single fixed-width format.
    /////
    // Read the variable-width tag
    Int32 iTmp;
    GetUnsignedBits(iTmp, 1); // 0 = Fixed-width, 1 = Variable width
    if (iTmp == 0) {
        // Read the min and max symbols from the stream
        nibblerGet(iMinSymbol);
        nibblerGet(iMaxSymbol);
        cNumCurBits = bitsize(UInt32(iMaxSymbol – iMinSymbol));

        // Read each fixed-width field and output the value
        while (nBits < nTotalBits || nSyms < nValues) {
            GetUnsignedBits(iSymbol, cNumCurBits);
            iSymbol += iMinSymbol;
            *paiValues++ = iSymbol;
            nSyms++;
        }
    }
    /////
    // Otherwise, encode with variable-length fields
    /////
    else {
        // Write out the mean value
        ValueType iMean;
        nibblerGet(iMean);

        // Set the initial field-width
        Int32 cMaxFieldDecr = -(1 << (cBlkValBits - 1)),       // -ve number
              cMaxFieldIncr =  (1 << (cBlkValBits - 1)) - 1;   // +ve number
        UInt32 cCurFieldWidth = 0, cRunLen, k;
        Int32 cDeltaFieldWidth;
        ValueType iTmp;
        for (Int32 ii = 0 ; ii < nValues ;) {
            // Adjust the current field width to the target field width
            do {
                GetSignedBits(cDeltaFieldWidth, cBlkValBits);
                cCurFieldWidth += cDeltaFieldWidth;
            } while (cDeltaFieldWidth == cMaxFieldDecr || cDeltaFieldWidth ==
cMaxFieldIncr);

            // Read in the run length
            GetUnsignedBits(cRunLen, cBlkLenBits);

            // Read in the data bits for the run
            for (k = ii ; k < ii + cRunLen ; k++) {
                GetSignedBits(iTmp, cCurFieldWidth);
                *paiValues++ = iTmp + iMean;
            }

            // Advance to the end of the run
            ii += cRunLen;
        }
    }

    // Assert that we have consumed exactly all of the bits
    Assert(nValBits == 0);
    Assert(uVal == 0);
```

```
    return True;
}

// Number of bits necessary to encode a SIGNED integer
UInt32 bitsize(Int32 x) const
{
    x = x ^ (x >> 31);
    return 33 - nlz(UInt32(x));
}

// Number of bits necessary to encode an UNsigned integer
UInt32 bitsize(UInt32 x) const
{
    return 32 - nlz(x);
}

// Number of Leading Zeros
UInt32 nlz(UInt32 x)
{
    x = x | (x >> 1);
    x = x | (x >> 2);
    x = x | (x >> 4);
    x = x | (x >> 8);
    x = x | (x >> 16);
    return popcnt(~x);
}

// Population count - # of 1 bits in x
UInt32 popcnt(UInt32 x)
{
    x = x - ((x >> 1) & 0x55555555);
    x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
    x = (x + (x >> 4)) & 0x0f0f0f0f;
    x = x + (x >> 8);
    x = x + (x >> 16);
    return x & 0x3f;
}

Bool BitLengthCodec2::getNextCodeText (UInt32 &uCodeText, Int32 &nBits)
{
    uCodeText = _pvCodeText->value(_iCurCodeText);
    nBits = ::min(32, *_pcCodeTextLen - 32 * _iCurCodeText);
    _iCurCodeText++;
    return True;
}
```

B.3 Arithmetic decoding classes

The following sub-sections contain a sample implementation of the decoding portion of the Arithmetic CODEC algorithm. A summary technical explanation of the Arithmetic CODEC can be found in the12.2.3 Arithmetic CODEC description.

B.3.1 ArithmeticCodec class

```
template <class ValueType>
class ArithmeticCodec: public Codec
{
  public:
    Bool encode( const VecValue &vValues,
                 VecValue       &ovOOBValues,
                 Vecu           &ovCodeText,
                 Int32          &onBitsCodeText,
                 ProbContextV   *pProbCntx       );
    Bool decode( Int32           nValues,
                 VecValue       &ovOOBValues,
```

```
                 const Vecu      &vCodeText,
                 Int32            nBitsCodeText,
                 VecValue        &ovValues,
                 ProbContextV    *pProbCntx       );

  protected:
    Bool _encodeSymbol(UInt16 uLowCt, UInt16 uHighCt, UInt16 uScale );
    Bool _flushEncoder();
    Bool _removeSymbolFromStream( UInt16 uLowCt, UInt16 uHighCt, UInt16 uScale );
    Bool _flushDecoder();

    Bool getNextCodeText (UInt32 &uCodeText, Int32 &nBits);

    UInt16  _code;              // Present input code value, for decoding only
    UInt16  _low;               // Start of the current code range
    UInt16  _high;              // End of the current code range
    Int32   _nUnderflowBits;    // Number of underflow bits pending

    Vecu   *_pvCodeText;
    Int32  *_pcCodeTextLen;
    Int32   _iCurCodeText;

    UInt32  _uBitBuff;
    Int32   _nBitBuff;
};

// Reads a bit and places it into ouBit
#define ReadBit(ouBit)                               \
    if (_nBitBuff==0) {                              \
        getNextCodeText(_uBitBuff, _nBitBuff);  \
    }                                                \
    ouBit = (_uBitBuff >> 31);                       \
    _uBitBuff <<= 1;                                 \
    _nBitBuff--;

// Reads a bit and ORs it into bit 0 of ouBit
#define ReadBit0(ouBit)                              \
    if (_nBitBuff==0) {                              \
        getNextCodeText(_uBitBuff, _nBitBuff);  \
    }                                                \
    ouBit |= (_uBitBuff >> 31);                      \
    _uBitBuff <<= 1;                                 \
    _nBitBuff--;
// Writes bit 0 of uBit
#define WriteBit(uBit)                               \
    if (_nBitBuff==32) {                             \
        addCodeText(_uBitBuff, 32);                  \
        _uBitBuff = _nBitBuff = 0;                   \
    }                                                \
    _uBitBuff <<= 1;                                 \
    _uBitBuff  |= (UInt32(uBit) & 0x1);              \
    _nBitBuff++;

Bool ArithmeticCodec::encode(const VecValue &vValues,
                             Veci           &ovOOBValues,
                             Vecu           &ovCodeText,
                             Int32          &onBitsCodeText,
                             ProbContextV   *pProbCntx)
{
    // Initialize output state
    ovOOBValues.setLength(0);
    ovCodeText.setLength(0);
    onBitsCodeText = 0;
    _pvCodeText = &ovCodeText;
    _pcCodeTextLen = &onBitsCodeText;

    // Initialize the encoder state
```

```
    _low  = 0x0000;
    _high = 0xffff;
    _nUnderflowBits = 0;

    // Prime the bit buffer
    _uBitBuff = 0;
    _nBitBuff = 0;

    const ValueType *paiValues = vValues.ptr();
    const CntxEntry *pEntry;
    Int32 nValues = vValues.length();
    Int32 cTotalCount = pProbCntx->totalCount();
    for (Int32 i = 0; i < nValues; i++ ) {
        // Look up the value in the prob context
        pProbCntx->lookupValue(paiValues[i], pEntry );

        // If this is not the null context, then we emit an escape symbol,
        // move the context it specifies, and restart the tranlation of
        // the same value.  Thus, a value may emit more than one symbol.
        if (pEntry->isEscape()) {
            ovOOBValues.append(paiValues[i]);
        }

        _encodeSymbol(pEntry->_cCumCount,
                      pEntry->_cCumCount + pEntry->_cCount,
                      cTotalCount));
    }

    _flushEncoder();

    return True;
}

Bool ArithmeticCodec2::_encodeSymbol(UInt16 uLowCt, UInt16 uHighCt, UInt16 uScale )
{
    // These three lines rescale _high and _low for the new symbol.
    UInt32 uRange = UInt32(_high - _low) + 1;
    _high  = _low + (uRange * uHighCt) / uScale - 1;
    _low   = _low + (uRange * uLowCt ) / uScale;

    // This loop turns out new bits until _high and _low are far enough
    // apart to have stabilized.
    for (;;) {
        // If this test passes, it means that the most signif digits match,
        // and can be sent to the output stream.
        if ( (_high & 0x8000) == (_low & 0x8000) )
        {
            // Flush the bit buff if the MSB and underflow bits
            // won't all fit in what's left
            if (1+_nUnderflowBits > 32 - _nBitBuff) {
                addCodeText(_uBitBuff, _nBitBuff);
                _uBitBuff = 0;
                _nBitBuff = 0;
            }
            // Write the MSB and all uflow bits at once
            if (1+_nUnderflowBits <= 32 - _nBitBuff) {
                _uBitBuff <<= (1 + _nUnderflowBits);
                _uBitBuff |= (1 << _nUnderflowBits)
                          + (Int32(Int16(~_high)) >> 15);
                _nBitBuff += 1 + _nUnderflowBits;
                _nUnderflowBits = 0;
            }
            else {
                // We're writing more than 32 bits!
                _uBitBuff = (1 << 31)
                          + (Int32(Int16(~_high)) >> 15);
                addCodeText(_uBitBuff, 32);
```

```
                _nBitBuff = 0;
                _nUnderflowBits -= 31;
                // Emit the rest of the underflow bits
                _uBitBuff = (_uBitBuff << 1) | (_uBitBuff & 1);
                while (_nUnderflowBits >= 32) {
                    addCodeText(_uBitBuff, 32);
                    _nUnderflowBits -= 32;
                }
                addCodeText(_uBitBuff, _nUnderflowBits);
                _nUnderflowBits = 0;
                _uBitBuff = 0;
            }
        }

        // If this test passes, the numbers are in danger of underflow, because
        // the most sigif digits don't match, and the 2nd digits are just one
apart.
        //
        // _low =  01... and _high = 10...
        else if ( ( _low & 0x4000 ) && !( _high & 0x4000 ))
        {
            _nUnderflowBits++;
            _low  &= 0x3fff;
            _high |= 0x4000;
        }
        else
            break;

        //Shift all bits left.  Move 0 into _low and 1 into _high.
        _low  <<= 1;
        _high <<= 1;
        _high |=  1;
    }

    return True;
}

Bool ArithmeticCodec2::_flushEncoder()
{
    // Write out some underflow bits and misc.
    WriteBit((_low & 0x4000)>>14)
    _nUnderflowBits++;
    while (_nUnderflowBits-- > 0)
        WriteBit((~_low & 0x4000)>>14)

    //Need 16 zeros at the end, for this decoding algorithm
    UInt32 zeroBit = 0x0000;
    for (Int32 ii=0; ii<16; ii++) {
        WriteBit(zeroBit)
    }

    // Flush out the local buffer
    addCodeText(_uBitBuff, _nBitBuff);

    return True;
}

template <class ValueType>
Bool ArithmeticCodec2::decode( Int32          nValues,
                               const VecValue &vOOBValues,
                               const Vecu     &vCodeText,
                               Int32           nBitsCodeText,
                               Veci           &ovValues,
                               ProbContextV   *pProbCntx     )
{
    ovValues.setLength(0);
    const ValueType *paiOOBValues = vOOBValues.ptr();
```

```
    ovValues.setLength(nValues);
    ValueType *paiValues = ovValues.ptr();
    Int32 cSymbolsCurrCtx = pProbCntx->totalCount();
    const CntxEntryV   *pCntxEntry = 0;

    // Initialize the arithmetic decoder state
    _iCurCodeText = 0;
    _pvCodeText = (Vecu*) &vCodeText;
    _pcCodeTextLen = &nBitsCodeText;
    getNextCodeText(_uBitBuff, _nBitBuff);
    _low  = 0;
    _high = 0xffff;
    _code = (_uBitBuff >> 16);
    _uBitBuff <<= 16;
    _nBitBuff -= 16;

    // Decode each symbol
    for (Int32 i = 0 ; i < nValues ; i++) {
        // Scale the current "code" into the range of counts presented by
        // the probcontext so we can look up the code.
        UInt16 rescaledCode = (((UInt32)(_code - _low) + 1) *
(UInt32)cSymbolsCurrCtx -
1)
             / ((UInt32)(_high - _low) + 1);
        pProbCntx->lookupEntryByCumCount( (Int32)rescaledCode, pCntxEntry );

        // Emit the value corresponding to the symbol we just decoded
        if (!pCntxEntry->isEscape())
            *paiValues++ = pCntxEntry->_val;
        else
            *paiValues++ = *paiOOBValues++;

        // Set up the symbol's range and adjust the decoder state
        // to "remove" it.
        _removeSymbolFromStream( pCntxEntry->_cCumCount,
                                 pCntxEntry->_cCumCount + pCntxEntry->_cCount,
                                 cSymbolsCurrCtx );
    }

    _flushDecoder();

    return True;
}

Bool ArithmeticCodec2::_flushDecoder()
{
    UInt32 dummyBit;
    ReadBit(dummyBit)
    ReadBit(dummyBit)

    Assert( _uBitBuff == 0 );
    _nBitBuff = 0;

    return True;
}

Bool ArithmeticCodec2::_removeSymbolFromStream( UInt16 uLowCt, UInt16 uHighCt,
UInt16
uScale )
{
    // First, the range is expanded to account for the symbol removal.
    UInt32 uRange = UInt32(_high - _low)  + 1;
    _high  = _low + (UInt32)((uRange * uHighCt) / uScale - 1);
    _low   = _low + (UInt32)((uRange * uLowCt ) / uScale);

    //Next, any possible bits are shipped out.
    for (;;) {
```

```
        // If the most signif digits match, the bits will be shifted out.
        if (UInt16(~(_high ^ _low)) >> 15){
        }
        // Else, if underflow is threatening, shift out the 2nd most signif digit.
        //else if ((_low & 0x4000) && !(_high & 0x4000))
        // If high=10xx and low=01xx
        else if (((_low >> 14) == 1) & ((_high >> 14) == 2)) {
            _code ^= 0x4000;
            _low  &= 0x3fff;
            _high |= 0x4000;
        }
        // Otherwise, nothing can be shifted out, so return.
        else {
            return True;
        }

        _low  <<= 1;
        _high <<= 1;
        _high |=  1;
        _code <<= 1;

        ReadBit0( _code )
    }
}
```

B.4 Deering Normal decoding classes

The following sub-sections contain a sample implementation of the decoding portion of the Deering Normal CODEC algorithm. A summary technical explanation of the Deering Normal CODEC can be found in Deering Normal CODEC description.

B.4.1 DeeringNormalLookupTable class

The DeeringNormalLookupTable class represents a lookup table used by the DeeringNormalCodec class for faster conversion from the compressed normal representation to the standard 3-float representation. The tables hold precomputed results of the trig functions called during conversion.

```
class DeeringNormalLookupTable
{
public:
   DeeringNormalLookupTable();

   // Lookup and return the result of converting iTheta and iPsi to
   // real angles and taking the sine and cosine of both.  This gives
   // a slight speedup for normal decoding.
   Bool lookupThetaPsi(Int32 iTheta,
                      Int32 iPsi,
                      UInt32 numberBits,
                      Float32 outCosTheta,
                      Float32 outSinTheta,
                      Float32 outCosPsi,
                      Float32 outSinPsi );

   UInt32 numBitsPerAngle() {return nBits;}

private:
   UInt32 nBits;
   Vector vCosTheta;
   Vector vSinTheta;
   Vector vCosPsi;
   Vector vSinPsi;
};

DeeringNormalLookupTable::DeeringNormalLookupTable()
{
```

```
    UInt32 numberbits = 8;
    nBits = min(numberbits, (UInt32)31);

    Int32 tableSize = (1 << nBits);

    vCosTheta.setLength(tableSize+1);
    vSinTheta.setLength(tableSize+1);
    vCosPsi.setLength(tableSize+1);
    vSinPsi.setLength(tableSize+1);

    Float32 fPsiMax = 0.615479709;
    Float32 fTableSize = (Float32)tableSize;

    for( Int32 ii = 0; ii <= tableSize; ii++ )
    {
    Float32 fTheta =
        asin(tan(fPsiMax * Float32(tableSize - ii) /  fTableSize));

        Float32 fPsi  = fPsiMax * (((Float32)ii) / fTableSize);
        vCosTheta[ii] = cos(fTheta);
        vSinTheta[ii] = sin(fTheta);
        vCosPsi[ii]   = cos(fPsi);
        vSinPsi[ii]   = sin(fPsi);
    }
}

Bool DeeringNormalLookupTable::lookupThetaPsi(Int32 iTheta,
                                                  Int32 iPsi,
                                                  UInt32 numberBits,
     Float32 outCosTheta,
     Float32 outSinTheta,
     Float32 outCosPsi,
     Float32 outSinPsi)
{
    Int32 offset = nBits - numberBits;

    outCosTheta = vCosTheta[iTheta << offset];
    outSinTheta = vSinTheta[iTheta << offset];
    outCosPsi   = vCosPsi[iPsi << offset];
    outSinPsi   = vSinPsi[iPsi << offset];

    return True;
}
```

B.4.2 DeeringNormalCodec class

The DeeringNormalCodec class converts a normal vector to and from the standard 3-float representation and a lower-precision representation. The precision can be adjusted using the nbits parameter.

```
class DeeringNormalCodec
{
public:
    DeeringNormalCodec(Int32 numberbits = 6)
    {
        numBits = numberbits;
    }

    // Converts a compressed normal into a vector.
    Bool convertCodeToVec(UInt32 code, Vector& outVec);

    // Converts a compressed normal into a vector.
    Bool convertCodeToVec (UInt32 iSextant,
                           UInt32 iOctant,
                           UInt32 iTheta,
                           UInt32 iPsi,
```

```
                         Vector& outVec);

   // Separates an encoded normal into its 4 pieces
   Bool unpackCode (UInt32  code,
                    UInt32& outSextant,
                    UInt32& outOctant,
                    UInt32& outTheta,
                    UInt32& outPsi );

   private:
      Int32 numBits;
}

Bool DeeringNormalCodec::convertCodeToVec(UInt32 code, Vector& outVec)
{
   UInt32 s=0, o=0, t=0, p=0;
   unpackCode(code, s, o, t, p);
   convertCodeToVec(s, o, t, p, outVec);
   return True;
}

Bool DeeringNormalCode::convertCodeToVec(UInt32 iSextant,
                                        UInt32 iOctant,
                                        UInt32 iTheta,
                                        UInt32 iPsi,
                                        Vector& outVec)
{
   // Size of code = 6+2*numBits, and max code size is 32 bits,
   // so numBits shall be <= 13.

   // Code layout: [sextant:3][octant:3][theta:numBits][psi:numBits]

   outVec.setValues(0,0,0);
   Float32 fPsiMax = 0.615479709;

   UInt32  iBitRange = 1<<numBits;
   Float32 fBitRange = Float32(iBitRange);

   // For sextants 1, 3, and 5, iTheta needs to be incremented
   iTheta += (iSextant & 1);

   Float32 fCosTheta, fSinTheta, fCosPsi, fSinPsi;

   DeeringNormalLookupTable LookupTable;

   if( (LookupTable.numBitsPerAngle() < (UInt32)numBits) ||
       !LookupTable.lookupThetaPsi(iTheta, iPsi, numBits,
                                   fCosTheta, fSinTheta,
                                   fCosPsi, fSinPsi) )
   {
      Float32 fTheta = asin(tan(fPsiMax * Float32(iBitRange - iTheta) /
                                fBitRange));

      Float32 fPsi = fPsiMax * (iPsi / fBitRange);
      fCosTheta = cos(fTheta);
      fSinTheta = sin(fTheta);
      fCosPsi   = cos(fPsi);
      fSinPsi   = sin(fPsi);
   }

   Float32 x,y,z;
   Float32 xx = x = fCosTheta * fCosPsi;
   Float32 yy = y = fSinPsi;
   Float32 zz = z = fSinTheta * fCosPsi;

   //Change coordinates based on the sextant
   switch( iSextant )
```

```
    {
        case 0:     // No op
            break;

        case 1:     // Mirror about x=z plane
            z = xx;
            x = zz;
            break;

        case 2:     // Rotate CW
            z = xx;
            x = yy;
            y = zz;
            break;

        case 3:     // Mirror about x=y plane
            y = xx;
            x = yy;
            break;

        case 4:     // Rotate CCW
            y = xx;
            z = yy;
            x = zz;
            break;

        case 5:     // Mirror about y=z plane
            z = yy;
            y = zz;
            break;
    };

    //Change some more based on the octant

    //if first bit is 0, negate x component
    if( !(iOctant & 0x4) )
        x = -x;

    //if second bit is 0, negate y component
    if( !(iOctant & 0x2) )
        y = -y;

    //if third bit is 0, negate z component
    if( !(iOctant & 0x1) )
        z = -z;

    outVec.setValues(x,y,z);

    return True;
}

Bool DeeringNormalCodec::unpackCode(UInt32 code,
                                        UInt32& outSextant,
                                        UInt32& outOctant,
                                        UInt32& outTheta,
                                        UInt32& outPsi)
{
    UInt32 mask = (1<<numBits)-1;

    outSextant = (code >> (numBits+numBits+3)) & 0x7;
    outOctant  = (code >> (numBits+numBits))   & 0x7;
    outTheta   = (code >> (numBits))           & mask;
    outPsi     = (code)                        & mask;

    return True;
}
```

Annex C
Hashing – An Implementation

This Appendix provides a sample C++ implementation for the creation of hash values (as detailed in Encoding Algorithms) used in the JT format.

```
unsigned int hash32( const unsigned int *pWords,
                     int nWords,
                     unsigned int uSeedHashValue )
{ return hash2(pWords, nWords, uSeedHashValue); }

unsigned int jthash16(const unsigned short *pBytes,
                      int nShort,
                      unsigned int uSeedHashValue)
{ return hash3(pBytes, nShort, uSeedHashValue); }

//--------------------------------------------------------------------
//  mix -- mix 3 32-bit values reversibly.
// For every delta with one or two bit set, and the deltas of all three
//   high bits or all three low bits, whether the original value of a,b,c
//   is almost all zero or is uniformly distributed,
// * If mix() is run forward or backward, at least 32 bits in a,b,c
//   have at least 1/4 probability of changing.
// * If mix() is run forward, every bit of c will change between 1/3 and
//   2/3 of the time.  (Well, 22/100 and 78/100 for some 2-bit deltas.)
// mix() was built out of 36 single-cycle latency instructions in a
//   structure that could supported 2x parallelism, like so:
//       a -= b;
//       a -= c; x = (c>>13);
//       b -= c; a ^= x;
//       b -= a; x = (a<<8);
//       c -= a; b ^= x;
//       c -= b; x = (b>>13);
//       ...
//   Unfortunately, superscalar Pentiums and Sparcs can't take advantage
//   of that parallelism.  They've also turned some of those single-cycle
//   latency instructions into multi-cycle latency instructions.  Still,
//   this is the fastest good hash I could find.  There were about 2^^68
//   to choose from.  I only looked at a billion or so.
--------------------------------------------------------------------

#define mix(a,b,c) \
{ \
  a -= b; a -= c; a ^= (c>>13); \
  b -= c; b -= a; b ^= (a<<8); \
  c -= a; c -= b; c ^= (b>>13); \
  a -= b; a -= c; a ^= (c>>12);  \
  b -= c; b -= a; b ^= (a<<16); \
  c -= a; c -= b; c ^= (b>>5); \
  a -= b; a -= c; a ^= (c>>3);  \
  b -= c; b -= a; b ^= (a<<10); \
  c -= a; c -= b; c ^= (b>>15); \
}

--------------------------------------------------------------------
// hash() -- hash a variable-length key into a 32-bit value
//   k     : the key (the unaligned variable-length array of bytes)
//   len   : the length of the key, counting by bytes
//   level : can be any 4-byte value
// Returns a 32-bit value.  Every bit of the key affects every bit of
// the return value.  Every 1-bit and 2-bit delta achieves avalanche.
// About 36+6len instructions.
```

```
// The best hash table sizes are powers of 2.  There is no need to do
// mod a prime (mod is sooo slow!).  If you need less than 32 bits,
// use a bitmask.  For example, if you need only 10 bits, do
//   h = (h & hashmask(10));
// In which case, the hash table should have hashsize(10) elements.
//
// If you are hashing n strings (JtUInt8 **)k, do it like this:
//   for (i=0, h=0; i<n; ++i) h = hash( k[i], len[i], h);
//
// By Bob Jenkins, 1996.  bob_jenkins@burtleburtle.net.  You may use this
// code any way you wish, private, educational, or commercial.  It's free.
//
// See http://burtleburtle.net/bob/                 // 2010/02/12
// See http://burtleburtle.net/bob/hash/doobs.html   // 2010/02/12
//
// Use for hash table lookup, or anything where one collision in 2^32 is
// acceptable.  Do NOT use for cryptographic purposes.
//----------------------------------------------------------------------

//----------------------------------------------------------------------
// This works on all machines.  hash2() is identical to hash() on
// little-endian machines, except that the length has to be measured
// in ub4s instead of bytes.  It is much faster than hash().  It
// requires
// -- that the key be an array of UInt32's, and
// -- that all your machines have the same endianness, and
// -- that the length be the number of UInt32's in the key
// ----------------------------------------------------------------------
unsigned int hash(const usigned char *k,        // key
                  unsigned int        length,   // length of the key
                  unsigned int        initval)  // prev hash, or an arbitrary value
{
   register unsigned int a,b,c,len;

   /* Set up the internal state */
   len = length;
   a = b = 0x9e3779b9;  /* the golden ratio; an arbitrary value */
   c = initval;           /* the previous hash value */
   /*--------------------------------------- handle most of the key */
   while (len >= 12) {
      a += (k[0] +((UInt32)k[1]<<8) +((UInt32)k[2]<<16) +((UInt32)k[3]<<24));
      b += (k[4] +((UInt32)k[5]<<8) +((UInt32)k[6]<<16) +((UInt32)k[7]<<24));
      c += (k[8] +((UInt32)k[9]<<8) +((UInt32)k[10]<<16)+((UInt32)k[11]<<24));
      mix(a,b,c);
      k += 12; len -= 12;
   }
   /*------------------------------------- handle the last 11 bytes */
   c += length;
   switch(len) {             /* all the case statements fall through */
     case 11: c+=((UInt32)k[10]<<24);
     case 10: c+=((UInt32)k[9]<<16);
     case 9 : c+=((UInt32)k[8]<<8);
      /* the first byte of c is reserved for the length */
     case 8 : b+=((UInt32)k[7]<<24);
     case 7 : b+=((UInt32)k[6]<<16);
     case 6 : b+=((UInt32)k[5]<<8);
     case 5 : b+=k[4];
     case 4 : a+=((UInt32)k[3]<<24);
     case 3 : a+=((UInt32)k[2]<<16);
     case 2 : a+=((UInt32)k[1]<<8);
     case 1 : a+=k[0];
     /* case 0: nothing left to add */
   }
   mix(a,b,c);
   /*-------------------------------------------- report the result */
   return c;
}
```

```
unsigned int hash3(const unsigned short *k,       /* the key */
                   unsigned int          length,  /* the length of the key */
                   unsigned int          initval) /* the previous hash, or an
arbitrary value */
{
   unsigned int a,b,c,len;

   /* Set up the internal state */
   len = length;
   a = b = 0x9e3779b9;    /* the golden ratio; an arbitrary value */
   c = initval;           /* the previous hash value */

   /*---------------------------------------- handle most of the key */
   while (len >= 6)
   {
      a += (k[0] + (UInt32(k[1]) << 16));
      b += (k[2] + (UInt32(k[3]) << 16));
      c += (k[4] + (UInt32(k[5]) << 16));
      mix(a,b,c);
      k += 6; len -= 6;
   }

   /*-------------------------------------- handle the last 2 uint32s */
   c += length;
   switch(len)              /* all the case statements fall through */
   {
       case 5 : c+=(UInt32(k[4]) << 16);
       /* c is reserved for the length */
       case 4 : b+=(UInt32(k[3]) << 16);
       case 3 : b+=k[2];
       case 2 : a+=(UInt32(k[1]) << 16);
       case 1 : a+=k[0];
       /* case 0: nothing left to add */
   }
   mix(a,b,c);
   /*-------------------------------------------- report the result */
   return c;
}
```

Annex D
Polygon Mesh Topology Coder

The topology coding algorithm described here is used to code the *dual* of the desired mesh. Thus, for example, the reader will need to take the dual of the decoded mesh in order to obtain the original primal mesh. Presented below are classes suitable for representing the dual of a polygon mesh and the dual topology decoding algorithm.

At a high level, the topology coder works by traversing the dual mesh to be encoded one vertex and one face at a time. The coder maintains a queue of faces to be processed; the initial queue is created using the valence of an arbitrary vertex of the mesh followed by the degrees of the faces adjacent to that vertex, and adds the adjacent faces to the face queue. Each time it visits a face, it encodes the *degree* of that face and emits any incident vertices that have not yet been visited. Each time the coder visits a vertex, it encodes the *valence* of the vertex (usually 3 in the current case), and emits any incident faces that have not yet been visited. It works its way through the mesh in this fashion until all vertices and faces have been encoded. Thus, the primary output from the topology coder is a list of vertex valences and face degrees. These two fields plus two more encoding so-called *split faces*, coupled with the exact coder implementation completely encode the mesh topology in a very compact manner[1].

In addition to these two basic fields are added a number of other fields that organize the dual vertices into *vertex groups*, and also encode the *vertex attributes* (e.g. normals, colours, and texture coordinates) around each dual face's *degree ring*.

The topological coder can only encode *closed, manifold* meshes. It cannot encode *boundaries*; it can only encode edges with exactly two incident faces. But, as we know, real-world data is chock full of meshes with boundaries. In order to encode these types of meshes, it is necessary to add *cover faces* incident to all boundary loops whose sole job is to turn the mesh into a *closed* mesh. It is the dual of this closed, manifold mesh that is actually encoded. Thus, most meshes encoded in JT files contain a few cover faces. These faces may be of arbitrarily high degree, and they represent the only exceptions to the general rule that the numbers in the dual vertex valence array are usually three. It is necessary to flag all such artificially introduced cover faces so that they can be removed by the loader. These flags are encoded below in the Face Flags array. Primal faces are flagged with zero, while cover faces are flagged with one.

Now, let us make the connection between topological vertices and how vertex attributes relate to them. Several faces may be incident on the same topological mesh vertex. While this topological vertex has only a single 3D coordinate, it may have a different set of *vertex attributes* for each incident face. Vertex attributes include colour, normal, and texture coordinates. An important observation in real-world data is that adjacent faces tend to share the same vertex attributes. Thus, a natural way to encode which vertex attributes map to which faces within a given valence ring (the counter-clockwise ordered set of faces incident on a given vertex) is by way of a bit vector. The bit vector begins at the first face the coder encounters that is incident to the vertex, and proceeds counter clockwise around the vertex, allocating one bit per incident face. A value of 0 is assigned to the bit if all vertex attributes for the face are the same as the face immediately clockwise. A value of 1 is assigned if the vertex attributes for the face are different. Recall that these bits from the original primal mesh are encoded as face attributes in the dual mesh.

Thus, at the end of the coding process, there will be one such bit vector per topological vertex in the mesh. These bit vectors will be of disparate lengths because all vertex valences are not the same. Though there is no theoretical limit to the valence of any given vertex, in practice, the vertex valences

[1] Similar methods of topology coding are described in [24] and US patent # 7,098,916. The topology coding algorithm described herein differs from such methods in that while they utilize a queue of active *vertices*, the instant algorithm utilizes a queue of active *faces*. Other differences include the tracking of face group numbers and per-vertex attributes such as normals, colours, and texture coordinates.

seldom rise above six, and only rarely rise into the dozens. As a matter of practicality, then, we break this list of bit vectors into those of length 64 and smaller into one group, and all others into a list of so-called “high-valence” bit vectors. The low-valence bit vectors are encoded into two fields of 32 bits each. The high-valence bit vectors are adjoined end-to-end into a single long bit vector, and encoded as a single array of integers. As an additional optimization, the low-valence bit vectors are grouped into 8 “context groups” depending on the valence of the vertex being coded. This is done in order to improve compression performance because the valence bit vectors in each of the most common groups typically share similar statistics. Context group number 8 is the only one that encodes valence rings up to valence 64. Again, recall that these attribute bits from the original primal mesh are encoded as face attribute bits in the dual mesh.

D.1 DualVFMesh

The DualVFMesh (Dual Vertex-Facet Mesh) is a support class paired with the topology decoder itself, and represents a closed two-manifold polygon mesh. The topology decoder reconstructs the encoded dual mesh into a DualVFMesh, building it one vertex and one facet at a time. When the decoder is finished, it will have visited each vertex and each face of the dual mesh exactly once. DualVFMesh is not intended as a work-horse in-memory storage container because its way of encoding the topological connections between faces and vertices is memory-intensive.

```
class DualVFMesh
{
  public:
    // ========== Housekeeping Interface ==========
    DualVFMesh();
    DualVFMesh (const DualVFMesh &rhs);
    DualVFMesh &operator=(const DualVFMesh &rhs);

    // ========== Topology Interface ==========

    // Vtx creation
    bool          isValidVtx (Int32  iVtx) const;
    bool          newVtx     (Int32  iVtx,
                              Int32  iValence,
                              UInt16 uFlags = 0);
    bool          setVtxFlags(Int32  iVtx,
                              UInt16 uFlags);
    bool          setVtxGrp  (Int32  iVtx,
                              Int32  iVGrp);
    UInt16        vtxFlags   (Int32  iVtx) const;
    Int32         vtxGrp     (Int32  iVtx) const;

    // Face creation
    bool          isValidFace  (Int32  iFace) const;
    bool          newFace      (Int32  iFace,
                                Int32  cDegree,
                                Int32  cFaceAttrs = 0,
                                UInt64 uFaceAttrMask = 0,
                                UInt16 uFlags = 0);
    bool          newFace      (Int32  iFace,
                                Int32  cDegree,
                                Int32  cFaceAttrs,
                                const BitVec *pvbFaceAttrMask,
                                UInt16 uFlags);
    bool          setFaceFlags (Int32  iFace,
                                UInt16 uFlags);
    UInt16        faceFlags    (Int32  iVtx) const;
    bool          setFaceAttr  (Int32  iFace,
                                Int32  iAttrSlot,
                                Int32  iFaceAttr);
    Int32         faceAttr     (Int32  iFace,
                                Int32  iAttrSlot) const;

    // Topology connection
```

```
    bool          setVtxFace(Int32  iVtx,
                             Int32  iFaceSlot,
                             Int32  iFace);
    bool          setFaceVtx(Int32  iFace,
                             Int32  iVtxSlot,
                             Int32  iVtx);

    // Queries
    Int32         valence    (Int32  iVtx) const
        { return _vVtxEnts[iVtx].cVal; }
    Int32         degree   (Int32  iFace ) const
        { return _vFaceEnts[iFace].cDeg; }
    Int32         face    (Int32  iVtx,
                           Int32  iFaceSlot) const
          { return _viVtxFaceIndices[(_vVtxEnts[iVtx]).iVFI + iFaceSlot]; }
    Int32         vtx      (Int32  iFace,
                            Int32  iVtxSlot) const
          { return _viFaceVtxIndices[_vFaceEnts[iFace].iFVI + iVtxSlot]; }
    Int32         numVts       () const
        { return _vVtxEnts.length(); }
    Int32         numFaces     () const
        { return _vFaceEnts.length(); }
    Int32         numAttrs     () const
        { return _viFaceAttrIndices.length(); }
    Int32         numAttrs       (Int32 iFace) const
        { return _vFaceEnts[iFace].cFaceAttrs; }
    UInt64        attrMask       (Int32 iFace) const
        { return _vFaceEnts[iFace].u.uAttrMask; }
    const BitVec *attrMaskV      (Int32 iFace) const
        { return _vFaceEnts[iFace].u.pvbAttrMask; }
    Int32         findVtxSlot   (Int32 iFace,
                                 Int32 iTargVtx) const;
    Int32         findFaceSlot    (Int32 iVtx,
                                   Int32 iTargFace) const;
    Int32         emptyFaceSlots  (Int32 iFace) const
        { return _vFaceEnts[iFace].cEmptyDeg; }

    // ========== VFMesh Data Members ==========
  public:
    class VtxEnt {
      public:
        VtxEnt() : cVal(0), uFlags(0), iVGrp(-1), iVFI(-1) {}
        UInt16    cVal;   // Vtx valence
        UInt16    uFlags; // User flags
        Int32     iVGrp;  // Vtx group
        Int32     iVFI;   // Idx into _viVtxFaceIndices of cVal incident faces
    };

    // Number of optimized mask bits.
    static const Int32 cMBits = 64;

    class FaceEnt {
      public:
        FaceEnt() : cDeg(0), uFlags(0), cEmptyDeg(0),
                    cFaceAttrs(0), iFVI(-1), iFAI(-1) { u.uAttrMask = 0; }
        FaceEnt(const FaceEnt &rhs) : cDeg(rhs.cDeg), cEmptyDeg(rhs.cEmptyDeg),
                                      cFaceAttrs(rhs.cFaceAttrs), iFVI(rhs.iFVI),
                                      iFAI(rhs.iFAI)
        {
            if (cDeg <= cMBits)
                u.uAttrMask = rhs.u.uAttrMask;
            else
                JtWrapNew(u.pvbAttrMask, new BitVec(*rhs.u.pvbAttrMask));
        }
        ~FaceEnt() { if (cDeg > cMBits && u.pvbAttrMask) delete u.pvbAttrMask; }
        UInt16    cDeg;       // Face degree
        UInt16    cEmptyDeg;  // Empty degrees (opt for emptyFaceSlots())
```

```
        UInt16    cFaceAttrs;  // Number of face attributes
        UInt16    uFlags;     // User flags
        union {
            UInt64  uAttrMask;    // Degree-ring attr mask as a UInt64
            BitVec *pvbAttrMask;  // Degree-ring attr mask as a BitVec
        } u;
        Int32     iFVI; // Idx into _viFaceVtxIndices of cDeg incident vts
        Int32     iFAI; // Idx into _viFaceAttrIndices of cAttr attributes
    };

  protected:
    // Subscripted by atom number, the entry contains the vtx valence and
    // points to the location in _viVtxFaceIndices of valence consecutive
    // integers that in turn contain the indices of the incident faces
    // in _vFaceRecs to the vtx.
    JtVec<VtxEnt>  _vVtxEnts;

    // Subscripted by unique vertex record number, the entry contains the
    // face degree and points to the location in _viFaceVtxIndices of
    // cDeg consecutive integers that in turn contain the indices of the
    // vertices indicent upon the face, in CCW order, in _vVtxRecs.
    JtVec<FaceEnt>   _vFaceEnts;

    // Combined storage for all vtxs.
    JtVeci          _viVtxFaceIndices;

    // Combined storage for all faces.
    JtVeci          _viFaceVtxIndices;

    // Combined storage for all face attribute record identifiers
    JtVeci          _viFaceAttrIndices;
};

bool
DualVFMesh::isValidVtx(Int32 iVtx) const
{
    bool bRet = JtFalse;
    if (iVtx >= 0 && iVtx < _vVtxEnts.length()) {
        const VtxEnt &rFE = _vVtxEnts[iVtx];
        bRet = (rFE.cVal != 0);
    }
    return bRet;
}

bool
DualVFMesh::newVtx(Int32 iVtx,
                   Int32 iValence,
                   UInt16 uFlags)
{
    VtxEnt &rFE = _vVtxEnts[iVtx];
    if (rFE.cVal != iValence) {
        rFE.cVal   = iValence;
        rFE.uFlags = uFlags;
        rFE.iVFI   = _viVtxFaceIndices.length();
        _viVtxFaceIndices.verify(rFE.iVFI + iValence - 1);
        for (Int32 i = rFE.iVFI ; i < rFE.iVFI + iValence ; i++)
            _viVtxFaceIndices[i] = -1;
    }
    return true;
}

bool
DualVFMesh::setVtxGrp(Int32  iVtx,
                     Int32  iVGrp)
{
    VtxEnt &rFE = _vVtxEnts[iVtx];
    rFE.iVGrp = iVGrp;
```

```
    return true;
}

bool
DualVFMesh::setVtxFlags(Int32  iVtx,
                       UInt16 uFlags)
{
    VtxEnt &rFE = _vVtxEnts[iVtx];
    rFE.uFlags = uFlags;
    return true;
}

Int32
DualVFMesh::vtxGrp   (Int32  iVtx) const
{
    Int32 u = -1;
    if (iVtx >= 0 && iVtx < _vVtxEnts.length()) {
        const VtxEnt &rFE = _vVtxEnts[iVtx];
        u = rFE.iVGrp;
    }
    return u;
}

UInt16
DualVFMesh::vtxFlags   (Int32  iVtx) const
{
    UInt16 u = 0;
    if (iVtx >= 0 && iVtx < _vVtxEnts.length()) {
        const VtxEnt &rFE = _vVtxEnts[iVtx];
        u = rFE.uFlags;
    }
    return u;
}


bool
DualVFMesh::isValidFace(Int32 iFace) const
{
    bool bRet = JtFalse;
    if (iFace >= 0 && iFace < _vFaceEnts.length()) {
        const FaceEnt &rVE = _vFaceEnts[iFace];
        bRet = (rVE.cDeg != 0);
    }
    return bRet;
}

bool
DualVFMesh::newFace(Int32  iFace,
                    Int32  cDegree,
                    Int32  cFaceAttrs,
                    UInt64 uFaceAttrMask,
                    UInt16 uFlags)
{
    FaceEnt &rVE = _vFaceEnts[iFace];
    if (rVE.cDeg != cDegree) {
        rVE.cDeg        = cDegree;
        rVE.cEmptyDeg   = cDegree;
        rVE.cFaceAttrs   = cFaceAttrs;
        rVE.uFlags      = uFlags;
        rVE.u.uAttrMask = uFaceAttrMask;
        rVE.iFVI        = _viFaceVtxIndices.length();
        rVE.iFAI        = _viFaceAttrIndices.length();
        _viFaceVtxIndices.verify(rVE.iFVI + cDegree  - 1);
        if (cFaceAttrs > 0)
            _viFaceAttrIndices.verify(rVE.iFAI + cFaceAttrs - 1);
        for (Int32 i = rVE.iFVI ; i < rVE.iFVI + cDegree ; i++)
            _viFaceVtxIndices[i] = -1;
```

```
        for (Int32 i = rVE.iFAI ; i < rVE.iFAI + cFaceAttrs ; i++)
            _viFaceAttrIndices[i] = -1;
    }
    return true;
}

bool
DualVFMesh::newFace(Int32  iFace,
                    Int32  cDegree,
                    Int32  cFaceAttrs,
                    const BitVec *pvbFaceAttrMask,
                    UInt16 uFlags)
{
    FaceEnt &rVE = _vFaceEnts[iFace];
    if (rVE.cDeg != cDegree) {
        rVE.cDeg        = cDegree;
        rVE.cEmptyDeg   = cDegree;
        rVE.cFaceAttrs   = cFaceAttrs;
        rVE.uFlags      = uFlags;
        rVE.u.pvbAttrMask = new BitVec(*pvbFaceAttrMask);
        rVE.iFVI        = _viFaceVtxIndices.length();
        rVE.iFAI        = _viFaceAttrIndices.length();
        _viFaceVtxIndices.verify(rVE.iFVI + cDegree  - 1);
        if (cFaceAttrs > 0)
            _viFaceAttrIndices.verify(rVE.iFAI + cFaceAttrs - 1);
        for (Int32 i = rVE.iFVI ; i < rVE.iFVI + cDegree ; i++)
            _viFaceVtxIndices[i] = -1;
        for (Int32 i = rVE.iFAI ; i < rVE.iFAI + cFaceAttrs ; i++)
            _viFaceAttrIndices[i] = -1;
    }
    return true;
}

bool
DualVFMesh::setFaceFlags(Int32  iFace,
                         UInt16 uFlags)
{
    FaceEnt &rVE = _vFaceEnts[iFace];
    rVE.uFlags = uFlags;
    return true;
}

UInt16
DualVFMesh::faceFlags   (Int32  iFace) const
{
    UInt16 u = 0;
    if (iFace >= 0 && iFace < _vFaceEnts.length()) {
        const FaceEnt &rVE = _vFaceEnts[iFace];
        u = rVE.uFlags;
    }
    return u;
}

bool
DualVFMesh::setFaceAttr(Int32  iFace,
                        Int32  iAttrSlot,
                        Int32  iFaceAttr)
{
    FaceEnt &rVE = _vFaceEnts[iFace];
    Int32 *paiFAI = _viFaceAttrIndices.ptr();
    paiFAI[rVE.iFAI + iAttrSlot] = iFaceAttr;
    return true;
}

Int32
DualVFMesh::faceAttr(Int32  iFace,
                     Int32  iAttrSlot) const
```

```
{
    Int32 u = 0;
    if (iFace >= 0 && iFace < _vFaceEnts.length()) {
        const FaceEnt &rVE = _vFaceEnts[iFace];
        if (iAttrSlot >= 0 && iAttrSlot < rVE.cDeg) {
            const Int32 *paiFAI = _viFaceAttrIndices.ptr();
            u = paiFAI[rVE.iFAI + iAttrSlot];
        }
    }
    return u;
}

// Attaches VF face iFace to VF vertex iVtx in the vertex's
// face slot iFaceSlot
bool
DualVFMesh::setVtxFace(Int32 iVtx,
                       Int32 iFaceSlot,
                       Int32 iFace)
{
    VtxEnt &rFE = _vVtxEnts[iVtx];
    _viVtxFaceIndices[rFE.iVFI + iFaceSlot] = iFace;
    return true;
}

// Attaches VF vertex iVtx to VF face iFace in the face's
// vertex slot iVtxSlot
bool
DualVFMesh::setFaceVtx(Int32 iFace,
                       Int32 iVtxSlot,
                       Int32 iVtx)
{
    FaceEnt &rVE = _vFaceEnts[iFace];
    Int32 *paiFVI = _viFaceVtxIndices.ptr();
    rVE.cEmptyDeg -= (paiFVI[rVE.iFVI + iVtxSlot] != iVtx);
    paiFVI[rVE.iFVI + iVtxSlot] = iVtx;
    return true;
}

// Searches the list of incident vts to face iFace for
// iTargVtx and returns the vtx slot at which it is found
// or -1 if iTargVtx is not found.
Int32
DualVFMesh::findVtxSlot(Int32 iFace,
                        Int32 iTargVtx) const
{
    const FaceEnt &rVE = _vFaceEnts[iFace];
    const Int32 *const pFaceVtxIndices = _viFaceVtxIndices.ptr() + rVE.iFVI;
    Int32 cDeg = rVE.cDeg;
    Int32 iSlot = -1;
    for (Int32 iVtxSlot = 0 ; iVtxSlot < cDeg ; iVtxSlot++) {
        if (pFaceVtxIndices[iVtxSlot] == iTargVtx) {
            iSlot = iVtxSlot;
            break;
        }
    }
    return iSlot;
}

// Searches the list of incident faces to vertex iVtx for
// iTargFace and returns the face slot at which it is found
// or -1 if iTargFace is not found.
Int32
DualVFMesh::findFaceSlot (Int32 iVtx,
                          Int32 iTargFace) const
{
    const VtxEnt &rFE = _vVtxEnts[iVtx];
    const Int32 *const pVtxFaceIndices = _viVtxFaceIndices.ptr() + rFE.iVFI;
```

```
    for (Int32 iFaceSlot = 0 ; iFaceSlot < rFE.cVal ; iFaceSlot++) {
        if (pVtxFaceIndices[iFaceSlot] == iTargFace) {
            return iFaceSlot;
        }
    }
    return -1;
}
```

D.2 Topology Decoder

Partial implementations of three classes are given here for MeshCoderDriver, MeshCodec, and MeshDecoder. MeshCodec contains the abstract implementation of the topology coder. MeshDecoder implements the functionality needed to *decode* a mesh from the input data read from a JT file (see Topologically Compressed Rep Data). MeshCoderDriver manages the input data, the output VFMesh, and the MeshDecoder itself, providing a simple three-step API.

D.2.1 MeshCoderDriver class

```
// This class serves as a coordinating driver for mesh coding and decoding.
class MeshCoderDriver
{
  public:
    MeshCoderDriver ();

    // ========== Operations Interface ==========
    void      setInputData(const Veci   vviOutValSyms[/*8*/],
                           const Veci  &viOutDegSyms,
                           const Veci  &viOutFGrpSyms,
                           const Vecus &vuOutFaceFlags,
                           const Veclu  vvuOutAttrMasks[/*8*/],
                           const Vecu  &vuOutAttrMasksLrg,
                           const Veci  &viOutSplitVtxSyms,
                           const Veci  &viOutSplitPosSyms)
                           { /* Copy into 22 fields below */ }
    void      decode();
    VFMesh   *vfm() const { return _pOutVFM; }

    // ========== Utility Methods ==========
    Int32         _nextDegSymbol    (Int32 iCCntx);
    Int32         _nextValSymbol    ();
    Int32         _nextFGrpSymbol   ();
    UInt16        _nextVtxFlagSymbol();
    UInt64        _nextAttrMaskSymbol(Int32 iCCntx);     // <= 64-bit attrmask
    void          _nextAttrMaskSymbol(BitVec *iopvbAttrMask,
                                      Int32   cDegree);  // > 64 bit attrmask
    Int32         _nextSplitFaceSymbol();
    Int32         _nextSplitPosSymbol();
    Int32         _faceCntxt(Int32 iVtx, JtDualVFMesh *pVFM);

    // ========== Member Data ==========
  protected:
    SharedPtr<MeshCodec>     _pMC;      // The mesh coder or decoder being used
    SharedPtr<JtDualVFMesh>  _pOutVFM;  // Back-end VFMesh built by decoder
    SharedPtr<MeshDecoder>   _pMeshDecoder;

    // Coding symbols generated by encoding operation, auxiliary data such as
    // offsets, etc.
    Veci          _vviOutDegSyms[8];  // Face degree + SPLIT symbols for multiple
contexts
    Veci          _viOutValSyms;      // Vtx valence symbols
    Veci          _viOutVGrpSyms;     // Vtx group of each encoded vtx
    Vecus         _vuOutVtxFlags;     // Vtx flags; parallel to _viOutValSyms.
    Veclu         _vvuOutAttrMasks[8];// Attribute bitmasks per face for multiple
contexts.
                                      //  One per non-split entry in _viOutValSyms.
```

```
    Vecu          _vuOutAttrMasksLrg; // > 64-bit attrmasks
    Veci          _viOutSplitFaceSyms;// Split face offsets
    Veci          _viOutSplitPosSyms; // Split face vtx slots

    // The next symbol to be consumed by _next*Symbol()
    Int32         _iValReadPos[8];
    Int32         _iDegReadPos;
    Int32         _iVGrpReadPos;
    Int32         _iFFlagReadPos;
    Int32         _iAttrMaskReadPos[8];
    Int32         _iAttrMaskLrgReadPos;
    Int32         _iSplitFaceReadPos;
    Int32         _iSplitPosReadPos;
};

void MeshCoderDriver::decode()
{
    // Allocate a coder
    if (!_pMeshDecoder) {
        _pMeshDecoder = new MeshDecoder(this);
    }
    _pMC = _pMeshDecoder;
    _pMC->setTopoDualMeshCoder(this);

    // Reset the symbol counters
    for (Int32 i = 0 ; i < 8 ; i++) {
        _iValReadPos[i] = 0;
        _iAttrMaskReadPos[i] = 0;
    }
    _iDegReadPos = 0;
    _iVGrpReadPos = 0;
    _iFFlagReadPos = 0;
    _iAttrMaskLrgReadPos = 0;
    _iSplitFaceReadPos = 0;
    _iSplitPosReadPos = 0;

    // Run the decoder
    _pMC->run();

    // Assert that ALL symbols have been consumed
    for (Int32 i = 0 ; i < 8 ; i++) {
        Assert(_iValReadPos[i]         == _vviOutDegSyms[i].length());
        Assert(_iAttrMaskReadPos[i]    == _vvuOutAttrMasks[i].length());
    }
    Assert(_iDegReadPos         == _viOutValSyms.length());
    Assert(_iVGrpReadPos        == _viOutVGrpSyms.length());
    Assert(_iFFlagReadPos       == _vuOutVtxFlags.length());
    Assert(_iAttrMaskLrgReadPos == _vuOutAttrMasksLrg.length());
    Assert(_iSplitFaceReadPos    == _viOutSplitFaceSyms.length());
    Assert(_iSplitPosReadPos    == _viOutSplitPosSyms.length());

    // Set output VFMesh
    _pOutVFM = _pMC->vfm();
}

Int32 MeshCoderDriver::_nextDegSymbol  (Int32 iCCntx)
{
    Int32 eSym = -1;
    if (_iValReadPos[iCCntx] < _vviOutDegSyms[iCCntx].length())
        eSym = _vviOutDegSyms[iCCntx].value(_iValReadPos[iCCntx]++);
    return eSym;
}

Int32
MeshCoderDriver::_nextValSymbol  ()
{
    Int32 eSym = -1;
```

```
    if (_iDegReadPos < _viOutValSyms.length())
        eSym = _viOutValSyms.value(_iDegReadPos++);
    return eSym;
}

Int32 MeshCoderDriver::_nextFGrpSymbol()
{
    Int32 eSym = -1;
    if (_iVGrpReadPos < _viOutVGrpSyms.length())
        eSym = _viOutVGrpSyms.value(_iVGrpReadPos++);
    return eSym;
}

UInt16 MeshCoderDriver::_nextVtxFlagSymbol  ()
{
    UInt16 eSym = 0;
    if (_iFFlagReadPos < _vuOutVtxFlags.length())
        eSym = _vuOutVtxFlags.value(_iFFlagReadPos++);
    return eSym;
}

UInt64 MeshCoderDriver::_nextAttrMaskSymbol  (Int32 iCCntx)
{
    UInt64 eSym = 0;
    if (_iAttrMaskReadPos[iCCntx] < _vvuOutAttrMasks[iCCntx].length())
        eSym = _vvuOutAttrMasks[iCCntx].value(_iAttrMaskReadPos[iCCntx]++);
    return eSym;
}

void MeshCoderDriver::_nextAttrMaskSymbol(BitVec *iopvbAttrMask, Int32 cDegree)
{
    if (_iAttrMaskLrgReadPos < _vuOutAttrMasksLrg.length()) {
        iopvbAttrMask->setLength(cDegree);
        UInt32 *pu = iopvbAttrMask->ptr();
        Int32 nWords = (cDegree + BitVec::cWordBits - 1) >> BitVec::cBitsLog2;
        memcpy(pu, &_vuOutAttrMasksLrg.value(_iAttrMaskLrgReadPos), nWords * 
sizeof(UInt32));
        _iAttrMaskLrgReadPos += nWords;
    }
    else {
        iopvbAttrMask->setLength(0);
    }
}

Int32 MeshCoderDriver::_nextSplitFaceSymbol  ()
{
    Int32 eSym = -1;
    if (_iSplitFaceReadPos < _viOutSplitFaceSyms.length())
        eSym = _viOutSplitFaceSyms.value(_iSplitFaceReadPos++);
    return eSym;
}

Int32 MeshCoderDriver::_nextSplitPosSymbol  ()
{
    Int32 eSym = -1;
    if (_iSplitPosReadPos < _viOutSplitPosSyms.length())
        eSym = _viOutSplitPosSyms.value(_iSplitPosReadPos++);
    return eSym;
}

// Computes a "compression context" from 0 to 7 inclusive for
// faces on vertex iVtx.  The context is based on the vertex's
// valence, and the total _known_ degree of already-coded
// faces on the vertex at the time of the call.
Int32 MeshCoderDriver::_faceCntxt(JtInt32 iVtx, JtDualVFMesh *pVFM)
{
    // Here, we are going to gather data to be used to determine a
```

```
    // compression contest for the face degree.
    JtInt32 cVal = pVFM->valence(iVtx);
    JtInt32 nKnownFaces = 0;
    JtInt32 cKnownTotDeg = 0;
    for (JtInt32 i = 0 ; i < cVal ; i++) {
        JtInt32 iTmpFace = pVFM->face(iVtx, i);
        if (!pVFM->isValidFace(iTmpFace))
            continue;
        nKnownFaces++;
        cKnownTotDeg += pVFM->degree(iTmpFace);
    }
    JtInt32 iCCntxt = 0;
    if (cVal == 3) {
        // Regular tristrip-like meshes tend to have degree 6 faces
        iCCntxt = (cKnownTotDeg <  nKnownFaces * 6) ? 0 :
                  (cKnownTotDeg == nKnownFaces * 6) ? 1 : 2;
    }
    else if (cVal == 4) {
        // Regular quadstrip-like meshes tend to have degree 4 faces
        iCCntxt = (cKnownTotDeg <  nKnownFaces * 4) ? 3 :
                  (cKnownTotDeg == nKnownFaces * 4) ? 4 : 5;
    }
    else if (cVal == 5)
        // Pentagons are all lumped into context 6
        iCCntxt = 6;
    else
        // All other polygons are lumped into context 7
        iCCntxt = 7;

    return iCCntxt;
}
```

D.2.2 MeshCodec class

```
// This class serves as the abstract base class from which two concrete classes
// are derived to implement the core operations for a polygonal
// mesh coder or decoder.  An instance of this object is used by the
// MeshCoderDriver to encode and decode polygonal meshes.
//
// This class makes extensive use of DualVFMesh objects as the primary source and
// destination mesh topology storage data structures. This mediating data
// structure is necessary because the mesh coding scheme is deeply cooperative
// with and dependent upon such a vertex-facet data structure.  Please refer to
// DualVFMesh for more information.
class MeshCodec {
  public:
    // ========== Housekeeping Interface ==========
    MeshCodec (MeshCoderDriver *pTMC = NULL);
  protected:
    virtual ~MeshCodec() {}
  public:

    // ========== Setup and Apply Interface ==========
    void setMeshCoderDriver(MeshCoderDriver *pTMC) { _pTMC = pTMC; }
    JtDualVFMesh *vfm() const                      { return _pDstVFM; }
    void  run();

    // ========== Generic encode/decode Driver Chain ==========
    void  clear();
    void  runComponent(bool &obFoundComponent);
    void  initNewComponent(bool &obFoundComponent);
    void  completeV(Int32 iFace);
    Int32 activateV(Int32 iVtx,  Int32 iVSlot);
    Int32 activateF(Int32 iFace, Int32 iFSlot);
    void  completeF(Int32 iVtx,  Int32 jFSlot);
    void  addVtxToFace (Int32 iVtx, Int32 iVSlot,
                        Int32 iFace,  Int32 iFSlot);
```

```
    // Active face list management
    void  addActiveFace(Int32 iFace);
    Int32 nextActiveFace();
    void  removeActiveFace(Int32 iFace);
    Int32 activeFaceOffset(Int32 iFace) const;

  private:
    // ========== Polymorphic I/O Interface ==========
    virtual Int32 ioVtxInit   ()                          = 0;
    virtual Int32 ioVtx       (Int32 iFace, Int32 jFSlot) = 0;
    virtual Int32 ioFace      (Int32 iVtx,  Int32 iVSlot) = 0;
    virtual Int32 ioSplitFace (Int32 iVtx,  Int32 iVSlot) = 0;
    virtual Int32 ioSplitPos  (Int32 iVtx,  Int32 iVSlot) = 0;

    // ========== Member Data ==========
  protected:
    MeshCoderDriver         *_pTMC;          // TopoDualMeshCoder this codec is
attached to
    SharedPtr<JtDualVFMesh>  _pSrcVFM;       // Input VFMesh
    SharedPtr<JtDualVFMesh>  _pDstVFM;       // Output VFMesh
    Veci                     _viActiveFaces; // Stack of incomplete "active faces"
    BitVec                   _vbRemovedActiveFaces;   // Helper bitvec parallel to
above
    // Used by decoder to assign running attr indices
    Int32              _iFaceAttrCtr;
};


// Runs the mesh encoder/decoder machine.
// If decoding is being performed, it consumes the mesh
// coding symbols from pre-filled member variables to produce
// the output VFMesh _pDstVFM.
void MeshCodec::run()
{
    // Assert state is consistent and ready to co/dec
    if (!_pDstVFM)
        _pDstVFM = new JtDualVFMesh();
    Assert(_pDstVFM);
    _pDstVFM->clear();
    clear();

    // Co/dec connected mesh components one at a time
    bool bFoundComponent = JtTrue;
    while (bFoundComponent) {
        Bool bRetVal = runComponent(bFoundComponent);
        Assert (bRetVal);
    }
}

void MeshCodec::clear()
{
    // Setup
    _viActiveFaces.setLength(0);
    _vbRemovedActiveFaces.setLength(0);
    _iFaceAttrCtr = 0;
}

// Decodes one "connected component" (contiguous group of polygons) into
// _pDstVFM. Because the polygonal model may be formed of multiple
//  disconnected mesh components, it may be necessary for run() to call this
// method multiple times.  This method returns obFoundComponent = True
// if it actually encoded a new mesh component, and obFoundComponent = False
// if it did not.
void MeshCodec::runComponent(bool &obFoundComponent)
{
    Int32 iFace;
```

```
    initNewComponent(obFoundComponent);
    if (!obFoundComponent)
        return;
    while ((iFace = nextActiveFace()) != -1) {
        completeF(iFace);
        removeActiveFace(iFace);
    }
}

// Locates an unencoded vertex and begins the encoding
// process for the newly-found mesh component.
void MeshCodec::initNewComponent(bool &obFoundComponent)
{
    obFoundComponent = JtTrue;

    // Call ioVtxInit() to start us off with the seed face
    // from a new "connected component" of polygons.
    Int32 iVtx, i;
    if ((iVtx = ioVtxInit()) == -1) {
        obFoundComponent = JtFalse;  // All vtxs are processed
        return;
    }
    Int32 cVal = _pDstVFM->valence(iVtx);
    for (i = 0 ; i < cVal ; i++) {
        Int32 iFace = activateF(iVtx, i);    // Process all faces
        if (iFace == -2) {
            Assert(0 && "Mesh traversal failed");
            return false;
        }
    }
}

// Completes the VFMesh face iFace on _pDstVFM by calling activateV() and
// completeV() for each as-yet inactive incident vertexes in the face's
// degree ring.
void MeshCodec::completeF(Int32 iFace)
{
    // While there is an empty vtx slot on the face
    Int32 jVtxSlot, iVtx;
    Int32 iVSlot = 0;
    while ((jVtxSlot = _pDstVFM->findVtxSlot(iFace, -1)) != -1) {
        // Create and return a vtx iVtx, attaching it to iFace at vtx
        // slot jVtxSlot.
        iVtx = activateV(iFace, jVtxSlot);

        // Assert FV consistency
        Assert(_pDstVFM->vtx (iFace,  jVtxSlot) == iVtx &&
               _pDstVFM->face(iVtx,   iVSlot)   == iFace   );

        // Process the faces of iVtx starting from face slot
        // jVtxSlot where iVtx is incident on iFace.
        completeV(iVtx, jVtxSlot);

        // Invariant "VF": vtx(iVtx).face(iVSlot) == iFace &&
        //                 face(iFace).vtx(jVtxSlot) == iVtx
    }
}

// "Activates" the VFMesh face, on _pDstVFM, at face iFace vertex slot iVSlot
// by calling ioFace() to obtain a new vertex number and hooking it up to the
// topological structure.  If the face is a SPLIT face, then call
// ioSplitFace() and ioSplitPos() to get the information necessary to connect
// to an already-active face.  Note that we use the term "activate" here to
// mean "read" for mesh decoding.
Int32 MeshCodec::activateF(Int32 iVtx, Int32 iVSlot)
{
    Int32 jFSlot;
```

```
    // ioFace might return -2 as an error condition
    Int32 iFace = ioFace(iVtx, iVSlot);
    if (iFace >= 0) {   // If a new active face
        if (!_pDstVFM->setVtxFace(iVtx, iVSlot, iFace) ||
            !_pDstVFM->setFaceVtx(iFace, 0, iVtx)      ||
            !addActiveFace(iFace)                          )
        {
            return -2;
        }
    }
    else if (iFace == -1) {                   // Face already exists, so Split
        iFace = ioSplitFace(iVtx, iVSlot);    // v's index in ActiveSet, returns v
        jFSlot = ioSplitPos(iVtx, iVSlot);    // Position of iVtx in v
        if (iFace == -2 || iVSlot == -1)
            return -2;
        _pDstVFM->setVtxFace(iVtx, iVSlot, iFace);
        addVtxToFace(iVtx, iVSlot, iFace, jFSlot);
    }
    return iFace;
}

// "Activates" the VFMesh vertex, on _pDstVFM, at face iFace vertex slot iVSlot
// by calling ioFace() to obtain a new face number and hooking it up to the
// topological structure.  Note that we use the term "activate" here to
// mean "read" for mesh decoding.
Int32 MeshCodec::activateV(Int32 iFace, Int32 iVSlot)
{
    Int32 iVtx = ioVtx(iFace, iVSlot);   // I/O valence; create a vtx
    _pDstVFM->setVtxFace(iVtx, 0, iFace);
    addVtxToFace (iVtx, 0, iFace, iVSlot);
    return iVtx;
}

// Completes the vertex iVtx on _pDstVFM by activating all inactive faces
// incident upon it.  As an optimization, the user shall also pass in iVSlot
// which is the vertex slot on face 0 of iVtx where iVtx is located.  This
// method begins its examination of iVtx's faces at face 0 by working its
// way around the vertex in both CCW and CW directions, checking to see if there
// are any faces that can be hooked into iVtx without calling activateF().
// This can happen when a face is completed by a nearby vertex before coming
// here.  The situation can be detected by traversing the topology of the
// _pDstVFM over to the neighboring vertex and checking if it already has a
// face number for the corresponding face entry on iVtx.  If so, then
// iVtx and the already completed face are connected together, and the
// next face around iVtx is examined. When the process can go no further,
// this method calls _activateF() on the remaining unresolved span of faces
// around the vertex.
void MeshCodec::completeF(Int32 iVtx, Int32 iVSlot)
{
    JtDualVFMesh *pDstVFM = _pDstVFM;
    Int32 i, vp, vn, jp, jn,
            iVtx2,
            cVal = pDstVFM->valence(iVtx);

    // Walk CCW from face slot 0, attempting to link in as many
    // already-reachable faces as possible until we reach one
    // that is inactive.
    vp = pDstVFM->face(iVtx, 0);
    jp = iVSlot;
    i = 1;
    JtDebugOnly(_assertParallelValRings(vp);)
    while ((vn = pDstVFM->face(iVtx, i)) != -1) {  // Forces "FV" in the "next"
direction
        DecModN(jp, pDstVFM->degree(vp));
        iVtx2 = pDstVFM->vtx(vp, jp);
        if (iVtx2 == -1)
            break;
```

```
        jn = pDstVFM->findVtxSlot(vn, iVtx2);
        Assert(jn > -1);
        DecModN(jn, pDstVFM->degree(vn));
        addVtxToFace(iVtx, i, vn, jn);
        vp = vn;
        jp = jn;
        i++;
        if (i >= cVal)
            return;
    }

    // Walk CW from face slot 0, attempting to link in as many
    // already-reachable faces as possible until we reach one
    // that is inactive.
    Int32 ilast = i;
    vp = pDstVFM->face(iVtx, 0);
    jp = iVSlot;
    i = pDstVFM->valence(iVtx) - 1;
    while ((vn = pDstVFM->face(iVtx, i)) != -1) { // Forces "VF" in "prev"
direction
        IncModN(jp, pDstVFM->degree(vp));
        iVtx2 = pDstVFM->vtx(vp, jp);
        if (iVtx2 == -1)
            break;
        jn = pDstVFM->findVtxSlot(vn, iVtx2);
        Assert(jn > -1);
        IncModN(jn, pDstVFM->degree(vn));
        addVtxToFace(iVtx, i, vn, jn);
        vp = vn;
        jp = jn;
        i--;
        if (i < ilast)
            return;
    }

    // Activate the remaining faces on iVtx that cannot be decuced from
    // the already-assembled topology in the destination VFMesh.
    for (; ilast <= i ; ilast++) {
        Int32 iFace = activateV(iVtx, ilast);
        JtDemandState(iFace >= -1);
    }
}

// This method connects vertex iVtx into the topology of
// _pDstVFM at and around iFace.  First, it connects iVtx
// to iFace's degree ring at position iVSlot.  Next, it
// will connect iVtx into the faces at the other ends of
// the shared edges between iVtx and the next vertices CS and
// CCW about iFace if necessary.
void MeshCodec::addVtxToFace (Int32 iVtx,  Int32 jFSlot,
                              Int32 iFace, Int32 iVSlot)
{
    Int32   iVSlotCW  = iVSlot,
            iVSlotCCW = iVSlot,
            fp, ip,
            fn, in;
    JtDualVFMesh *pDstVFM = _pDstVFM;
    IncModN(iVSlotCCW, pDstVFM->degree(iFace));
    DecModN(iVSlotCW,  pDstVFM->degree(iFace));

    // Connect iVtx to iFace/iVSlot
    JtRethrow(pDstVFM->setFaceVtx(iFace, iVSlot, iVtx));

    // Connect iVtx across the shared edge between iVtx and the vtx CW
    // from iVtx at iFace.  Connect iVtx into the face at the other
    // end of this edge if it is not already connected there.
    if ((fp = pDstVFM->vtx(iFace, iVSlotCW)) != -1) {
```

```
        ip = pDstVFM->findFaceSlot(fp, iFace);
        Int32 iVSlotCCW = jFSlot;
        IncModN(iVSlotCCW, pDstVFM->valence (iVtx));
        if (pDstVFM->face(iVtx, iVSlotCCW) == -1) {
            DecModN(ip, pDstVFM->valence(fp));
            pDstVFM->setVtxFace(iVtx, iVSlotCCW, pDstVFM->face(fp, ip));
        }
    }

    // Connect iVtx across the shared edge between iVtx and the vtx CCW
    // from iVtx at iFace.  Connect iVtx into the face at the other
    // end of this edge if it is not already connected there.
    if ((fn = pDstVFM->vtx(iFace, iVSlotCCW)) != -1) {
        in = pDstVFM->findFaceSlot(fn, iFace);
        Int32 iVSlotCW  = jFSlot;
        DecModN(iVSlotCW,  pDstVFM->valence (iVtx));
        if (pDstVFM->face(iVtx, iVSlotCW) == -1) {
            IncModN(in, pDstVFM->valence(fn));
            pDstVFM->setVtxFace(iVtx, iVSlotCW, pDstVFM->face(fn, in));
        }
    }
}

void MeshCodec::addActiveFace(Int32 iFace)
{
    JtRethrow(_viActiveFaces.pushBack(iFace));
}

// Returns a face from the active queue to be completed. This needn't be the
// one at the end of the queue, because the choice of the next active face
// can affect how many SPLIT symbols are produced.  This method employs a
// fairly simple scheme of searching the most recent 16 active faces for the
// fist one with the smallest number of incomplete slots in its degree ring.
Int32 MeshCodec::nextActiveFace()
{
    Int32 iFace = -1;
    // Search the 16 face record at the end of the
    // queue for the one with lowest remaining degree.
    while (_viActiveFaces.length() > 0 &&
_vbRemovedActiveFaces.test(_viActiveFaces.back()))
        _viActiveFaces.popBack();
    Int32 cLowestEmptyDegree = 9999999;
    Int32 i, iFace0, cEmptyDeg;
    const Int32 cWidth = 16;
    JtDualVFMesh *pDstVFM = _pDstVFM;
    for (i = _viActiveFaces.length() - 1 ;
         i >= ::jtmax(0, _viActiveFaces.length() - cWidth) ;
         i--)
    {
        iFace0 = _viActiveFaces[i];
        if (_vbRemovedActiveFaces.test(iFace0)) {
            _viActiveFaces.remove(i); // TOXIC: O(N^2)
            continue;
        }
        cEmptyDeg = pDstVFM->emptyFaceSlots(iFace0);
        if (cEmptyDeg < cLowestEmptyDegree) {
            cLowestEmptyDegree = cEmptyDeg;
            iFace = iFace0;
        }
    }

    // Return the selected active face
    return iFace;
}

// Removes iFace from the active face queue.
void MeshCodec::removeActiveFace(Int32 iFace)
```

```
{
    _vbRemovedActiveFaces.set(iFace);
}

// Searches the active face queue for iFace and returns
// its index position from the _end_ of the queue.  This is
// needed by the ioFace() method when encoding a SPLIT
// symbol.
Int32 MeshCodec::activeFaceOffset(Int32 iFace) const
{
    Int32 iOffset = -1;
    Int32 i, cLen = _viActiveFaces.length();
    const Int32 *paiActiveFaces = _viActiveFaces.ptr();
    for (i = cLen - 1 ; i >= 0 ; i--) {
        if (paiActiveFaces[i] == iFace) {
            // The offset is how far FROM THE END of the active
            // face list we found iFace.  This serves the make
            // the iOffset a much smaller number, which is better
            // for compression!
            iOffset = cLen - i;
            break;
        }
    }
    return iOffset;
}
```

D.2.3 MeshDecoder class

```
// This class implements the five abstract methods from
// MeshCodec to realize a mesh decoder.
class MeshDecoder : public MeshCodec {
  public:
    // ========== Housekeeping Interface ==========
    MeshDecoder (MeshCoderDriver *pTMC = NULL);
  protected:
    virtual ~MeshDecoder() {}

  private:
    // ========== Polymorphic I/O Interface ==========
    virtual Int32 ioVtxInit  ()                         ;
    virtual Int32 ioVtx      (Int32 iFace, Int32 iVSlot);
    virtual Int32 ioFace     (Int32 iVtx , Int32 jFSlot);
    virtual Int32 ioSplitFace(Int32 iVtx , Int32 jFSlot);
    virtual Int32 ioSplitPos (Int32 iVtx , Int32 jFSlot);
};

// Begins decoding a new connected mesh component by calling
// ioVtx() to read the next vertex from the symbol stream.
Int32 MeshDecoder::ioVtxInit()
{
    return ioVtx(-1, -1);
}

// Read a vertex valence symbol, vertex group number, and vertex
// flags from the input symbols stream.  Create a new vertex
// on _pDstVFM with this data, and return the new vertex number.
// It is this method's responsibility to detect the end of
// the input symbol stream by returning -1 when that happens.
Int32 MeshDecoder::ioVtx (Int32 /*iFace*/ , Int32 /*iVSlot*/)
{
    // Obtain a VERTEX VALENCE symbol
    Int32 eSym = _pTMC->_nextValSymbol();
    Int32 iVtxVal, iVtx = -1;
    if (eSym > -1) {
        // Create a new vtxt on the VFMesh
        iVtx = _pDstVFM->numVts();
        iVtxVal = eSym;
```

```
        _pDstVFM->newVtx     (iVtx, iVtxVal);
        _pDstVFM->setVtxGrp  (iVtx, _pTMC->_nextFGrpSymbol());
        _pDstVFM->setVtxFlags(iVtx, _pTMC->_nextVtxFlagSymbol());
    }

    return iVtx;
}

// Read a face degree symbol, and attribute mask bit
// vector, create a new DualVFMesh face, initialize the
// face attribute record numbers from a running counter,
// and return the new face number.  If the degree symbol
// read from the input symbol stream is 0, signify this by
// returning -1.
Int32
MeshDecoder::ioFace     (Int32 iVtx, Int32 /*jFSlot*/)
{
    // Obtain a FACE DEGREE symbol
    Int32 iCntxt = _pTMC->_faceCntxt(iVtx, _pDstVFM);
    Int32 eSym = _pTMC->_nextDegSymbol(iCntxt);
    Int32 cDeg, iFace = -1;
    if (eSym != 0) {
        // Create a new face on the VFMesh
        iFace = _pDstVFM->numFaces();
        cDeg = eSym;
        Int32 nFaceAttrs = 0;
        if (cDeg <= JtDualVFMesh::cMBits) {
            UInt64 uAttrMask = _pTMC-
>_nextAttrMaskSymbol(/*iCntxt*/::jtmin(7,::jtmax(0,cDeg-2)));
            for (UInt64 uMask = uAttrMask ; uMask ; nFaceAttrs += (uMask & 1),
uMask >>= 1);
            _pDstVFM->newFace(iFace, cDeg, nFaceAttrs, uAttrMask);
        }
        else {
            BitVec vbAttrMask;
            _pTMC->_nextAttrMaskSymbol(&vbAttrMask, cDeg);
            for (Int32 i = 0 ; i < cDeg ; i++) {
                if (vbAttrMask.test(i))
                    nFaceAttrs++;
            }
            _pDstVFM->newFace(iFace, cDeg, nFaceAttrs, &vbAttrMask, 0);
        }

        // Error check for a corrupt degree or attrmask
        if (nFaceAttrs > cDeg) {
            Assert (nFaceAttrs <= cDeg);
            return -2;
        }

        // Set up the face attributes
        for (Int32 iAttrSlot = 0 ; iAttrSlot < nFaceAttrs ; iAttrSlot++) {
            _pDstVFM->setFaceAttr(iFace, iAttrSlot, _iFaceAttrCtr++);
        }
    }

}

// Consumes a split offset symbol from the SPLIT offset
// symbol stream, and determines the face number referenced
// by the offset.  Returns the referenced face number.
Int32 MeshDecoder::ioSplitFace(Int32 /*iVtx*/, Int32 /*jFSlot*/)
{
    // Obtain a SPLITFACE symbol
    Int32 eSym = _pTMC->_nextSplitFaceSymbol();
    Assert(eSym >= -1);
    Int32 iOffset = -1, iFace = -1;
    if (eSym > -1) {
```

```
        // Use the offset to index into the active face queue
        // to determine the actual face number.
        iOffset = eSym;
        Int32 cLen = _viActiveFaces.length();
        // Error check for a corrupt offset
        if (iOffset <= 0 || iOffset > cLen) {
            Assert(iOffset > 0 && iOffset <= cLen);
            return -2;
        }
        iFace = _viActiveFaces[cLen - iOffset];
    }

    return iFace;
}

// Consumes a split position symbol from the associated symbol
// stream, and returns the vertex slot number on the current
// split face at which the topological split/merge occurred.
Int32 MeshDecoder::ioSplitPos   (Int32 /*iVtx*/, Int32 /*jFSlot*/)
{
    // Obtain a SPLITVTX symbol
    Int32 eSym = _pTMC->_nextSplitPosSymbol();
    Assert(eSym >= -1);
    Int32 iVSlot = -1;
    if (eSym > -1) {
        // Return the vtx slot number
        iVSlot = eSym;
    }

    return iVSlot;
}
```

Annex E
Per Face Group Attributes

With JT V10.5 the ability to apply attributes, such as material and texture image, to a group of faces in the Logical Scene Graph (LSG) is introduced. This is refered to as per face group attributes. An Attribute Element in the logical scene graph (LSG) may be scoped so that it applies only to the geometry contained in a specific subset of grouped faces. Previous versions of the JT file format were capable of only applying Attributes to an entire Shape node.

A Palette Map Attribute is used on a shape such that any face group can be rendered with a chosen entry from the palette. Each Attribute entry in the palette is inherited down the scene graph independent from any other palette entry. Attributes Elements in the scene graph are able to specify to which palette entry they apply.

There are two pieces of information involved in scoping a Palette Map Attribute Element to a face group: The U32:Palette Index field in the Base Attribute Data Fields V2 logical collection, and the PaletteMap Attribute Element.

E.1 U32:Palette Index field description

The U32:Palette Index field is used to implicitly create a State Palette that may later be indexed by a PaletteMap Attribute.

Three rules are used to accumulate the state for each explicitly mentioned palette index:

- Rule 1: The palette index value -1 is regarded as the "fallback palette entry". All Attributes are by default created with palette index -1.
- Rule 2: All Attributes sharing the same Palette Index value are accumulated separately, and constitute an entry in the State Palette.
- Rule 3: The fallback palette entry serves as the initial basis for each other distinct palette entry when the first Attribute of a given Palette Index is encountered during Attribute accumulation.

Rule 3 is best understood by example.

With the example case a user has a scene graph consisting of three nodes organized vertically in a depth 3 scene graph, each bearing various Attributes.The three attribute types are chosen because each accumulates differently in the scene graph.

- N is used to number the nodes, as the type of node is immaterial,
- N2 is the child of N1,
- N3 is the child of N2.
- M represents a Material attribute,
- X represents a Geometric Transform attribute,
- T represents a Texture Image attribute.
- PI denotes the Palette Index setting on an Attribute.
- TC denotes the Texture Channel index on a Texture Image.

N1: JtPartition - M1 (PI:-1), X1 (PI:-1), T0 (PI:-1, TC:0)

N2: VisPartNode - M2 (PI:0), M3 (PI:1), X2 (PI:1), X4(PI:3), T1 (PI:0, TC:1), T2 (PI:0, TC:2)

N3: TriStripSet - M4 (PI:2), M5(PI:0), X3 (PI:2), T3(PI:1, TC:1), T4 (PI:2, TC:0)

The table below shows the accumulated State Palette at node N3 in the example. Each row in the table represents the full accumulated state for a single palette entry.

Table 84 — N3: Tristrip Set Accumulation

N3: TriStripSet - M4 (PI:2), M5 (PI:0), X3 (PI:2), T3(PI:1, TC:1), T4 (PI:2, TC:0)

Palette Index	Material	Transform	Texture(s)
-1 (Fallback)	M1	X1	T0 (TC:0)
0	M5	X1	T0 (TC:0) T1 (TC:1) T2 (TC:2)
1	M3	X1*X2	T0 (TC:0) T3 (TC:1)
2	M4	X1*X3	T4 (TC:0)
3	M1	X1*X4	T0 (TC:0)

The table entries are obtained the following way:

Palette Entry -1 (the fallback): this entry comes directly from the two attributes on node N1 (i.e. M1 and X1). Since there are no other Attributes in the scene graph bearing palette index -1 below them, these two attributes stand as the fallback state on all three nodes. (Rule 1)

Transform state for palette entry 0: Because there is no Transform tagged with Palette Index 0 at or below N1, the fallback entry's Transform state also stands for this palette entry. (Rule 3).

Material state for palette entry 0: The Material attributes involved in this entry are M1, M2, and M5. M1 is a fallback (PI = -1) and so by Rule 3 serves as the basis for all material palette entries. M2 and M5 are explicitly tagged with PI=0 (Rule 2). Since Material attributes accumulate via replacement, M5 is left as the surviving Material. All other Material and Transform entries in the table follow this same logic.

Texture Image accumulation is slightly more subtle because it involves the independent concept of Texture Channel. Palette Index is conceptually similar to the way that multiple Texture Images accumulate via their Texture Channel, however, the two concepts are orthogonal to one another. Each entry in the State Palette can have a set of accumulated Texture Images.

Now draw special attention to Rule 1 above (this rule specifies how an Attribute with Palette Index -1 is accumulated). How would adding Material M6 (PI:-1) to Node N3 affect the State Palette at N3? The answer is that M6 would replace the previously accumulated Material entries in all palette entries. This addition would result in the palette described in the table below.

Table 85 — N3: Tristrip Set Accumulation Updated

N3: TriStripSet - M4 (PI:2), M5 (PI:0), X3 (PI:2), T3(PI:1, TC:1), T4 (PI:2, TC:0), M6 (PI: -1)

Palette Index	Material	Transform	Texture(s)
-1 (Fallback)	M6	X1	T0 (TC:0)
0	M6	X1	T0 (TC:0) T1 (TC:1) T2 (TC:2)
1	M6	X1*X2	T0 (TC:0) T3 (TC:1)
2	M6	X1*X3	T4 (TC:0)
3	M6	X1*X4	T0 (TC:0)

This is how a default-constructed Attribute behaves. A default/fallback Attribute is assertive with respect to any specific palettized state above it in the scene graph.

As a final example of this behavior, it is important to emphasize that a fallback does not override the state that came before it, but rather asserts itself into the normal accumulation mechanism. To illustrate this point, consider the addition of a Transform X5 (PI:-1) to node N2 after the existing attributes:

N1: JtPartition: M1 (PI:-1), X1 (PI:-1), T0 (PI:-1, TC:0)

N2: VisPartNode: M2(PI:0), M3(PI:1), X2(PI:1), X4(PI:3), T1(PI:0,TC:1), T2(PI:0,TC:2), X5 (PI:-1)

N3: TriStripSet: M4 (PI:2), M5(PI:0), X3 (PI:2), T3(PI:1, TC:1), T4 (PI:2, TC:0)

In this case, the new State Palette would appear as in the table below:

Table 86 — N2 Accumulation Updated

N2: VisPartNode: M2(PI:0), M3(PI:1), X2(PI:1), X4(PI:3), T1(PI:0,TC:1), T2(PI:0,TC:2), X5 (PI:-1)

Palette Index	Material (M)	Transform	Texture(s)
-1 (Fallback)	M1	X1*X5	T0 (TC:0)
0	M5	X1*X5	T0 (TC:0) T1 (TC:1) T2 (TC:2)
1	M3	X1*X2*X5	T0 (TC:0) T3 (TC:1)
2	M4	X1*X5*X3	T4 (TC:0)
3	M1	X1*X4*X5	T0 (TC:0)

E.2 PaletteMap Attribute

The PaletteMap attribute maps some or all of the Palette Entries onto the facegroups of a Shape.

Note that even though the PaletteMap is an attribute, its own Palette Index is ignored and must have a value of -1.

Also note that a PaletteMap attribute accumulates (by replacement) down the logical scene graph just like other Attributes. As such it need not be attached directly to a Shape node. It can be placed higher up in the scene graph, for example on a Part Node where all Shapes below it are expected to have the same number of facegroups.

The palette mapping vector is indexed by facegroup number, and contains a Palette Index. The face group geometry will be rendered with the Attribute state obtained by indexing the State Palette with this retrieved Palette Index.

The length of the Palette Map vector must be exactly equal to the number of facegroups in the Shape to which it is applied. If this is not the case, then the PaletteMap is ignored, and the Shape will be rendered using the fallback state as if the PaletteMap did not exist.

Two values have special significance in the palette mapping vector.

- The value -1 refers to the fallback state.
- The value -2 is used to denote that a given facegroup should be inhibited and not rendered at all.

The render-inhibit value affects nothing other than rendering. It does not make the facegroup unpickable, nor does it cause any change to the Node's bounding box, vertex count, or any other quantity.

E.3 Additional information on per face group attributes

Field-final flags, field-inhibit flags, and the force flag all behave as normal for palettized Attributes.

E.4 Addressing Forward Compatibility

To deal with proper forward compatability as it relate to scengraph attributes of JT content the attribute Sabot V104 is added is introduced with JT V10.5. See Sabot V104 definitioin in this document for full details

Annex F
XT B- Rep data segment

The XT B-Rep Segment contains an Element that defines the precise geometric Boundary Representation data for a particular Part in XT boundary representation format.

XT B-Rep segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The XT B-Rep Segment type supports LZMA compression on all element data, so all elements in XT B-Rep Segment use the Logical Element Header Compressed form of element header data.

F.1 XT B-Rep Element

Object Type ID: 0x873a70e0, 0x2ac9, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

XT B-Rep Element represents a particular part’s precise data in XT boundary representation) format.

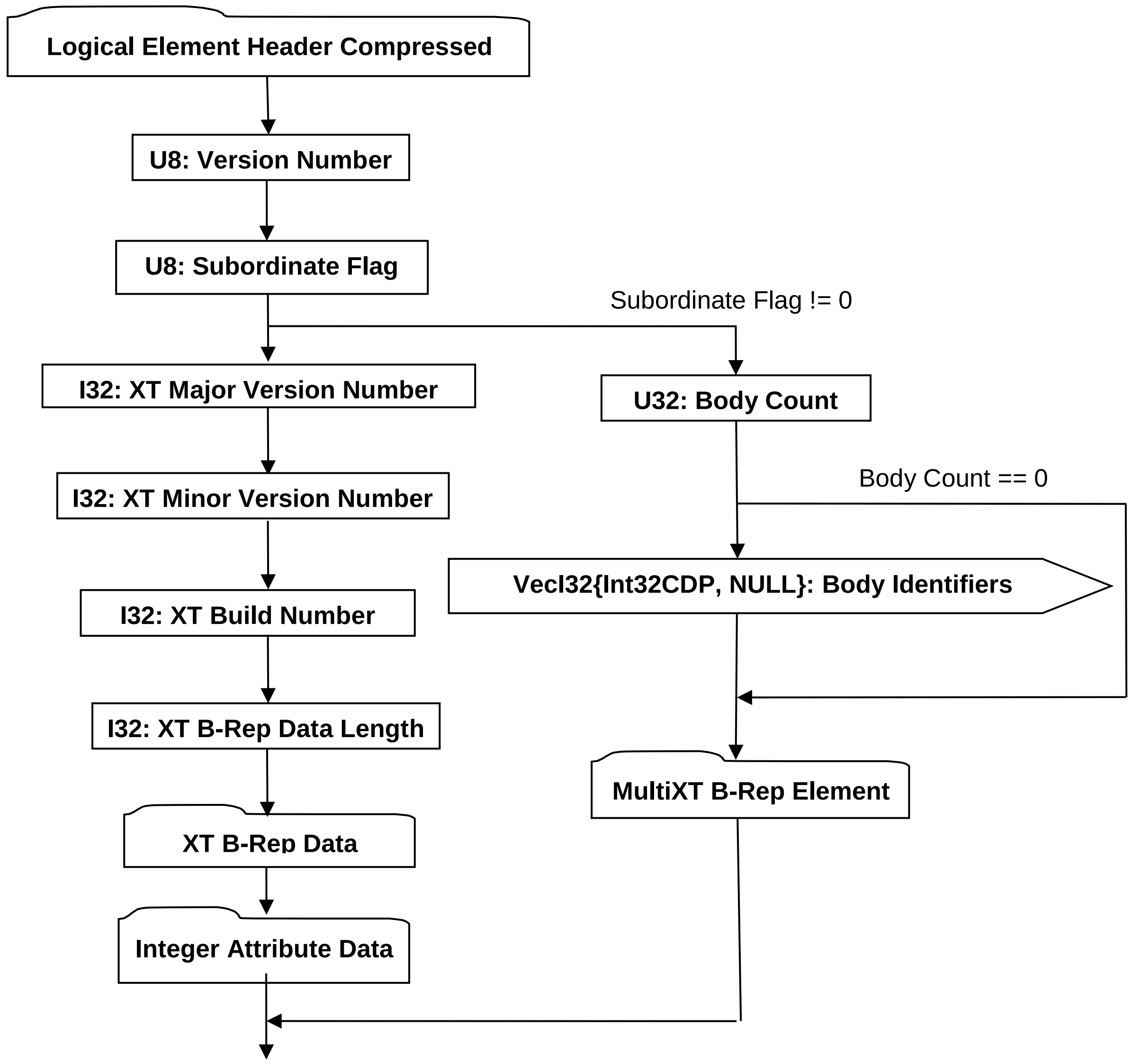


Figure 157 — XT B-Rep Element data collection

Complete description for Logical Element Header Compressed can be found in the File Header section of Base Format Description under Data Segment, Data.

U8: Version Number

Version Number is the version identifier for this XT B-Rep Element. For information on local version numbers see Common Data Conventions and Construct Local version numbers.

U8: Subordinate Flag

Subordinate Flag indicates if this XT B-Rep segment is a subordinate of a XT B-Rep segment. If its value is set to be 0, then this XT B-Rep segment contains complete Parasolid data representation and therefore is not a subordinate. Otherwise if this XT B-Rep segment is a subordinate, meaning that its Parasolid data representation resides in a XT B-Rep segment.

I32: XT Major Version Number

XT Major Version Number specifies the major version number for the XT B-Rep data in the JT File. Major version number is an informative field which can be set to 0 without negative impact to implementation.

I32: XT Minor Version Number

XT Minor Version Number specifies the minor version number for the XT B-Rep data in the JT File. Minor version number is an informative field which can be set to 0 without negative impact to implementation.

I32: XT Build Number

XT Build Number specifies the build number for the XT B-Rep data in the JT File. XT build number is an informative field which can be set to 0 without negative impact to implementation.

I32: XT B-Rep Data Length

XT B-Rep Data Length specifies the length in bytes of the XT B-Rep Element collection. A JT file loader/reader may use this information to compute the end position of the XT B-Rep Data within the JT file and thus skip (for whatever reason) reading the remaining XT B-Rep Data.

U32: Body Count

Body Count specifies the number of XT bodies in this XT B-Rep Element.

VecI32{Int32CDP, NULL}: Body Identifiers

Body Identifiers is an integer array with its length equal to Body Count. The value of each element in this array represents the persisted identifier of the corresponding XT entity. This array is a subset of Body Identifiers array described in MultiXT B-Rep Element, and is used to indicate which XT bodies in MultiXT B-Rep Element belong to this XT B-Rep Element.

F.1.1 XT B-Rep Data

The XT B-Rep Data collection specifies the raw stream of bytes used to represent a Part's XT B-Rep Body(s).

F.1.2 Integer Attribute Data

Integer Attribute Data represents the collection of integer values that may be associated with each face and edge in the XT B-Rep representation.

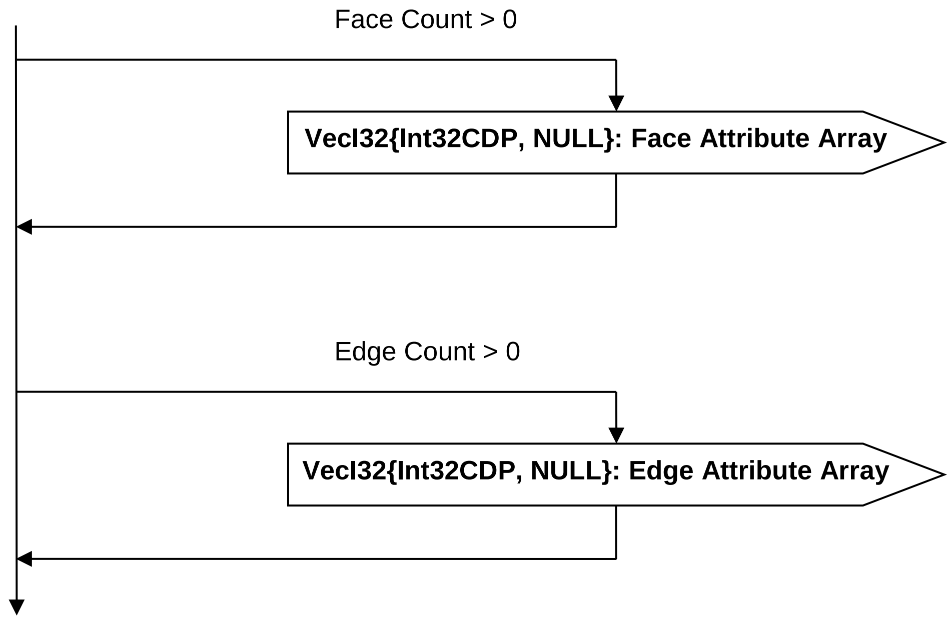


Figure 158 — Integer Attribute Data collection

Face Count

Face Count specifies the number of XT B-Rep faces, which can be found from XT B-Rep Data.

VecI32{Int32CDP, NULL}: Face Attribute Array

Face Attribute Array specifies the collection of integer attributes, one for each XT B-Rep face. The attributes are arranged according to XT B-Rep face index as described in B-Rep Face Group Associations.

Edge Count

Edge Count specifies the number of XT B-Rep edges, which can be found from XT B-Rep Data.

VecI32{Int32CDP, NULL}: Edge Attribute Array

Edge Attribute Array specifies the collection of integer attributes, one for each XT B-Rep edge. The attributes are arranged in the same sequence according XT B-Rep edge index, decided in a similar way as XT B-Rep face index as described in B-Rep Face Group Associations.

F.1.3 MultiXT B-Rep Segment

MultiXT B-Rep Segment contains an Element that defines the precise geometric Boundary Representation data in XT boundary representation format, combined together for one or more Parts. These Parts that refer to a common MultiXT B-Rep Segment typically have similar geometry such that the JT file size can be reduced by this practice. MultiXT B-Rep Segments are typically referenced by all the Part Node Elements (see Part Node Element) that refer to them using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The MultiXT B-Rep Segment type supports LZMA compression on all element data, so all elements in MultiXT B-Rep Segment use the Logical Element Header Compressed form of element header data.

MultiXT B-Rep Element

Object Type ID: 0x49829521, 0x1835, 0x49c3, 0x8b, 0xef, 0xdd, 0xc4, 0x3b, 0xfe, 0x5e, 0x88

MultiXT B-Rep Element represents precise data in XT boundary representation format, combined together for one or more Parts of similar geometry for better compression.

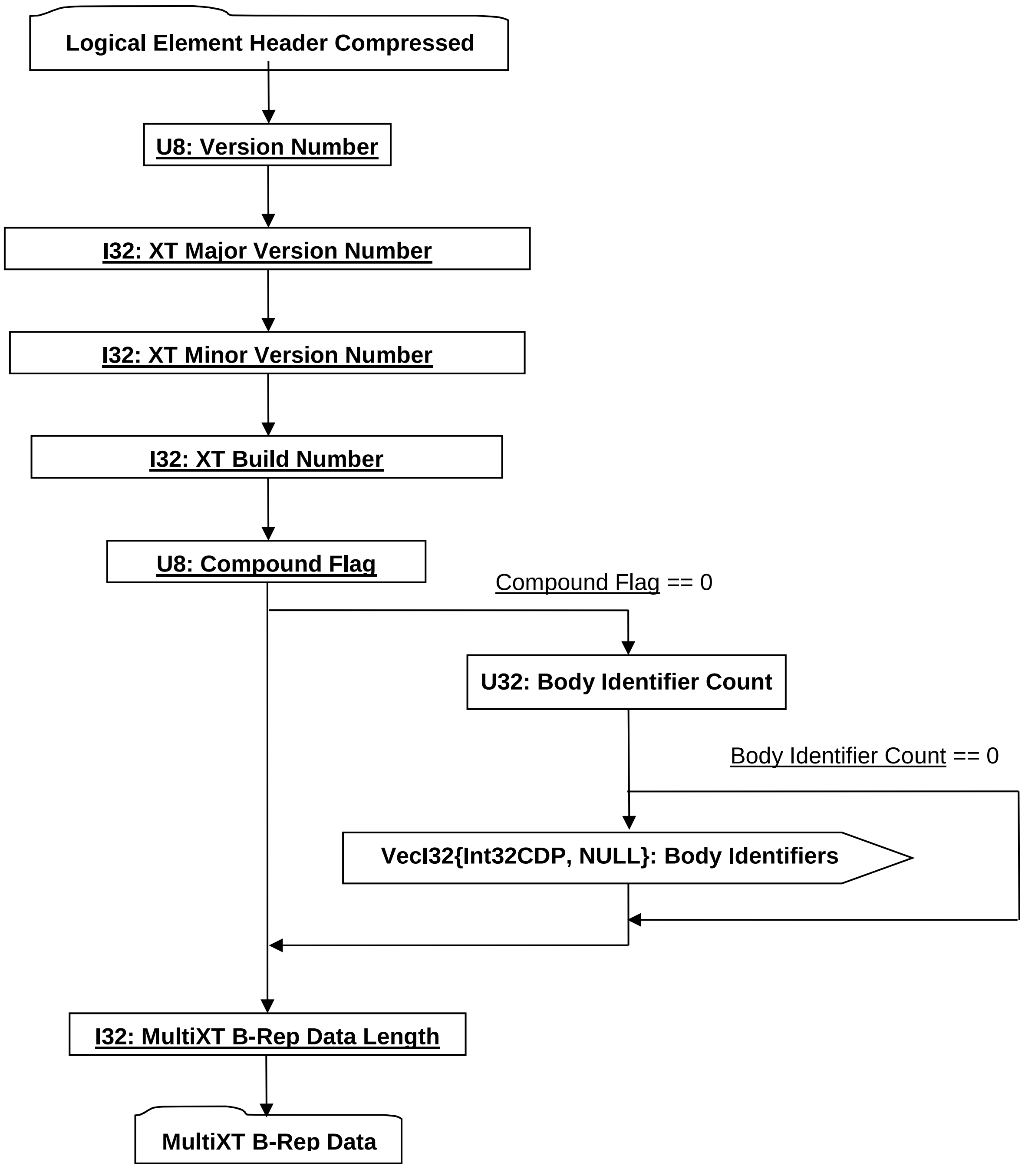


Figure 159 — MultiXT B-Rep Element data collection

U8: Version Number

Version Number is the version identifier for this XT B-Rep Element. For information on local version numbers see Common Data Conventions and Construct Local version numbers.

I32: XT Major Version Number

XT Major Version Number specifies the major version number for the for the XT B-Rep data in the JT File. Major version number is an informative field which can be set to 0 without negative impact to implementation.

I32: XT Minor Version Number

XT Minor Version Number specifies the minor version number for the XT B-Rep data in the JT File. Minor version number is an informative field which can be set to 0 without negative impact to implementation.

I32: XT Build Number

XT Build Number specifies the build number for the XT B-Rep data in the JT File. XT build number is an informative field which can be set to 0 without negative impact to implementation.

U8: Compound Flag

Compound Flag specifies if the XT boundary representation has been turned into compound type. If the compound flag is not zero, then the XT B-Rep Body read from MultiXT B-Rep Data is of compound type. The compound body can be decomposed into multiple XT B-Rep bodies, and a Parasolid entity identifier can be retrieved from each Parasolid body using Parasolid library. The retrieved XT entity identifiers can be used to formulate Body Identifiers array, which is explicitly stored when the compound flag is zero, in order for each Part that refers to this MultiXT B-Rep Segment to know its XT B-Rep body(s).

U32: Body Identifier Count

Body Identifier Count specifies the number of entity identifiers in this MultiXT B-Rep Element.

VecI32{Int32CDP, NULL} : Body Identifiers

Body Identifiers is an integer array with its length equal to Body Identifier Count. The array is parallel to the XT B-Rep Body array read from MultiXT B-Rep Data, and each element specifices the identifier for the corresponding XT body.

I32: MultiXT B-Rep Data Length

MultiXT B-Rep Data Length specifies the length in bytes of the MultiXT B-Rep Data collection. A JT file loader/reader may use this information to compute the end position of the MultiXT B-Rep Data within the JT file and thus skip (for whatever reason) reading the remaining MultiXT B-Rep Data.

MultiXT B-Rep Data

The MultiXT B-Rep Data collection specifies the raw stream of bytes used to represent XT B-Rep Body(s), that are the super set of all the XT B-Rep Body(s) in all the Parts that refer to this MultiXT B-Rep Segment.

F.2 XT B-Rep Data Segment Description

The XT B-Rep data segment in a JT file is a neutral binary definition of the solid body. A proprietary kernel is not required to read the XT B-Rep data segment. The model definition in the XT B-Rep data segment of a JT file is a fully described geometric and topological representation.

F.2.1 Logical Layout

The logical layout of a XT B- Rep data segment is:

- A short flag sequence describing the data format, followed by modeller identification information and user field size.
    - The various flag sequences (mixtures of text and numbers) are documented under `Physical layout'.
    - The content of the modeller identification information is:

- The version of the Parasolid Kernel (if applicable) used to write the data, as a text string of the form:
- TRANSMIT FILE created by modeler version 30000. The schema version describing the field sequences of the part nodes as a text string of the form:
- SCH_3000000_30000
- This example above denotes XT data written by the Parasolid Kernel V30.0.000 using schema number 30000.
- Applications writing XT data segments should use version 3000000 and schema number 30000 (i.e.):
- TRANSMIT FILE created by modeler version 30000 SCH_3000000_30000
- The user field size is a simple integer.

— The objects (known as 'nodes') in the XT data in an unordered sequence, followed by a terminator.

- Every node in the XT data is assigned an integer index from 1 upwards (some indices may not be used). Pointer fields are output as these indices, or as zero for a null pointer.
- Each node entry begins with the node type. If the node is of variable length (see below), this is followed by the length of the variable field. The index of the node is then output, followed by the fields of the node.
- The terminator which follows the sequence of nodes is a two-byte integer with value 1, followed by an index with value 0. The index is output as a 2-byte integer with value 1 in binary XT data.
- The node with index 1 is the root node of the data as follows:

Table 87 — Object Nodes

Contents of XT data	Type of root node
Body	BODY
Assembly	ASSEMBLY
Array of parts	POINTER_LIS_BLOCK
Partition	WORLD

F.2.1.1 Schema

XT permanent structures are defined in a language akin to C which generates the appropriate files for a C compiler, the runtime information used by the Parasolid Kernel, along with an optional schema file that can be used during transmit and receive. The schema file for version 30000 is named SCH_3000000_30000 and is distributed with the Parasolid kernel. However, it is not necessary for applications reading and writing XT data directly to have a copy of this schema file to understand the XT.

F.2.1.2 Embedded schemas

XT parts, partitions and deltas can be transmitted with extra information that is intended to replace the schema used to describe the data layout. This information contains the differences between its schema and a defined base schema. The only fields that are included in this information are those which can be referenced in a cut-down version of the schema pertaining only to the XT part data that is present. Specifically, a full schema definition can contain fields that are not relevant in the context of the transmitted data and these fields are excluded.

Fields that are included are referred to as effective fields, and are either transmittable (`xmt_code` == 1) or have variable-length (`n_elts` == 1).

F.2.1.3 Physical layout

Most of the XT data are composed of integers, logical flags, and strings, but are of restricted ranges and so transmitted specially in binary format. The binary representation is given in bold type, such as “integer (**byte**)”. This is relevant to applications that attempt to read or write XT data directly. Two important elements are

- short strings

These are transmitted as an integer length (byte) followed by the characters (without trailing zero).

- positive integers

These are transmitted similarly to the pointer indices which link individual objects together, i.e., small values 0..32766 are transmitted as a single **short** integer, larger ones encoded into two.

F.2.2 XT format

Presence of the new format is indicated by a change to the standard header: the archive name is extended by the number of the base schema, e.g., `SCH_1400068_14000_13006`, and then the maximum number of node types is inserted (short).

Transmission then continues as normal, except that when transmitting the first node of any particular type, extra information is inserted between the nodetype and the variable-length, index data as follows:

— The arrays of effective fields in the base schema node and the current schema node are assembled.

— If the nodetype does not exist in the base schema then it is output as follows:

– number of fields (byte),

– name and description (short strings), and

– fields one by one as shown in the Table 2.

Table 88 — Field types in order one by one

Name	Short String	Notes
ptr_class	Short	
n_elts	Positive integer	
type	short string	The field type. Allowed values are described in “Field types”, below. Omitted if ptr_class non-zero
xmt_code	logical (byte)	Omitted for fixed-length (n_elts != 1)

— If the two arrays match (equal length and all fields match in `name`, `xmt_code`, `ptr_class`, `n_elts` and `type`) then output the flag value 255 (**byte** 0xff).

— If the two arrays do not match, output the number of effective fields in the current schema (byte), and an edit sequence as follows:

– Initialize pointers to the first base field and first current field, then while there are still unprocessed base and current fields, output a sequence of Copy, Delete and Insert instructions.

  – If the base field matches the current field, output 'C' (char) to indicate an unchanged (Copied) field and advance to the next base and current fields.

  – If the base field does not match any unprocessed current field, output 'D' (char) to indicate a Deleted field and advance to the next base field.

  – Otherwise, output 'I' (char) to indicate an Inserted field, followed by the current field in the above format, and advance to the next current field.

– If there are any unprocessed current fields, then output an Append sequence, each instruction being 'A' (**char**) followed by the field.

— Finally, output 'Z' (**char**) to signal the end.

F.2.2.1 Field types

The XT format is not itself a binary protocol, and so does not define data sizes; the only requirement is that a runtime implementation has sufficient room for the information. The available implementations run with 8bit ASCII characters, 8bit unsigned bytes (0..255), 16bit short integers (0..65535 or -32768..32767), 32bit integers (0..4G-1, -2G..2G-1) and IEEE reals. The implementation used in the given binary XT data is specified by the "PS<code>" at the start of the XT data. See the chapter on “Physical Layout” for more information.

The full list of field types used in XT segment data is as follows:

u unsigned byte 0-255

c char

l unsigned byte 0-1 (i.e. logical)

```
[1]           typedef char logical;
```

n short int

w unicode character, output as a short int

d Int

p pointer-index

Small indices (less than 32767) are treated specially in binary XT data to save space. See the section below on binary format.

f double

i These correspond to a region of the real line:

```
typedef struct { double low, high; }interval;
```

v array [3] of doubles

These correspond to a 3-space position or direction:

```
typedef struct { double x,y,z; } vector;
```

b array [6] of doubles

These correspond to a 3-spce region:

```
typedef struct { interval x,y,z; } box;
```

h array [3] of doubles

These represent points of intersection between two surfaces; only the position vector is written to the XT data. The structure is documented further in the section on intersection curves.

F.2.2.2 Variable-length nodes

Variable-length nodes differ from fixed-length nodes in that their last field is of variable length, i.e. different nodes of the same type may have different lengths.

The number of entries in each such node is indicated by an integer in the XT data between its nodetype and index, so an example might be

```
83 3 15 1 2 3
```

F.2.2.3 Unresolved indices

In some cases a node will contain an index field which does not correspond to a node in the XT data, in this case the index is to be interpreted as zero.

F.2.3 Physical Layout

Binary

The flag sequence is followed by the length of the modeller version as a 2-byte integer, the characters of the modeller version, the length of the schema version as a 4-byte integer, the characters of the schema version, and finally the userfield size as a 4-byte integer.

There are two special numeric values (-32764 for integral values, -3.14158e13 for floating point) which are used to mark an ‘unset’ or ‘null’ value.

Neutral Binary

In the XT Segment neutral binary, data is represented in big-endian format, with IEEE floating point numbers and ASCII characters. The flag sequence is the 4-byte sequence "PS" followed by two zero bytes, i.e., 'P' 'S' '\0' '\0'. The initial letters are ASCII, thus '\120' '\123'.The nodetype at the start of a node is a 2-byte integer, the variable length which may follow it is a 4-byte integer.

Logical values (0,1) are represented as themselves in 1 byte.

Small pointer indices (in the range 0-32766) are implemented as a 2-byte integer, larger indices are represented as a pair, thus:

```
if (index < 32767)
      {                                // case: small index
      op_short( index + 1 );           // offset so is > 0
      }
else
      {                                // case: big index
      op_short( -(index % 32767 +      // remainder: add 1 so > 0
      1) );
      op_short( index / 32767 );       // nonzero quotient
```

```
        }
```

where op_short outputs a 2-byte integer.

The inverse is performed on reading:

```
    short q = 0, r;
    ip_short( &r );
    if (r < 0)
        {
        ip_short( &q );
        r = -r;
        }
    index = q * 32767 + r - 1;
```

where ip_short reads a 2-byte integer.

F.2.4 Model Structure

F.2.4.1 Topology

This section describes the XT Topology model, it gives an overview of how the nodes in the XT data are joined together.

The topological representation allows for:

- non-manifold solids,
- solids with internal partitions,
- bodies of mixed dimension (i.e. with wire, sheet, and solid `bits'),
- pure wire-frame bodies, and
- disconnected bodies.

Each entity is described, and its properties and links to other entities given.

General points

In this section a set is called finite if it can be enclosed in a ball of finite radius - not that it has a finite number of members.

A set of points in 3-dimensional space is called open if it does not contain its boundary.

Back-pointers, next and previous pointers in a chain, and derived pointers are not described explicitly here. For information on this see the following description of the schema-level model.

F.2.4.2 Entity definitions

Assembly

An assembly is a collection of instances of bodies or assemblies. It may also contain construction geometry. An assembly has the following fields:

- a set of instances, and
- a set of geometry (surfaces, curves and points).

Instance

An instance is a reference to a body or an assembly, with an optional transform:

- Body or assembly.
- Transform. If null, the identity transform is assumed.

Body

A body is a collection of faces, edges and vertices, together with the 3-dimensional connected regions into which space is divided by these entities. Each region is either **solid** or **void** (indicating whether it represents material or not).

The point-set represented by the body is the disjoint union of the point-sets represented by its solid regions, faces, edges, and vertices. This point-set need not be connected, but it shall be finite.

A body has the following fields:

- A set of regions.

  A body has one or more regions. These, together with their boundaries, make up the whole of 3-space, and do not overlap, except at their boundaries. One region in the body is distinguished as the exterior region, which shall be infinite; all other regions in the body shall be finite.

- A set of geometry (surfaces, curve and/or points).
- A body-type. This may be wire, sheet, solid or general.

Region

A region is an open connected subset of 3-dimensional space whose boundary is a collection of vertices, edges, and oriented faces.

Regions are either solid or void, and they may be non-manifold. A solid region contributes to the point-set of its owning body; a void region does not (although its boundary will).

Two regions may share a face, one on each side.

A region may be infinite, but a body shall have exactly one infinite region. The infinite region of a body shall be void.

A region has the following fields:

- A logical indicating whether the region is solid.
- A set of shells. The positive shell of a region, if it has one, is not distinguished.

The shells of a region do not overlap or share faces, edges or vertices.

A region may have no shells, in which case it represents all space (and will be the only region in its body, which will have no faces, edges or vertices).

Shell

A shell is a connected component of the boundary of a region. As such it will be defined by a collection of faces, each used by the shell on one `side', or on both sides; and some edges and vertices.

A shell has the following fields:

- A set of (face, logical) pairs.

  Each pair represents one side of a face (where true indicates the front of the face, i.e. the side towards which the face normal points), and means that the region to which the shell belongs lies on that side of the face. The same face may appear twice in the shell (once with each orientation), in which case the face is a 2-dimensional cut subtracted from the region which owns the shell.

- A set of wireframe edges.

  Edges are called **wireframe** if they do not bound any faces, and so represent 1-dimensional cuts in the shell's region. These edges are not shared by other shells.

- A vertex.

  This is only non-null if the shell is an **acorn** shell, i.e. it represents a 0-dimensional hole in its region, and has one vertex, no edges and no faces.

A shell shall contain at least one vertex, edge, or face.

Face

A face is an open finite connected subset of a surface, whose boundary is a collection of edges and vertices. It is the 2-dimensional analogy of a region.

A face has the following fields:

- A set of loops. A face may have zero loops (e.g. a full spherical face), or any number.

- Surface. This may be null, and may be used by other faces.

- Sense. This logical indicates whether the normal to the face is aligned with or opposed to that of the surface.

Loop

A loop is a connected component of the boundary of a face. It is the 2-dimensional analogy of a shell. As such it will be defined by a collection of fins and a collection of vertices.

A loop has the following fields:

- An ordered ring of fins.

  Each fin represents the oriented use of an edge by a loop. The sense of the fin indicates whether the loop direction and the edge direction agree or disagree. A loop may not contain the same edge more than once in each direction.

  The ordering of the fins represents the way in which their owning edges are connected to each other via common vertices in the loop (i.e. nose to tail, taking the sense of each fin into account).

  The loop direction is such that the face is locally on the left of the loop, as seen from above the face and looking in the direction of the loop.

- A vertex.

This is only non-null if the loop is an isolated loop, i.e. has no fins and represents a 0-dimensional hole in the face.

Consequently, a loop shall consist either of:

- A single fin whose owning ring edge has no vertices, or
- At least one fin and at least one vertex, or
- A single vertex.

Fin

A fin represents the oriented use of an edge by a loop.

A fin has the following fields:

- A logical **sense** indicating whether the fin's orientation (and thus the orientation of its owning loop) is the same as that of its owning edge, or different.
- A curve. This is only non-null if the fin’s edge is tolerant, in which case every fin of that edge will reference a trimmed SP-curve. The underlying surface of the SP-curve shall be the same as that of the corresponding face. The curve shall not deviate by more than the edge tolerance from curves on other fins of the edge, and its ends shall be within vertex tolerance of the corresponding vertices.

Edge

An edge is an open finite connected subset of a curve; its boundary is a collection of zero, one or two vertices. It is the 1-dimensional analogy of a region.

An edge has the following fields:

- Start vertex.
- End vertex. If one vertex is null, then so is the other; the edge will then be called a **ring** edge.
- An ordered ring of distinct fins.
- The ordering of the fins represents the spatial ordering of their owning faces about the edge (with a right-hand screw rule, i.e. looking in the direction of the edge the fin ordering is clockwise). The edge may have zero or any number of fins; if it has none, it is called a **wireframe** edge.
- A curve. This will be null if the edge has a tolerance. Otherwise, the vertices shall lie within vertex tolerance of this curve, and if it is a Trimmed Curve, they shall lie within vertex tolerance of the corresponding ends of the curve. The curve shall also lie in the surfaces of the faces of the edge, to within modeller resolution.
- Sense. This logical indicates whether the direction of the edge (start to end) is the same as that of the curve.
- A tolerance. If this is null-double, the edge is **accurate** and is regarded as having a tolerance of half the modeller linear resolution, otherwise the edge is called **tolerant**.

Vertex

A vertex represents a point in space. It is the 0-dimensional analogy of a region.

A vertex has the following fields:

- A geometric point.
- A tolerance. If this is null-double, the vertex is **accurate** and is regarded as having a tolerance of half the modeller linear resolution.

Attributes

An attribute is an entity which contains data, and which can be attached to any other entity except attributes, fins, lists, transforms or attribute definitions. An attribute has the following fields:

- Definition. An attribute definition is an entity which defines the number and type of the data fields in a specific type of attribute, which entities may have such an attribute attached, and what happens to the attribute when its owning entity is changed. XT data shall not contain duplicate attribute definitions. Each attribute of a given type should reference the same instance of the attribute definition for that type. It is incorrect, for example, to create a copy of an attribute definition for each instance of the attribute of that type. Only those attribute definitions referenced by attributes in the part occur in the data.
- Owner.
- Fields. These are data fields consisting of one or more integers, doubles, vectors etc.

There are a number of system attribute definitions which may be present in the XT data. These are documented in the section `System Attribute Definitions'. User attribute definitions can also be created. These are included in the XT data along with any attributes that use them.

Groups

A group is a collection of entities in the same part. Groups in assemblies may contain instances, surfaces, curves and points. Groups in bodies may contain regions, faces, edges, vertices, surfaces, curves and points. Groups have:

- Owning part.
- A set of member entities.
- Type. The type of the group specifies the allowed type of its members, e.g. a ‘face’ group in a body may only contain faces, whereas a ‘mixed’ group may have any valid members.

Node-ids

All entities in a part, other than fins, have a non-zero integer node-id which is unique within a part. This is intended to enable the entity to be identified within the XT data.

Entity matrix

Thus the relations between entities can be represented in matrix form as follows in Table 3. The numbers represent the number of distinct entities connected (either directly or indirectly) to the given one.

Table 89 — Entity Matrix relations

	Body	Region	Shell	Face	Loop	Fin	Edge	Vertex
Body	-	>0	any	any	any	any	any	any
Region	1	-	any	any	any	any	any	any
Shell	1	1	-	any	any	any	any	any

Face	1	1-2	1-2	-	any	any	any	any
Loop	1	1-2	1-2	1	-	any	any	any
Fin	1	1-2	1-2	1	1	-	1	0-2
Edge	1	any	any	any	any	any	any	any
Vertex	1	any	any	any	any	any	any	-

F.2.4.3 Representation of manifold bodies

F.2.4.3.1 Body types

XT bodies have a field body_type which takes values from an enumeration indicating whether the body is:

- **solid**, representing a manifold 3-dimensional volume, possibly with internal voids. It need not be connected;
- **sheet**, representing a 2-dimensional subset of 3-space which is either manifold or manifold with boundary (certain cases are not strictly manifold – see below for details). It need not be connected;
- **wire**, representing a 1-dimensional subset of 3-space which is either manifold or manifold with boundary, and which need not be connected. An **acorn** body, which represents a single 0-dimensional point in space, also has body-type wire;
- **general** - none of the above.

A general body is not necessarily non-manifold, but at the same time it is not constrained to be manifold, connected, or of a particular dimensionality (indeed, it may be of mixed dimensionality).

Restrictions on entity relationships for manifold body types

Solid, sheet, and wire bodies are best regarded as special cases of the topological model; for convenience we call them the manifold body types (although as stated above, a general body may also be manifold).

In particular, bodies of these manifold types shall obey the following constraints:

- An acorn body shall consist of a single void region with a single shell consisting of a single vertex.
- A wire body shall consist of a single void region, with one or more shells, consisting of one or more wireframe edges and zero or more vertices (and no faces). Every vertex in the body shall be used by exactly one or two of the edges (so, in particular, there are no acorn vertices).
- So each connected component will be either: closed, where every vertex has exactly two edges; or open, where all but two vertices have exactly two edges each.
- A wire is called open if all its components are open, and closed if all its components are closed
- Solid and sheet bodies shall each contain at least one face; they may not contain any wireframe edges or acorn vertices.
- A solid body shall consist of at least two regions; at least one of its regions shall be solid. Every face in a solid body shall have a solid region on its negative side and a void region on its positive side (in other words, every face forms part of the boundary of the solid, and the face normals always point away from the solid).

— Every edge in a solid body shall have exactly two fins, which will have opposite senses. Every vertex in a solid body shall either belong to a single isolated loop, or belong to one or more edges; in the latter case, the faces which use those edges shall form a single edgewise-connected set (when considering only connections via the edges which meet at the vertex).

— These constraints ensure that the solid is manifold.

— All the regions of a sheet body shall be void. It is known as an open sheet if it has one region, and a closed sheet if it has no boundary.

— Every edge in a sheet body shall have exactly one or two fins; if it has two, these shall have opposite senses. In a closed sheet body, all the edges will have exactly two fins. Every vertex in a sheet body shall either belong to a single isolated loop, or belong to one or more edges; in the latter case, the faces which use those edges shall either form a single edgewise-connected set where all the edges involved have exactly two fins, or any number of edgewise-connected sets, each of which shall involve exactly two edges with one fin each (again, considering only connections via the edges which meet at the vertex).

Note that, although the constraints on edges and vertices in a sheet body are very similar to those which apply to a solid, in this case they do not guarantee that the body will be manifold; indeed, the rather complicated rules about vertices in an open sheet body specifically allow bodies which are non-manifold (such as a body consisting of two square faces which share a single corner vertex).

F.2.4.3.2 Schema Definition

Underlying types

```
union CURVE_OWNER_u
      {
      struct EDGE_s               *edge;
      struct FIN_s                *fin;
      struct BODY_s               *body;
      struct ASSEMBLY_s           *assembly;
      struct WORLD_s              *world;
      };

union SURFACE_OWNER_u
      {
      struct FACE_s               *face;
      struct BODY_s               *body;
      struct ASSEMBLY_s           *assembly;
      struct WORLD_s              *world;
      };

union ATTRIB_GROUP_u
      {
      struct ATTRIBUTE_s          *attribute;
      struct GROUP_s              *group;
      struct MEMBER_OF_GROUP_s    *member_of_group;
      };
typedef union ATTRIB_GROUP_u  ATTRIB_GROUP;
```

F.2.4.3.3 Geometry

```
union CURVE_u
      {
      struct LINE_s               *line;
      struct CIRCLE_s             *circle;
      struct ELLIPSE_s            *ellipse;
      struct INTERSECTION_s       *intersection;
```

```
      struct TRIMMED_CURVE_s       *trimmed_curve;
      struct PE_CURVE_s            *pe_curve;
      struct B_CURVE_s             *b_curve;
      struct SP_CURVE_s            *sp_curve;
      };
typedef union CURVE_u      CURVE;

union SURFACE_u
      {
      struct PLANE_s               *plane;
      struct CYLINDER_s            *cylinder;
      struct CONE_s                *cone;
      struct SPHERE_s              *sphere;
      struct TORUS_s               *torus;
      struct BLENDED_EDGE_s        *blended_edge;
      struct BLEND_BOUND_s         *blend_bound;
      struct OFFSET_SURF_s         *offset_surf;
      struct SWEPT_SURF_s          *swept_surf;
      struct SPUN_SURF_s           *spun_surf;
      struct PE_SURF_s             *pe_surf;
      struct B_SURFACE_s           *b_surface;
      };
typedef union SURFACE_u    SURFACE;

union GEOMETRY_u
      {
      union  SURFACE_u              surface;
      union  CURVE_u                curve;
      struct POINT_s               *point;
      struct TRANSFORM_s           *transform;
      };
typedef union GEOMETRY_u GEOMETRY;
```

Curves

In the following field tables, 'pointer0' means a reference to another node which may be null. 'pointer' means a non-null reference.

All curve nodes share the following common fields:

Table 90 — Curve node common fields

Field name	Data type	Description
node_id	int	Integer value unique to curve in part
attributes_groups	pointer0	Attributes and groups associated with curve
owner	pointer0	topological owner
next	pointer0	next curve in geometry chain
previous	pointer0	previous curve in geometry chain
geometric_owner	pointer0	geometric owner node
sense	char	sense of curve: '+' or '-' (see end of Geometry section)

```
struct ANY_CURVE_s               // Any Curve
     {
     int                           node_id;                   // $d
     union  ATTRIB_GROUP_u         attributes_groups;         // $p
     union  CURVE_OWNER_u          owner;                     // $p
     union  CURVE_u                next;                      // $p
```

```
    union  CURVE_u                previous;                     // $p
    struct GEOMETRIC_OWNER_s     *geometric_owner;              // $p
    char                          sense;                        // $c
    };
typedef struct ANY_CURVE_s *ANY_CURVE;
```

Line

A straight line has a parametric representation of the form:

R(t) = P + t D

where

— P is a point on the line.

— D is its direction.

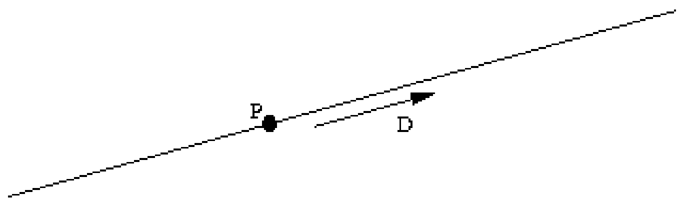


Table 91 — Line Fields

Field name	Data type	Description
pvec	vector	point on the line
direction	vector	direction of the line (a unit vector)

```
struct LINE_s == ANY_CURVE_s      // Straight line
    {
    int                           node_id;                      // $d
    union  ATTRIB_GROUP_u         attributes_groups;            // $p
    union  CURVE_OWNER_u          owner;                        // $p
    union  CURVE_u                next;                         // $p
    union  CURVE_u                previous;                     // $p
    struct GEOMETRIC_OWNER_s     *geometric_ owner;             // $p
    char                          sense;                        // $c
    vector                        pvec;                         // $v
    vector                        direction;                    // $v
    };
typedef struct LINE_s      *LINE;
```

CIRCLE

A circle has a parametric representation of the form

R(t) = C+ r X cos(t) + r Y sin(t)

Where

— C is the centre of the circle.

— r is the radius of the circle.

— X and Y are the axes in the plane of the circle.

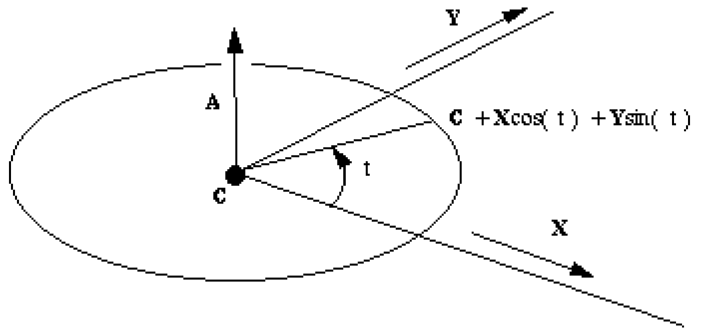


Table 92 — Circle fields

Field name	Data type	Description
centre	vector	Centre of circle
normal	vector	Normal to the plane containing the circle (a unit vector)
x_axis	vector	X axis in the plane of the circle (a unit vector)
radius	double	Radius of circle

The Y axis in the definition above is the vector cross product of the normal and x_axis.

```
struct CIRCLE_s == ANY_CURVE_s                        // Circle
    {
    int                            node_id;                      // $d
    union  ATTRIB_GROUP_u          attributes_groups;            // $p
    union  CURVE_OWNER_u           owner;                        // $p
    union  CURVE_u                 next;                         // $p
    union  CURVE_u                 previous;                     // $p
    struct GEOMETRIC_OWNER_s      *geometric_owner;              // $p
    char                           sense;                        // $c
    vector                         centre;                       // $v
    vector                         normal;                       // $v
    vector                         x_axis;                       // $v
    double                         radius;                       // $f
    };
typedef struct CIRCLE_s    *CIRCLE;
```

ELLIPSE

An ellipse has a parametric representation of the form

R(t) = C+ a X cos(t) + b Y sin(t)

where

— C is the centre of the circle.

— X is the major axis.

— r is the major radius.

— Y and b are the minor axis and minor radius respectively.

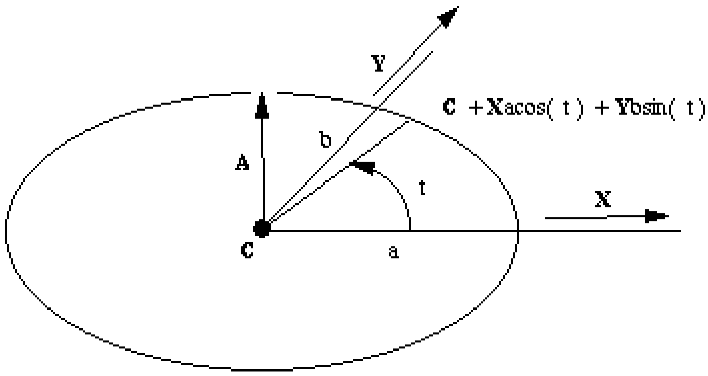


Table 93 — Ellipse fields

Field name	Data type	Description
centre	Vector	Centre of ellipse
normal	Vector	Normal to the plane containing the ellipse (a unit vector)
x_axis	Vector	major axis in the plane of the ellipse (a unit vector)
major_radius	Double	major radius
minor_radius	Double	minor radius

The minor axis (Y) in the definition above is the vector cross product of the normal and x_axis.

```
struct ELLIPSE_s == ANY_CURVE_s      // Ellipse
    {
    int                             node_id;                   // $d
    union  ATTRIB_GROUP_u           attributes_groups;         // $p
    union  CURVE_OWNER_u            owner;                     // $p
    union  CURVE_u                  next;                      // $p
    union  CURVE_u                  previous;                  // $p
    struct GEOMETRIC_OWNER_s       *geometric_owner;           // $p
    vector                          centre;                    // $v
    char                            sense;                     // $c
    vector                          normal;                    // $v
    vector                          x_axis;                    // $v
    double                          major_radius;              // $f
    double                          minor_radius;              // $f
    };
typedef struct ELLIPSE_s   *ELLIPSE;
```

B_CURVE (B-spline curve)

XT supports B spline curves in full NURBS format. The mathematical description of these curves is:

— Non Uniform Rational B-splines as (NURBS), and

$$P(t) = \frac{\sum_{i=0}^{n-1} b_i(t) w_i V_i}{\sum_{i=0}^{n-1} b_i(t) w_i}$$

— the more simple Non Uniform B-spline

$$P(t) = \sum_{i=0}^{n-1} b_i(t) V_i$$

— Where:

n = number of vertices (n_vertices in the PK standard form)

$V_0 \ldots V_{n-1}$ are the B-spline vertices

$w_0 \ldots w_{n-1}$ are the weights

$b_i(t), I = 0 \ldots n-1$ are the B-spline basis functions

Knot Vectors

The parameter t above is global. The user supplies an ordered set of values of t at specific points. The points are called knots and the set of values of t is called the knot vector. Each successive value in the set shall be greater than or equal to its predecessor. Where two or more such values are the same we say that the knots are coincident, or that the knot has multiplicity greater than 1. In this case it is best to think of the knot set as containing a null or zero length span. The principal use of coincident knots is to allow the curve to have less continuity at that point than is formally required for a spline. A curve with a knot of multiplicity equal to its *degree* can have a discontinuity of first derivative and hence of tangent direction. This is the highest permitted multiplicity except at the first or last knot where it can go as high as *(degree+1)*.

In order to avoid problems associated, for example with rounding errors in the knot set, XT stores an array of distinct values and an array of integer multiplicities. This is reflected in the standard form used by the PK for input and output of B-curve data.

Most algorithms in the literature, and the following discussion refer to the expanded knot set in which a knot of multiplicity n appears explicitly n times.

THE NUMBER OF KNOTS AND VERTICES

The knot set determines a set of basis functions which are bell shaped, and non zero over a span of *(degree+1)* intervals. One basis function starts at each knot, and each one finishes *(degree +1)* knots higher. The control vectors are the coefficients applied to these basis functions in a linear sum to obtain positions on the curve. Thus it can be seen that we require the number of knots n_knots = n_vertices + degree + 1.

THE VALID RANGE OF THE B-CURVE

So if the knot set is numbered $\{t_0$ to $t_{n\_knots-1}\}$ it can be seen then that it is only after t_{degree} that sufficient (*degree + 1*) basis functions are present for the curve to be fully defined, and that the B-curve ceases to be fully defined after $t_{n\_knots - 1 - degree}$.

The first *degree* knots and the last *degree* knots are known as the imaginary knots because their parameter values are outside the defined range of the B-curve.

PERIODIC B-CURVES

When the end of a B-curve meets its start sufficiently smoothly XT allows it to be defined to have periodic parametrization. That is to say that if the valid range were from t_{degree} to $t_{n\_knots - 1 - degree}$ then the difference between these values is called the period and the curve can continue to be evaluated with the same point reoccurring every period.

The minimal smoothness requirement for periodic curves in XT is tangent continuity, but we strongly recommend $C_{degree-1}$, or continuity in the (*degree-1*)th derivative. This in turn is best achieved by repeating the first *degree* vertices at the end, and by matching knot intervals so that counting from the start of the defined range, t_{degree}, the first *degree* intervals between knots match the last *degree* intervals, and similarly matching the last *degree* knot intervals before the end of the defined range to the first *degree* intervals.

CLOSED B-CURVES

A periodic B-curve shall also be closed, but is permitted to have a closed Bcurve that is not periodic.

In this case the rules for continuity are relaxed so that only C_0 or positional continuity is required between the start and end. Such closed non-periodic curves are not able to be attached to topology.

RATIONAL B-CURVE

In the rational form of the curve, each vertex is associated with a weight, which increases or decreases the effect of the vertex without changing the curve hull. To ensure that the convex hull property is retained, the curve equation is divided by a denominator which makes the coefficients of the vertices sum to one.

$$P(t) = \frac{\sum_{i=0}^{n-1} b_i(t) w_i V_i}{\sum_{i=0}^{n-1} b_i(t) w_i}$$

Where $w_0 \ldots w_{n-1}$ are weights.

Each weight may take any positive value, and the larger the value, the greater the effect of the associated vertex. However, it is the relative sizes of the weights which is important, as may be seen from the fact that in the equation given above, all the weights may be multiplied by a constant without changing the equation.

In XT the weights are stored with the vertices by treating these as having an extra dimension. In the usual case of a curve in 3-d cartesian space this means that vertex_dim is 4, the x, y, z values are multiplied through by the corresponding weight and the 4th value is the weight itself.

```
struct B_CURVE_s == ANY_CURVE_s                     // B curve
    {
    int                            node_id;                    // $d
    union  ATTRIB_GROUP_u          attributes_groups;          // $p
    union  CURVE_OWNER_u           owner;                      // $p
    union  CURVE_u                 next;                       // $p
    union  CURVE_u                 previous;                   // $p
    struct GEOMETRIC_OWNER_s      *geometric_owner;            // $p
    char                           sense;                      // $c
    struct NURBS_CURVE_s          *nurbs;                      // $p
    struct CURVE_DATA_s           *data;                       // $p
    };
typedef struct B_CURVE_s       *B_CURVE;
```

The data stored in the XT data for a NURBS_CURVE is

Table 94 — NURB curve fields

Field name	Data type	Description
degree	Short	degree of the curve
n_vertices	Int	number of control vertices (‘poles’)
vertex_dim	Short	dimension of control vertices
n_knots	Int	number of distinct knots
knot_type	Byte	form of knot vector
periodic	Logical	true if curve is periodic
closed	Logical	true if curve is closed
rational	Logical	true if curve is rational
curve_form	Byte	shape of curve, if special
bspline_vertices	Pointer	control vertices node
knot_mult	Pointer	knot multiplicities node
knots	Pointer	knots node

The knot_type enum is used to describe whether or not the knot vector has a certain regular spacing or other common property:

```
typedef enum
    {
    SCH_unset = 1,                // Unknown
    SCH_non_uniform = 2,          // Known to be not special
    SCH_uniform = 3,              // Uniform knot set
    SCH_quasi_uniform = 4,        // Uniform apart from bezier ends
    SCH_piecewise_bezier = 5,     // Internal multiplicity of order-1
    SCH_bezier_ends = 6           // Bezier ends, no other property
    }
     SCH_knot_type_t;
```

A uniform knot set is one where all the knots are of multiplicity one and are equally spaced. A curve has bezier ends if the first and last knots both have multiplicity ‘order’.

The curve_form enum describes the geometric shape of the curve. The parameterization of the curve is not relevant.

```
typedef enum
    {
    SCH_unset           = 1,     // Form is not known
    SCH_arbitrary       = 2,     // Known to be of no particular shape
    SCH_polyline        = 3,
    SCH_circular_arc    = 4,
    SCH_elliptic_arc    = 5,
    SCH_parabolic_arc   = 6,
    SCH_hyperbolic_arc = 7
    }
     SCH_curve_form_t;

struct NURBS_CURVE_s                                  // NURBS curve
    {
    short                          degree;                    // $n
    int                            n_vertices;                // $d
    short                          vertex_dim;                // $n
    int                            n_knots;                   // $d
    SCH_knot_type_t                knot_type;                 // $u
    logical                        periodic;                  // $l
    logical                        closed;                    // $l
    logical                        rational;                  // $l
    SCH_curve_form_t               curve_form;                // $u
    struct BSPLINE_VERTICES_s     *bspline_vertices;          // $p
    struct KNOT_MULT_s            *knot_mult;                 // $p
```

```
    struct KNOT_SET_s              *knots;                      // $p
    };
typedef struct NURBS_CURVE_s *NURBS_CURVE;
```

The bspline vertices node is simply an array of doubles; ‘vertex_dim’ doubles together define one control vertex. Thus the length of the array is n_vertices * vertex_dim.

```
struct BSPLINE_VERTICES_s              // B-spline vertices
    {
    double                   vertices[ 1 ];                // $f[]
    };
typedef struct BSPLINE_VERTICES_s *BSPLINE_VERTICES;
```

The knot vector of the NURBS _CURVE is stored as an array of distinct knots and an array describing the multiplicity of each distinct knot. Hence the two nodes

```
struct KNOT_SET_s                      // Knot set
    {
    double                   knots[ 1 ];                   // $f[]
    };
typedef struct KNOT_SET_s *KNOT_SET;
```

and

```
struct KNOT_MULT_s                     // Knot multiplicities
    {
    short                    mult[ 1 ];                    // $n[]
    };
typedef struct KNOT_MULT_s *KNOT_MULT;
```

The data stored in the XT data for a CURVE_DATA node is:

```
typedef enum
    {
    SCH_unset = 1,                         // check has not been performed
    SCH_no_self_intersections = 2,         // passed checks
    SCH_self_intersects = 3,               // fails checks
    SCH_checked_ok_in_old_version = 4      // see below
    }
    SCH_self_int_t;

struct CURVE_DATA_s                    // curve_data
     {
     SCH_self_int_t            self_int;                    // $u
     Struct HELIX_CU_FORM_s    *analytic_form               // $p
     };
typedef struct CURVE_DATA_s *CURVE_DATA;
```

The self-intersection enum describes whether or not the geometry has been checked for self-intersections, and whether such self-intersections were found to exist:

If the analytic_form field is not null, it will point to a HELIX_CU_FORM node, which indicates that the curve has a helical shape, as follows:

```
struct HELIX_CU_FORM_s
     {
     vector                    axis_pt                      // $v
     vector                    axis_dir                     // $v
```

```
    vector                      point                          // $v
    char                        hand                           // $c
    interval                    turns                          // $i
    double                      pitch                          // $f
    double                      tol                            // $f
    };
typedef struct HELIX_CU_FORM_s *HELIX_CU_FORM;
```

The axis_pt and axis_dir fields define the axis of the helix. The hand field is '+' for a right-handed and '-' for a left-handed helix. A representative point on the helix is at turn position zero. The turns field gives the extent of the helix relative to the point. For instance, an interval [0 10] indicates a start position at the point and an end 10 turns along the axis. Pitch is the distance travelled along the axis in one turn. Tol is the accuracy to which the owning bcurve fits this specification.

INTERSECTION

An intersection curve is one of the branches of a surface / surface intersection. XT represents these curves exactly; the information held in an intersection curve node is sufficient to identify the particular intersection branch involved, to identify the behaviour of the curve at its ends, and to evaluate precisely at any point in the curve. Specifically, the data is:

— The two surfaces involved in the intersection.

— The two ends of the intersection curve. These are referred to as the 'limits' of the curve. They identify the particular branch involved.

— An ordered array of points along the curve. This array is referred to as the 'chart' of the curve. It defines the parameterisation of the curve, which increases as the array index increases.

— The natural tangent to the curve at any point (i.e. in the increasing parameter direction) is given by the vector cross-product of the surface normals at that point, taking into account the senses of the surfaces.

Singular points where the cross-product of the surface normals is zero, or where one of the surfaces is degenerate, are called terminators. Intersection curves do not contain terminators in their interior. At terminators, the tangent to the curve is defined by the limit of the curve tangent as the curve parameter approaches the terminating value.

Field Name	Data Type	Description
surface	pointer array [2]	surfaces of intersection curve
chart	Pointer	array of hvecs on the curve – see below
start	Pointer	start limit of the curve
end	Pointer	end limit of the curve
intersection_data	Pointer	optional structure for storing additional information associated with an intersection curve

```
struct INTERSECTION_s == ANY_CURVE_s // Intersection
    {
    int                           node_id;              // $d
    union   ATTRIB_GROUP_u       attributes_groups;     // $p
    union  CURVE_OWNER_u            owner;              // $p
    union  CURVE_u                  next;               // $p
    union  CURVE_u                  previous;           // $p
    struct  GEOMETRIC_OWNER_s      *geometric_owner;    // $p
    char                            sense;              // $c
    union  SURFACE_u                surface[ 2 ];       // $p[2]
    struct CHART_s                 *chart;              // $p
    struct LIMIT_s                 *start;              // $p
    struct LIMIT_s                 *end;                // $p
nolog struct INTERSECTION_DATA *intersection_data   // $p
    };
typedef struct INTERSECTION_s *INTERSECTION;
```

A point on an intersection curve is stored in a data structure called an hvec (hepta-vec, or 7- vector):

```
typedef struct hvec_s    // hepta_vec
{
vector           Pvec;              //position
double double    u[2];              //surface parameters
                 v[2];
vector           Tangent;           //curve tangent
double           t;                 //curve parameter
} hvec;
```

Where

- pvec is a point common to both surfaces;
- u[] and v[] are the u and v parameters of the pvec on each of the surfaces;
- tangent is the tangent to the curve at pvec. This will be equal to the (normalized) vector cross product of the surface normals at pvec, when this cross product is non-zero. These surface normals take account of the surface sense fields;
- t is the parameter of the pvec on the curve.

Note: Only the pvec part of an hvec is actually transmitted.

The chart data structure essentially describes a piecewise-linear (chordal) approximation to the true curve. As well as containing the ordered array of hvecs defining this approximation, it contains extra information pertaining to the accuracy of the approximation:

```
struct CHART_s          //Chart
    double              Base_parameter;              // $f
    double              Base scale;                  // $f
    int                 Chart count;                 // $d
    double              Chordal error;               // $f
    double              Angular error;               // $f
    double              Parameter error[2];          // $f[2]
    hvec                Hvec[ 1 ];                   // $h[]
    };
```

Where

- base_parameter is the parameter of the first hvec in the chart;

- base_scale determines the scale of the parameterization (see below);
- chart_count is the length of the hvec array;
- chordal_error is an estimate of the maximum deviation of the curve from the piecewise- linear approximation given by the hvec array. It may be null;
- angular_error is the maximum angle between the tangents of two sequential hvecs. It may be null;
- parameter_error[] is always [null, null];
- hvec[] is the ordered array of hvecs.

The limits of the intersection curve are stored in the following data structure:

```
struct LIMIT_s                          // Limit
     {
     Char type;                        //$c
     Hvec hvec[1];                     //$h[]
     };
typdef struct LIMIT_s *LIMIT;
```

The ‘type’ field may take one of the following values

```
const char SCH_help          ='H'; // help hvec
const char SCH_terminator    ='T'; // terminator
const char SCH_limit         ='L'; // arbitrary limit
const char SCH_boundary      ='B'; // spine boundary
```

The length of the hvec array depends on the type of the limit

- a SCH_help limit is an arbitrary point on a closed intersection curve. There will be one
- hvec in the hvec array, locating the curve.
- a SCH_terminator limit is a point where one of the surface normals is degenerate, or where their cross-product is zero. Typically, there will be more than one branch of intersection between the two surfaces at these singularities. There will be two values in the hvec array. The first will be the exact position of the singularity, and the second will be a point on the curve a small distance away from the terminator. This ‘branch point’ identifies which branch relates to the curve in question. The branch point is the one which appears in the chart, at the corresponding end – so the singularity lies just outside the parameter range of the chart.
- a SCH_limit limit is an artificial boundary of an intersection curve on an otherwise potentially infinite branch. The single hvec describes the end of the curve.
- a SCH_boundary limit is used to describe the end of a degenerate rolling-ball blend. It is not relevant to intersection curves.

The parameterisation of the curve is given as follows. If the chart points are P_i, $i = 0$ to n, with parameters t_i, and natural tangent vectors T_i, then define

$$C_i = | P_{i+1} - P_i |$$

$\cos(a_i) = T_i.(P_{i+1} - P_i)/Ci$ $\cos(b_i) = T_i.(P_i - P_{i-1})/C_{i-1}$

Then at any chart point P_i the angles a_i and b_i are the deviations between the tangent at the chart point and the next and previous chords respectively.

Let f_0 = base_scale

$f_i = (\cos(b_i) / \cos(a_i))\ f_{i-1}$

Then t_0 = base_parameter $t_i = t_{i-1} + C_{i-1} f_{i-1}$

The factors f_i are chosen so that the parameterisation is C1. The parameter of a point between two chart points is given by projecting the point onto the chord between the previous and next chart point. The point on the intersection curve corresponding to a given parameter is defined as follows:

— For a parameter equal to that of a chart point, it is the position of the chart point.

— For a parameter interior to the chart, it is the local point of intersection of three surfaces: the two surfaces of the intersection, and a plane defined by the chart. If the parameter t lies between chart parameters ti, ti+1, then the chord point corresponding to t lies at

$(t_{i+1} - t) / (t_{i+1} - t_i)\ P_i + (t - t_i) / (t_{i+1} - t_i)\ P_{i+1}$

The plane lies through this point and is orthogonal to the chord (P_{i+1}, P_i).

For a parameter between a branch chart point and a terminator, it is the local point of intersection of three surfaces: one of the intersection surfaces and two planes. Surface[0] is used unless it is singular at the terminator and surface[1] is not singular at the terminator. The first plane contains the chord between the branch and the terminator, and the normal of the chosen intersection surface at the terminator or the curve tangent at the branch chart point if the surface normal cannot be defined. The second plane is the plane orthogonal to the chord between the branch and terminator points through the chord point as calculated above.

The intersection_data node is an optional structure for storing surface uv parameters from hvecs that are associated with an intersection curve.

```
logged struct INTERSECTION_DATA_s          //Intersection data
    {
    SCH_intersection_uv_type_t uv_type;    //$u
    double values [1]; ---$f[]
    };
typedef struct INTERSECTION_DATA_s*INTERSECTION_DATA;
inline double *SCH_INTERSECTION_DATA_values(INTERSECTION_DATA self)
{
   return self -> values;
}
SCH_define_init_fn_m(INTERSECTION_DATA_s, self,
                     self -> uv_type = SCH_intersection_uv_none;
                     double*values = SCH_INTERSECTION_DATA_values(self);
                        for (int i = 0; i< n_variable; ++i)
                        values [i] = null;
                    )
```

The intersection_data node contains an enum value and a variable length double array. The enum value specifies the uv values stored in the values array and is set based on the following:

```
typedef short short enum
       {
       SCH_intersection_uv_none =1,
       SCH_intersection_uv_first =2,
       SCH_intersection_uv_second=3,
       SCH_intersection_uv_both=4,
       }
```

```
SCH_intersection_uv_type_t;
char *SCH_intersection_uv_type_sprintf
```

The uv values are converted to the number of parameters which are stored for each chart `hvec` as follows:

- If SCH_intersection_uv_none, the number of parameters is 0
- If SCH_intersection_uv_first or SCH_intersection_uv_second, the number of parameters is 2
- If SCH_intersection_uv_both, the number of parameters is 4

The variable length double array contains these parameters, and the start and end limits. The values for the start and end limits can be found in the variable length arrays in the LIMIT start, and LIMIT end fields of the INTERSECTION node.

The number of values in the double array is calculated as:

(The number of chart points + The number of terminator limits) * (The number of parameters per `hvec`)

For each terminator present in the array the number of values will increase by 0, 2, or 4 depending on the `intersection_uv_type` field. For example, if both the start and the end limits are terminators and the `intersection_uv_type` is set to `SCH_intersection_uv_both` the value will increase by 8.

The order of values in the array is as follows:

If the start limit is a terminator:

If the intersection_uv_type is...	**The order of values in array is...**
`SCH_intersection_uv_first` or `SCH_intersection_uv_both`	intersection node ->start->hvec[0].u[0] intersection node ->start->hvec[0].v[0]
`SCH_intersection_uv_second` or `SCH_intersection_uv_both`	intersection node ->start ->hvec[0].u[1] intersection node ->start ->hvec[0].v[1]

For each `hvec` in the chart:

If the intersection_uv_type is...	**The order of values in array is...**
`SCH_intersection_uv_first` or `SCH_intersection_uv_both`	intersection node->chart ->hvec[i].u[0] intersection node->chart ->hvec[i].v[0]
`SCH_intersection_uv_second` or `SCH_intersection_uv_both`	intersection node->chart ->hvec[i].u[1] intersection node->chart ->hvec[i].v[1]

chart hvecs are wrapped in a loop where i = 0 to the (number of chart hvecs -1).If end limit is a terminator:

If the intersection_uv_type is...	**The order of values in array is...**
`SCH_intersection_uv_first` or `SCH_intersection_uv_both`	intersection node->end ->hvec[0].u[0] intersection node->end ->hvec[0].v[0]

`SCH_intersection_uv_second` or `SCH_intersection_uv_both`	intersection node->end ->hvec[0].u[1] intersection node->end ->hvec[0].v[1]

TRIMMED_CURVE

A trimmed curve is a bounded region of another curve, referred to as its basis curve. It is defined by the basis curve and two points and their corresponding parameters. Trimmed curves are most commonly attached to fins (fins) of tolerant edges in order to specify which portion of the underlying basis curve corresponds to the tolerant edge. They are necessary since the tolerant vertices of the edge do not necessarily lie exactly on the basis curve; the ‘point’ fields of the trimmed curve lie exactly on the basis curve, and within tolerance of the relevant vertex.

The rules governing the parameter fields and points are:

- point_1 and point_2 correspond to parm_1 and parm_2 respectively.
- If the basis curve has positive sense, parm_2 > parm_1.
- If the basis curve has negative sense, parm_2 < parm_1.

In addition,

For open basis curves.

- Both parm_1 and parm_2 shall be in the parameter range of the basis curve.
- point_1 and point_2 shall not be equal.

For periodic basis curves.

- parm_1 shall lie in the base range of the basis curve.
- If the whole basis curve is required then parm_1 and parm_2 should be a period apart and point_1 = point_2. Equality of parm_1 and parm_2 is not permitted.
- parm_1 and parm_2 shall not be more than a period apart.

For closed but non-periodic basis curves.

- Both parm_1 and parm_2 shall be in the parameter range of the basis curve.
- If the whole of the basis curve is required, parm_1 and parm_2 shall lie close enough to each end of the valid parameter range in order that point_1 and point_2 are coincident to XT tolerance (1.0e-8 by default).

The sense of a trimmed curve is positive.

Table 95 — Trimmed curve fields

Field name	Data type	Description
basis_curve	pointer	Basis curve
point_1	vector	start of trimmed portion
point_2	vector	end of trimmed portion
parm_1	double	parameter on basis curve corresponding to point_1

parm_2	double	parameter on basis curve corresponding to point_2

```
struct TRIMMED_CURVE_s == ANY_CURVE_s             // Trimmed Curve
    {
    int                            node_id;                      // $d
    union  ATTRIB_GROUP_u          attributes_groups;            // $p
    union  CURVE_OWNER_u           owner;                        // $p
    union  CURVE_u                 next;                         // $p
    union  CURVE_u                 previous;                     // $p
    struct GEOMETRIC_OWNER_s      *geometric_owner;              // $p
    char                           sense;                        // $c
    union  CURVE_u                 basis_curve;                  // $p
    vector                         point_1;                      // $v
    vector                         point_2;                      // $v
    double                         parm_1;                       // $f
    double                         parm_2;                       // $f
    };
typedef struct TRIMMED_CURVE_s      *TRIMMED_CURVE;
```

SP_CURVE

An SP curve is the 3D curve resulting from embedding a 2D curve in the parameter space of a surface.

The 2D curve shall be a 2D BCURVE; that is it shall either be a rational B curve with a vertex dimensionality of 3, or a non-rational B curve with a vertex dimensionality of 2.

Table 96 — SP curve fields

Field name	Data type	Description
surface	pointer	surface
b_curve	pointer	2D Bcurve
original	pointer0	not used
tolerance_to_original	double	not used

```
struct   SP_CURVE_s == ANY_CURVE_s                  // SP curve
    {
    int                           node_id;                       // $d
    union  ATTRIB_GROUP_u         attributes_groups;             // $p
    union  CURVE_OWNER_u          owner;                         // $p
    union  CURVE_u                next;                          // $p
    union  CURVE_u                previous;                      // $p
    struct GEOMETRIC_OWNER_s     *geometric_owner;               // $p
    char                          sense;                         // $c
    union  SURFACE_u              surface;                       // $p
    struct B_CURVE_s             *b_curve;                       // $p
    union  CURVE_u                original;                      // $p
    double                        tolerance_to_original;         // $f
    };
typedef struct SP_CURVE_s   *SP_CURVE;
```

Surfaces

All surface nodes share the following common fields:

Table 97 — Surface node fields

Field name	Data type	Description
node_id	int	Integer value unique to surface in part

attributes_groups	pointer0	Attributes and groups associated with surface
owner	pointer	topological owner
next	pointer0	next surface in geometry chain
previous	pointer0	previous surface in geometry chain
geometric_owner	pointer0	geometric owner node
sense	char	sense of surface: ‘+’ or ‘-’(see end of Geometry section)

```
struct ANY_SURF_s                                    // Any Surface
    {
    int                          node_id;                    // $d
    union  ATTRIB_GROUP_u        attributes_groups;          // $p
    union  SURFACE_OWNER_u       owner;                      // $p
    union  SURFACE_u             next;                       // $p
    union  SURFACE_u             previous;                   // $p
    struct GEOMETRIC_OWNER_s    *geometric_owner;            // $p
    char                         sense;                      // $c
    };
typedef struct ANY_SURF_s  *ANY_SURF;
```

PLANE

A plane has a parametric representation of the form

R(u, v) = P + uX + vY

where

— P is a point on the plan.

— X and Y are axes in the plane.

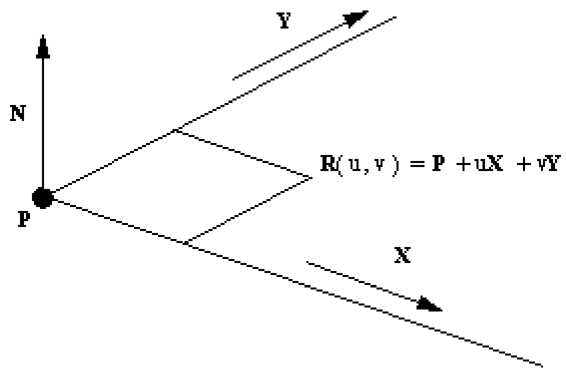


Table 98 — Plane fields

Field name	Data type	Description
pvec	vector	point on the plane
normal	vector	normal to the plane (a unit vector)
x_axis	vector	X axis of the plane (a unit vector)

The Y axis in the definition above is the vector cross product of the normal and x_axis.

```
struct PLANE_s == ANY_SURF_s                         // Plane
```

```
    {
    int                           node_id;                    // $d
    union  ATTRIB_GROUP_u         attributes_groups;          // $p
    union  SURFACE_OWNER_u        owner;                      // $p
    union  SURFACE_u              next;                       // $p
    union  SURFACE_u              previous;                   // $p
    struct GEOMETRIC_OWNER_s     *geometric_owner;            // $p
    char                          sense;                      // $c
    vector                        pvec;                       // $v
    vector                        normal;                     // $v
    vector                        x_axis;                     // $v
    };
typedef struct PLANE_s     *PLANE;
```

CYLINDER

A cylinder has a parametric representation of the form:

R(u,v) = P + rXcos(u) + rYsin(u) + vA

where

— P is a point on the cylinder axis.

— r is the cylinder radius.

— A is the cylinder axis.

— X and Y are unit vectors such that A, X and Y form an orthonormal set.

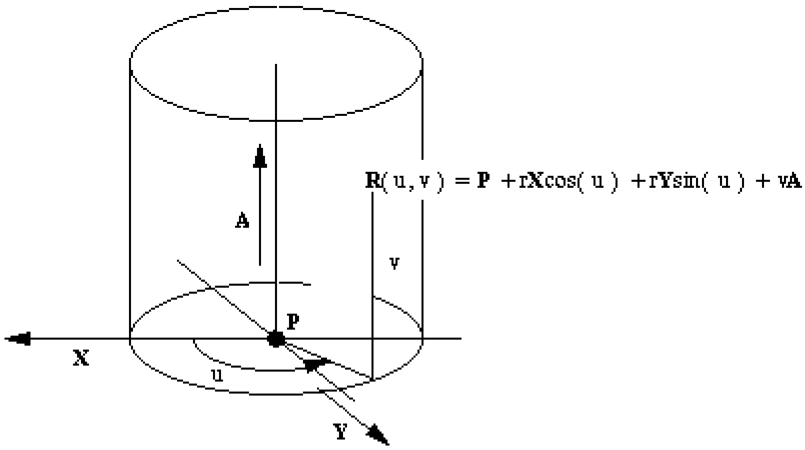


Table 99 — Cylinder fields

Field name	Data type	Description
pvec	vector	point on the cylinder axis
axis	vector	direction of the cylinder axis (a unit vector)
radius	double	radius of cylinder
x_axis	vector	X axis of the cylinder (a unit vector)

The Y axis in the definition above is the vector cross product of the axis and x_axis.

```
struct CYLINDER_s == ANY_SURF_s                     // Cylinder
    {
    int                           node_id;                    // $d
```

```
    union  ATTRIB_GROUP_u          attributes_groups;             // $p
    union  SURFACE_OWNER_u         owner;                         // $p
    union  SURFACE_u               next;                          // $p
    union  SURFACE_u               previous;                      // $p
    struct GEOMETRIC_OWNER_s      *geometric_owner;               // $p
    char                           sense;                         // $c
    vector                         pvec;                          // $v
    vector                         axis;                          // $v
    double                         radius;                        // $f
    vector                         x_axis;                        // $v
    };
typedef struct CYLINDER_s  *CYLINDER;
```

CONE

A cone in XT is only half of a mathematical cone. By convention, the cone axis points away from the half of the cone in use. A cone has a parametric representation of the form:

R(u, v) = P - vA + (Xcos(u) + Ysin(u))(r + vtan(a))

where

- P is a point on the cone axis.
- r is the cone radius at the point P.
- A is the cone axis.
- X and Y are unit vectors such that A, X and Y form an orthonormal set, i.e. Y = A x X.
- a is the cone half angle.

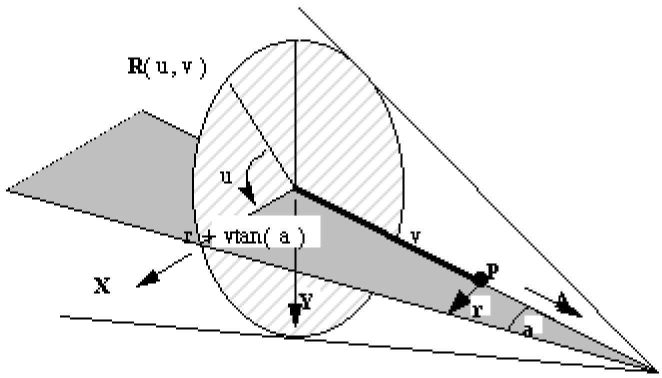


Table 100 — Cone fields

Field name	Data type	Description
pvec	vector	point on the cone axis
axis	vector	direction of the cone axis (a unit vector)
radius	double	radius of the cone at its pvec
sin_half_angle	double	sine of the cone’s half angle
cos_half_angle	double	cosine of the cone’s half angle
x_axis	vector	X axis of the cone (a unit vector)

The Y axis in the definition above is the vector cross product of the axis and x_axis.

```
struct CONE_s == ANY_SURF_s                        // Cone
     {
     int                            node_id;                        // $d
     union  ATTRIB_GROUP_u          attributes_groups;              // $p
     union  SURFACE_OWNER_u         owner;                          // $p
     union  SURFACE_u               next;                           // $p
     union  SURFACE_u               previous;                       // $p
     struct GEOMETRIC_OWNER_s      *geometric_owner;                // $p
     char                           sense;                          // $c
     vector                         pvec;                           // $v
     vector                         axis;                           // $v
     double                         radius;                         // $f
     double                         sin_half_angle;                 // $f
     double                         cos_half_angle;                 // $f
     vector                         x_axis;                         // $v
     };
typedef struct CONE_s      *CONE;
```

SPHERE

A sphere has a parametric representation of the form:

R(u, v) = C + (Xcos(u) + Ysin(u)) rcos(v) + rAsin(v)

where

- C is centre of the sphere.
- r is the sphere radius.
- A, X and Y form an orthonormal axis set.

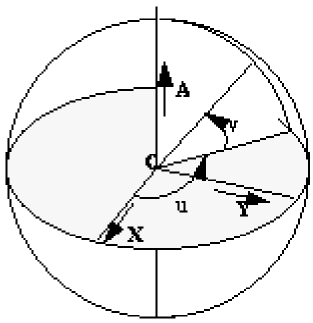


Table 101 — Sphere fields

Field name	Data type	Description
Centre	vector	centre of the sphere
Radius	double	radius of the sphere
Axis	vector	A axis of the sphere (a unit vector)
x_axis	vector	X axis of the sphere (a unit vector)

The Y axis of the sphere is the vector cross product of its A and X axes.

```
struct SPHERE_s == ANY_SURF_s                      // Sphere
     {
     int                            node_id;                        // $d
     union  ATTRIB_GROUP_u          attributes_groups;              // $p
     union  SURFACE_OWNER_u         owner;                          // $p
     union  SURFACE_u               next;                           // $p
```

```
    union  SURFACE_u              previous;                    // $p
    struct GEOMETRIC_OWNER_s     *geometric_owner;             // $p
    char                          sense;                       // $c
    vector                        centre;                      // $v
    double                        radius;                      // $f
    vector                        axis;                        // $v
    vector                        x_axis;                      // $v
    };
typedef struct SPHERE_s    *SPHERE;
```

TORUS

A torus has a parametric representation of the form

$$R(u, v) = C + (X\cos(u) + Y\sin(u))(a + b\cos(v)) + b A\sin(v)$$

where

- C is centre of the torus.
- A is the torus axis.
- a is the major radius.
- b is the minor radius.
- X and Y are unit vectors such that A, X and Y form an orthonormal set.

In XT, there are three types of torus:

Doughnut - the torus is not self-intersecting ($a > b$)

Apple - the outer part of a self-intersecting torus ($a <= b$, $a > 0$)

Lemon - the inner part of a self-intersecting torus ($a < 0$, $|a| < b$)

The limiting case $a = b$ is allowed; it is called an ‘osculating apple’, but there is no ‘lemon’ surface corresponding to this case.

The limiting case $a = 0$ cannot be represented as a torus; this is a sphere.

Table 102 — Torus fields

Field name	Data type	Description
centre	vector	centre of the torus
axis	vector	axis of the torus (a unit vector)
major_radius	double	major radius
minor_radius	double	minor radius
x_axis	vector	X axis of the torus (a unit vector)

The Y axis in the definition above is the vector cross product of the axis of the torus and the x_axis.

```
struct TORUS_s == ANY_SURF_s                        // Torus
    {
        int                        node_id;                    // $d
    union  ATTRIB_GROUP_u          attributes_groups;          // $p
    union  SURFACE_OWNER_u         owner;                      // $p
```

```
    union   SURFACE_u               next;                   // $p
    union   SURFACE_u               previous;               // $p
    struct  GEOMETRIC_OWNER_s      *geometric_owner;        // $p
    char                            sense;                  // $c
    vector                          centre;                 // $v
    vector                          axis;                   // $v
    double                          major_radius;           // $f
    double                          minor_radius;           // $f
    vector                          x_axis;                 // $v
    };

typedef struct TORUS_s      *TORUS;
```

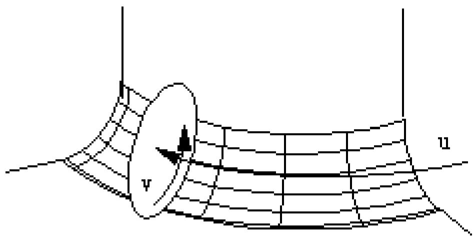
BLENDED_EDGE (Rolling Ball Blend)

XT supports exact rolling ball blends. They have a parametric representation of the form

R(u, v) = C(u) + rX(u)cos(v a(u)) + rY(u)sin(va(u))

where

- C(u) is the spine curve.
- r is the blend radius.
- X(u) and Y(u) are unit vectors such that C’(u) . X(u) = C’(u) . Y(u) = 0.
- a(u) is the angle subtended by points on the boundary curves at the spine.



X, Y and a are expressed as functions of u, as their values change with u.

The spine of the rolling ball blend is the centre line of the blend; i.e. the path along which the centre of the ball moves.

Table 103 — Blended edge fields

Field name	Data type	Description
type	char	type of blend: ‘R’ or ‘E’
surface	pointer[2]	supporting surfaces (adjacent to original edge)
spine	pointer	spine of blend
range	double[2]	offsets to be applied to surfaces
thumb_weight	double[2]	always [1,1]
boundary	pointer0[2]	always [0, 0]
start	pointer0	Start LIMIT in certain degenerate cases
end	pointer0	End LIMIT in certain degenerate cases

```
struct BLENDED_EDGE_s == ANY_SURF_s                 // Blended edge
    {
    int                         node_id;                // $d
    union   ATTRIB_GROUP_u      attributes_groups;      // $p
```

```
    union  SURFACE_OWNER_u        owner;                    // $p
    union  SURFACE_u              next;                     // $p
    union  SURFACE_u              previous;                 // $p
    struct GEOMETRIC_OWNER_s     *geometric_owner;          // $p
    char                          sense;                    // $c
    char                          blend_type;               // $c
    union  SURFACE_u              surface[2];               // $p[2]
    union  CURVE_u                spine;                    // $p
    double                        range[2];                 // $f[2]
    double                        thumb_weight[2];          // $f[2]
    union  SURFACE_u              boundary[2];              // $p[2]
    struct LIMIT_s               *start;                    // $p
    struct LIMIT_s               *end;                      // $p
    };
typedef struct BLENDED_EDGE_s *BLENDED_EDGE;
```

The parameterization of the blend is as follows. The u parameter is inherited from the spine, the constant u lines being circles perpendicular to the spine curve. The v parameter is zero at the blend boundary on the first surface, and one on the blend boundary on the second surface; unless the sense of the spine curve is negative, in which case it is the other way round. The v parameter is proportional to the angle around the circle.

XT data can contain blends of the following types:

```
const char SCH_rolling_ball = 'R';       //  rolling ball blend
const char SCH_cliff_edge   = 'E';       //  cliff edge blend
```

For rolling ball blends, the spine curve will be the intersection of the two surfaces obtained by offsetting the supporting surfaces by an amount given by the respective entry in range[]. Note that the offsets to be applied may be positive or negative, and that the sense of the surface is significant; i.e. the offset vector is the natural unit surface normal, times the range, times –1 if the sense is negative.

For cliff edge blends, one of the surfaces will be a blended_edge with a range of [0,0]; its spine will be the cliff edge curve, and its supporting surfaces will be the surfaces of the faces adjacent to the cliff edge. Its type will be R.

The limit fields will only be non-null if the spine curve is periodic but the edge curve being blended has terminators – for example if the spine is elliptical but the blend degenerates. In this case the two LIMIT nodes, of type ‘L’, determine the extent of the spine.

BLEND_BOUND (Blend boundary surface)

A blend_bound surface is a construction surface, used to define the boundary curve where a blend becomes tangential to its supporting surface. It is an implicit surface defined internally so that it intersects one of the supporting surfaces along the boundary curve. It is orthogonal to the blend and the supporting surface along this boundary curve. The supporting surface corresponding to the blend_bound is

Blend_bound -> blend.blended_edge -> surface[1-blend_bound->boundary]

Blend boundary surfaces have no parameterization, but are defined by the distance function

f(X) = f0 (X + r1 * grad_f1 (X)) – r0

Where

- f0 is the surface distance function of the supporting surface corresponding to the blend_bound.
- r0 is the blend radius corresponding to that supporting surface.
- f1 is the surface distance function of the other supporting surface of the blend.

— r1 is the blend radius corresponding to the other supporting surface.

Blend boundary surfaces are most commonly referenced by the intersection curve representing the boundary curve of the blend.

The data stored in the XT data for a blend_bound is only that necessary to identify the relevant blend and supporting surface:

Table 104 — Blend boundary surface fields

Field name	Data type	Description
boundary	short	index into supporting surface array
blend	pointer	corresponding blend surface

```
struct BLEND_BOUND_s == ANY_SURF_s          // Blend boundary
    {
    int                             node_id;                      // $d
    union  ATTRIB_GROUP_u           attributes_groups;            // $p
    union  SURFACE_OWNER_u          owner;                        // $p
    union  SURFACE_u                next;                         // $p
    union  SURFACE_u                previous;                     // $p
    struct GEOMETRIC_OWNER_s       *geometric_owner;              // $p
    char                            sense;                        // $c
    short                           boundary;                     // $n
    union  SURFACE_u                blend;                        // $p
    };
typedef struct BLEND_BOUND_s  *BLEND_BOUND;
```

The supporting surface corresponding to the blend_bound is

blend_bound->blend.blended_edge->surface[1 - blend_bound->boundary].

OFFSET_SURF

An offset surface is the result of offsetting a surface a certain distance along its normal, taking into account the surface sense. It inherits the parameterization of this underlying surface.

Table 105 — Offset surface fields

Field name	Data type	Description
Check	char	check status
true_offset	logical	not used
surface	pointer	underlying surface
offset	double	signed offset distance
scale	double	for internal use only – may be set to null

```
struct OFFSET_SURF_s == ANY_SURF_s       // Offset surface
    {
    int                              node_id;                     // $d
    union  ATTRIB_GROUP_u            attributes_groups;           // $p
    union  SURFACE_OWNER_u           owner;                       // $p
    union  SURFACE_u                 next;                        // $p
    union  SURFACE_u                 previous;                    // $p
    struct GEOMETRIC_OWNER_s        *geometric_owner;             // $p
    char                             sense;                       // $c
    char                             check;                       // $c
    logical                          true_offset;                 // $l
```

```
    union  SURFACE_u              surface;                    // $p
    double                        offset;                     // $f
    double                        scale;                      // $f
    };
typedef struct OFFSET_SURF_s    *OFFSET_SURF;
```

The offset surface is subject to the following restrictions:

- The offset distance shall not be within modeller linear resolution of zero.
- The sense of the offset surface shall be the same as that of the underlying surface.
- Offset surfaces may not share a common underlying surface.

The ‘check’ field may take one of the following values:

```
    [2] const char SCH_valid       = 'V';    [3] // valid
    [4] const char SCH_invalid     = 'I';    [5] // invalid
    [6] const char SCH_unchecked   = 'U';    [8] // has not been checked
    [7]
```

B_SURFACE

XT supports B spline surfaces in full NURBS format.

B-SURFACE DEFINITION

$$P(u,v) = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} b_i(u) \quad b_j(v) \quad w_{ij}V_{ij}}{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} b_i(u) \quad b_j(v) \quad w_{ij}}$$

The B-surface definition is best thought of as an extension of the B-curve definition into two parameters, usually called u and v. Two knot sets are required and the number of control vertices is the product of the number that would be required for a curve using each knot vector. The rules for periodicity and closure given in the B-curve documentation are extended to surfaces in an obvious way.

For attachment to topology a B-surface is required to have G_1 continuity. That is to say that the surface normal direction shall be continuous.

Surfaces that are self-intersecting or contain cusps are not permitted to be attached to topology.

Table 106 — B-Surface fields

Field name	Data type	Description
nurbs	pointer	Geometric definition
data	pointer0	Auxiliary information

```
struct B_SURFACE_s == ANY_SURF_s                     // B surface
    {
    int                           node_id;                    // $d
    union ATTRIB_GROUP_u          attributes_groups;          // $p
    union  SURFACE_OWNER_u        owner;                      // $p
    union  SURFACE_u              next;                       // $p
    union  SURFACE_u              previous;                   // $p
    struct GEOMETRIC_OWNER_s     *geometric_owner;            // $p
```

```
     char                           sense;                       // $c
     struct NURBS_SURF_s           *nurbs;                       // $p
     struct SURFACE_DATA_s         *data;                        // $p
     };
typedef struct B_SURFACE_s       *B_SURFACE;
```

The data stored in the XT data for a NURBS surface is described in the table below.

Table 107 — NURB Surface fields

Field name	Data type	Description
u_periodic	logical	true if surface is periodic in u parameter
v_periodic	logical	true if surface is periodic in v parameter
u_degree	short	u degree of the surface
v_degree	short	v degree of the surface
n_u_vertices	int	number of control vertices ('poles') in u direction
n_v_vertices	int	number of control vertices ('poles') in v direction
u_knot_type	byte	form of u knot vector – see "B curve"
v_knot_type	byte	form of v knot vector
n_u_knots	int	number of distinct u knots
n_v_knots	int	number of distinct v knots
Rational	logical	true if surface is rational
u_closed	logical	true if surface is closed in u
v_closed	logical	true if surface is closed in v
surface_form	byte	shape of surface, if special
vertex_dim	short	dimension of control vertices
bspline_vertices	pointer	control vertices (poles) node
u_knot_mult	pointer	multiplicities of u knot vector
v_knot_mult	pointer	multiplicities of v knot vector
u_knots	pointer	u knot vector
v_knots	pointer	v knot vector

The surface form enum is defined below.

```
typedef enum
    {
    SCH_unset = 1,                          // Unknown
    SCH_arbitrary = 2,                      // No particular shape
    SCH_planar = 3,
    SCH_cylindrical = 4,
    SCH_conical = 5,
    SCH_spherical = 6,
    SCH_toroidal = 7,
    SCH_surf_of_revolution = 8,
    SCH_ruled = 9,
    SCH_quadric = 10,
    SCH_swept = 11
    }
    SCH_surface_form_t;
```

```
struct NURBS_SURF_s                                 // NURBS surface
   {
   logical                     u_periodic;                // $l
   logical                     v_periodic;                // $l
   short                       u_degree;                  // $n
   short                       v_degree;                  // $n
   int                         n_u_vertices;              // $d
   int                         n_v_vertices;              // $d
   SCH_knot_type_t             u_knot_type;               // $u
   SCH_knot_type_t             v_knot_type;               // $u
   int                         n_u_knots;                 // $d
   int                         n_v_knots;                 // $d
   logical                     rational;                  // $l
   logical                     u_closed;                  // $l
   logical                     v_closed;                  // $l
   SCH_surface_form_t          surface_form;              // $u
   short                       vertex_dim;                // $n
   struct BSPLINE_VERTICES_s  *bspline_vertices;          // $p
   struct KNOT_MULT_s         *u_knot_mult;               // $p
   struct KNOT_MULT_s         *v_knot_mult;               // $p
   struct KNOT_SET_s          *u_knots;                   // $p
   struct KNOT_SET_s          *v_knots;                   // $p
   };
typedef struct NURBS_SURF_s *NURBS_SURF;
```

The ‘bspline_vertices’, ‘knot_set’ and ‘knot_mult’ nodes and the ‘knot_type’ enum are described in the documentation for B_CURVE.

The ‘surface data’ field in a B surface node is a structure designed to hold auxiliary or ‘derived’ data about the surface: it is not a necessary part of the definition of the B surface. It may be null, or the majority of its individual fields may be null.

```
struct  SURFACE_DATA_s           // auxiliary surface data
    {
    interval                  original_uint;                // $i
    interval                  original_vint;                // $i
    interval                  extended_uint;                // $i
    interval                  extended_vint;                // $i
    SCH_self_int_t            self_int;                     // $u
    char                      original_u_start;             // $c
    char                      original_u_end;               // $c
    char                      original_v_start;             // $c
    char                      original_v_end;               // $c
    char                      extended_u_start;             // $c
    char                      extended_u_end;               // $c
    char                      extended_v_start;             // $c
    char                      extended_v_end;               // $c
    char                      analytic_form_type;           // $c
    char                      swept_form_type;              // $c
    char                      spun_form_type;               // $c
    char                      blend_form_type;              // $c
    void                     *analytic_form;                // $p
    void                     *swept_form;                   // $p
    void                     *spun_form;                    // $p
    void                     *blend_form;                   // $p
    };
typedef struct SURFACE_DATA_s *SURFACE_DATA;
```

The ‘original_’ and ‘extended_’ parameter intervals and corresponding character fields original_u_start etc. are all connected with the ability to extend B surfaces when necessary – functionality which is commonly exploited in “local operation” algorithms for example. This is done automatically without the need for user intervention.

In cases where the required extension can be performed by adding rows or columns of control points, then the nurbs data will be modified accordingly – this is referred to as an 'explicit' extension. In some rational B surface cases, explicit extension is not possible - in these cases, the surface will be 'implicitly' extended. When a B surface is implicitly extended, the nurbs data is not changed, but it will be treated as being larger by allowing out-of-range evaluations on the surface. Whenever an explicit or implicit extension takes place, it is reflected in the following fields:

— "original_u_int" and "original_v_int" are the original valid parameter ranges for a B surface before it was extended.

— "extended_u_int" and "extended_v_int" are the valid parameter ranges for a B surface once it has been extended.

The character fields 'original_u_start' etc. all refer to the status of the corresponding parameter boundary of the surface before or after an extension has taken place. For B surfaces, the character can have one of the following values:

```
const char SCH_degenerate = 'D';      [9]// Degenerate edge
const char SCH_periodic   = 'P';      [10] // Periodic parameterization
const char SCH_bounded    = 'B';      [11] // Parameterization bounded
const char SCH_closed     = 'C';      [12] // Closed, but not periodic
```

The separate fields original_u_start and extended_u_start etc. are necessary because an extension may cause the corresponding parameter boundary to become degenerate.

If the surface_data node is present, then the original_u_int, original_v_int, original_u_start, original_u_end, original_v_start and original_v_end fields should be set to their appropriate values. If the surface has not been extended, the extended_u_int and extended_v_int fields should contain null, and the extended_u_start etc. fields should contain

```
const char SCH_unset_char = '?'; // generic uninvestigated value
```

As soon as any parameter boundary of the surface is extended, all the fields should be set, regardless of whether the corresponding boundary has been affected by the extension.

The SCH_self_int_t enum is documented in the corresponding curve_data structure under B curve.

The 'swept_form_type', 'spun_form_type' and 'blend_form_type' characters and the corresponding pointers swept_form, spun_form and blend_form, are not implemented in XT. The character fields should be set to SCH_unset_char ('?') and the pointers should be set to null pointer.

If the analytic_form field is not null, it will point to a HELIX_SU_FORM node, which indicates that the surface has a helical shape. In this case the analytic_form_type field will be set to 'H'.

```
struct HELIX_SU_FORM_s
    {
    vector                    axis_pt                          // $v
    vector                    axis_dir                         // $v
    char                      hand                             // $c
    interval                  turns                            // $i
    double                    pitch                            // $f
    double                    gap                              // $f
    double                    tol                              // $f
    };
typedef struct HELIX_SU_FORM_s *HELIX_SU_FORM;
```

The axis_pt and axis_dir fields define the axis of the helix. The hand field is '+' for a right-handed and '-' for a left-handed helix. The turns field gives the extent of the helix relative to the profile curve which was used to generate the surface. For instance, an interval [0 10] indicates a start position at the profile curve and an end 10 turns along the axis. Pitch is the distance travelled along the axis in one turn. Tol is the accuracy to which the owning bsurface fits this specification. Gap is for future expansion and will currently be zero. The v parameter increases in the direction of the axis.

SWEPT_SURF

A swept surface has a parametric representation of the form:

R(u, v) = C(u) + vD

where

- C(u) is the section curve.
- D is the sweep direction (unit vector).
- C shall not be an intersection curve or a trimmed curve.

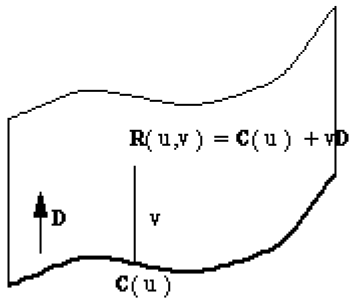


Table 108 — Swept surface fields

Field name	Data type	Description
section	pointer	section curve
sweep	vector	sweep direction (a unit vector)
scale	double	for internal use only – may be set to null

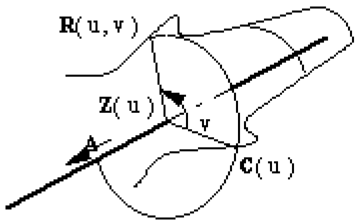
```
struct SWEPT_SURF_s == ANY_SURF_s            // Swept surface
     {
     int                            node_id;                     // $d
     union  ATTRIB_GROUP_u          attributes_groups;           // $p
     union  SURFACE_OWNER_u         owner;                       // $p
     union  SURFACE_u               next;                        // $p
     union  SURFACE_u               previous;                    // $p
     struct GEOMETRIC_OWNER_s      *geometric_owner;             // $p
     char                           sense;                       // $c
     union  CURVE_u                 section;                     // $p
     vector                         sweep;                       // $v
     double                         scale;                       // $f
     };
typedef struct SWEPT_SURF_s *SWEPT_SURF;
```

SPUN_SURF

A spun surface has a parametric representation of the form:

R(u, v) = Z(u) + (C(u) - Z(u))cos(v) + A X (C(u) - Z(u)) sin(v) where

- C(u) is the profile curve
- Z(u) is the projection of C(u) onto the spin axis.
- A is the spin axis direction (unit vector).
- C shall not be an intersection curve or a trimmed curve.



NOTE: Z(u) = P + ((C(u) - P) . A)A where P is a reference point on the axis.

Table 109 — Spun surface fields

Field name	Data type	Description
profile	pointer	profile curve
base	vector	point on spin axis
axis	vector	spin axis direction (a unit vector)
start	vector	position of degeneracy at low u (may be null)
end	vector	position of degeneracy at low v (may be null)
start_param	double	curve parameter at low u degeneracy (may be null)
end_param	double	curve parameter at high u degeneracy (may be null)
x_axis	vector	unit vector in profile plane if common with spin axis
scale	double	for internal use only – may be set to null

```
struct SPUN_SURF_s == ANY_SURF_s                    // Spun surface
    {
    int                          node_id;                      // $d
    union  ATTRIB_GROUP_u        attributes_groups;            // $p
    union  SURFACE_OWNER_u       owner;                        // $p
    union  SURFACE_u             next;                         // $p
    union  SURFACE_u             previous;                     // $p
    struct GEOMETRIC_OWNER_s    *geometric_owner;              // $p
    char                         sense;                        // $c
    union  CURVE_u               profile;                      // $p
    vector                       base;                         // $v
    vector                       axis;                         // $v
    vector                       start;                        // $v
    vector                       end;                          // $v
    double                       start_param;                  // $f
    double                       end_param;                    // $f
    vector                       x_axis;                       // $v
    double                       scale;                        // $f
    };
typedef struct SPUN_SURF_s *SPUN_SURF;
```

The ‘start’ and ‘end’ vectors correspond to physical degeneracies on the spun surface caused by the profile curve crossing the spin axis at that point. The values start_param and end_param are the corresponding parameters on the curve. These parameter values define the valid range for the u parameter of the surface. If either value is null, then the valid range for u is infinite in that direction. For example, for a straight line profile curve intersecting the spin axis at the parameter t=1, values of null for start_param and 1 for end_param would define a cone with u parameterization (-infinity, 1].

If the profile curve lies in a plane containing the spin axis, then x_axis shall be set to a vector perpendicular to the spin axis and in the plane of the profile, pointing from the spin axis to a point on the profile curve in the valid range. If the profile curve is not planar, or its plane does not contain the spin axis, then x_axis should be set to null.

Point

Table 110 — Point fields

Field name	Data type	Description
node_id	int	integer unique within part
attributes_groups	pointer0	attributes and groups associated with point
owner	pointer	Owner
next	pointer0	next point in chain
previous	pointer0	previous point in chain
pvec	vector	position of point

```
union POINT_OWNER_u
     {
     struct VERTEX_s              *vertex;
     struct BODY_s                *body;
     struct ASSEMBLY_s            *assembly;
     struct WORLD_s               *world;
     };

struct POINT_s                                      // Point
     {
     int                        node_id;                        // $d
     union  ATTRIB_GROUP_u      attributes_groups;              // $p
     union  POINT_OWNER_u       owner;                          // $p
     struct POINT_s            *next;                           // $p
     struct POINT_s            *previous;                       // $p
     vector                     pvec;                           // $v
     };
typedef struct POINT_s    *POINT;
```

Transform

Table 111 — Transform fields

Field name	Data type	Description
node_id	int	integer unique within part
owner	pointer	owning instance or world
next	pointer0	next transform in chain
previous	pointer0	previous pointer in chain
rotation_matrix	double[3][3]	rotation component
translation_vector	vector	translation component
scale	double	scaling factor
flag	byte	binary flags indicating non-trivial components
perspective_vector	vector	perspective vector (always null vector)

The transform acts as

$$x' = (rotation\_matrix . x + translation\_vector) * scale$$

The 'flag' field contains various bit flags which identify the components of the transformation:

Table 112 — Transform action fields

Flag Name	Binary Value	Description
translation	00001	set if translation vector non-zero
rotation	00010	set if rotation matrix is not the identity
scaling	00100	set if scaling component is not 1.0
reflection	01000	set if determinant of rotation matrix is negative
general affine	10000	set if the rotation_matrix is not a rigid rotation

```
union TRANSFORM_OWNER_u
     {
     struct INSTANCE_s          *instance;
     struct WORLD_s             *world;
     };

struct TRANSFORM_s                     // Transformation
     {
     int                          node_id;                   // $d
     union  TRANSFORM_OWNER_u     owner;                     // $p
     struct TRANSFORM_s          *next;                      // $p
     struct TRANSFORM_s          *previous;                  // $p
     double                       rotation_matrix[3][3];     // $f[9]
     vector                       translation_vector;        // $v
     double                       scale;                     // $f
     unsigned                     flag;                      // $d
     vector                       perspective_vector;        // $v
     };
typedef struct TRANSFORM_s *TRANSFORM;
```

Curve and Surface Senses

The 'natural' tangent to a curve is that in the increasing parameter direction, and the 'natural' normal to a surface is in the direction of the cross-product of dP/du and dP/dv. For some purposes these are modified by the curve and surfaces senses, respectively – for example in the definition of blend surfaces, offset surfaces and intersection curves.

At the PK interface, the edge/curve and face/surface sense orientations are regarded as properties of the topology/geometry combination. In the XT format, this orientation information resides in the curves, surfaces and faces as follows:

The edge/curve orientation is stored in the curve->sense field. The face/surface orientation is a combination of sense flags stored in the face->sense and surface->sense fields, so the face/surface orientation is true (i.e. the face normal is parallel to the natural surface normal) if neither, or both, of the face and surface senses are positive.

Geometric_owner

Where geometry has dependants, the dependants point back to the referencing geometry by means of Geometric Owner nodes. Each geometric node points to a doubly-linked ring of Geometric Owner nodes which identify its referencing geometry. Referenced geometry is as follows:

Intersection:	2 surfaces
SP-curve:	Surface
Trimmed curve:	basis curve
Blended edge:	2 supporting surfaces, 2 blend_bound surfaces, 1 spine curve
Blend bound:	blend surface
Offset surface:	underlying surface
Swept surface:	section curve
Spun surface:	profile curve

Note that the 2D B-curve referenced by an SP-curve is not a dependent in this sense, and does not need a geometric owner node.

Table 113 — Geometry owner fields

Field name	Data type	Description
owner	pointer	referencing geometry
next	pointer	next in ring of geometric owners referring to the same geometry
previous	pointer	previous in above ring
shared_geometry	pointer	referenced (dependent) geometry

```
struct GEOMETRIC_OWNER_s          //  geometric owner of geometry
    {
    union  GEOMETRY_u              owner;                         //  $p
    struct GEOMETRIC_OWNER_s      *next;                          //  $p
    struct GEOMETRIC_OWNER_s      *previous;                      // $p
    union  GEOMETRY_u              shared_geometry;               //  $p
    };
typedef struct GEOMETRIC_OWNER_s *GEOMETRIC_OWNER;
```

F.2.4.3.4 Topology

In the following tables, ‘ignore’ means this may be set to null (zero) and should be ignored.

Unless otherwise stated, all chains of nodes are doubly-linked and null-terminated.

WORLD

Table 114 — World topology fields

Field name	Type	Description
assembly	pointer0	Head of chain of assemblies
attribute	pointer0	Ignore
body	pointer0	Head of chain of bodies
transform	pointer0	Head of chain of transforms
surface	pointer0	Head of chain of surfaces
curve	pointer0	Head of chain of curves
point	pointer0	Head of chain of points
alive	logical	True unless partition is at initial pmark
attrib_def	pointer0	Head of chain of attribute definitions

highest_id	int	Highest pmark id in partition
current_id	int	Id of current pmark
index_map_offset	int	Shall be set to 0
index_map	pointer0	Shall be set to null
schema_embedding_map	pointer0	Shall be set to null

The World node is only used when a partition is transmitted. Because some of the attribute definitions may be referenced by nodes which have been deleted, but which may reappear on rollback, the attribute definitions are chained off the World node rather than simply being referenced by attributes.

The fields index_map_offset, index_map, and schema_embedding_map are used for Indexed Transmit; applications writing XT data shall set them to 0 and null.

```
struct WORLD_s                                      // World
    {
    struct ASSEMBLY_s          *assembly;                  // $p
    struct ATTRIBUTE_s         *attribute;                 // $p
    struct BODY_s              *body;                      // $p
    struct TRANSFORM_s         *transform;                 // $p
    union  SURFACE_u            surface;                   // $p
    union  CURVE_u              curve;                     // $p
    struct POINT_s             *point;                     // $p
    logical                     alive;                     // $l
    struct ATTRIB_DEF_s        *attrib_def;                // $p
    int                         highest_id;                // $d
    int                         current_id;                // $d
    };
typedef struct WORLD_s  *WORLD;
```

ASSEMBLY

Table 115 — Assembly fields

highest_node_id	int	Highest node-id in assembly
attributes_groups	pointer0	Head of chain of attributes of, and groups in, assembly
attribute_chains	pointer0	List of attributes, one for each attribute definition used in the assembly
list	pointer0	Null
surface	pointer0	Head of construction surface chain
curve	pointer0	Head of construction curve chain
point	pointer0	Head of construction point chain
key	pointer0	Ignore
res_size	double	Value of ‘size box’ when transmitted (normally 1000)
res_linear	double	Value of modeller linear precision when transmitted (normally 1.0e-8).
ref_instance	pointer0	Head of chain of instances referencing this assembly
next	pointer0	Ignore
previous	pointer0	Ignore
state	byte	Set to 1.
owner	pointer0	Ignore
type	byte	Always 1.
sub_instance	pointer0	Head of chain of instances in assembly

The value of the ‘state’ field should be ignored, as should any nodes of type ‘KEY’ referenced by the assembly. If XT data is constructed without use of the Parasolid Kernel, the state field should be set to 1, and the key to null.

The highest_node_id gives the highest node-id of any node in the assembly. Certain nodes within the assembly (namely instances, transforms, geometry, attributes and groups) have unique node-ids which are non-zero integers.

```
typedef enum
     {
     SCH_collective_assembly  = 1,
     SCH_conjunctive_assembly = 2,
     SCH_disjunctive_assembly = 3
     }
     SCH_assembly_type;

typedef enum
     {
     SCH_new_part       = 1,
     SCH_stored_part    = 2,
     SCH_modified_part  = 3,
     SCH_anonymous_part = 4,
     SCH_unloaded_part  = 5
     }
     SCH_part_state;

struct ASSEMBLY_s                              //  Assembly
     {
     int                         highest_node_id;               //  $d
     union ATTRIB_GROUP_u        attributes_groups;             //  $p
     struct LIST_s              *attribute_chains;              //  $p
     struct LIST_s              *list;                          //  $p
     union SURFACE_u             surface;                       //  $p
     union CURVE_u               curve;                         //  $p
     struct POINT_s             *point;                         //  $p
     struct KEY_s               *key;                           //  $p
     double                      res_size;                      //  $f
     double                      res_linear;                    //  $f
     struct INSTANCE_s          *ref_instance;                  //  $p
     struct ASSEMBLY_s          *next;                          //  $p
     struct ASSEMBLY_s          *previous;                      //  $p
     SCH_part_state              state;                         //  $u
     struct WORLD_s             *owner;                         //  $p
     SCH_assembly_type           type;                          //  $u
     struct INSTANCE_s          *sub_instance;                  //  $p
     };
typedef struct ASSEMBLY_s *ASSEMBLY;
struct KEY_s                            // Key
     {
     string[1];                 char                            // $c[]
     };
typedef struct KEY_s *KEY;
```

INSTANCE

Table 116 — Instance fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of instance and member_of_groups of instance
type	byte	Always 1

part	pointer	Part referenced by instance
transform	pointer0	Transform of instance
assembly	pointer	Assembly in which instance lies
next_in_part	pointer0	Next instance in assembly
prev_in_part	pointer0	Previous instance in assembly
next_of_part	pointer0	Next instance of instance->part
prev_of_part	pointer0	Previous instance of instance->part

```
typedef enum
     {
     SCH_positive_instance = 1,
     SCH_negative_instance = 2
     }
     SCH_instance_type;

union  PART_u
     {
     struct BODY_s              *body;
     struct ASSEMBLY_s          *assembly;
     };
typedef union PART_u       PART;

struct INSTANCE_s                                    //  Instance
     {
     int                        node_id;                        //  $d
     union  ATTRIB_GROUP_u      attributes_groups;              //  $p
     SCH_instance_type          type;                           //  $u
     union  PART_u              part;                           //  $p
     struct TRANSFORM_s        *transform;                      //  $p
     struct ASSEMBLY_s         *assembly;                       //  $p
     struct INSTANCE_s         *next_in_part;                   //  $p
     struct INSTANCE_s         *prev_in_part;                   //  $p
     struct INSTANCE_s         *next_of_part;                   //  $p
     struct INSTANCE_s         *prev_of_part;                   //  $p
     };
typedef struct INSTANCE_s *INSTANCE;
```

BODY

Table 117 — Body fields

Field name	Type	Description
highest_node_id	int	Highest node-id in body
attributes_groups	pointer0	Head of chain of attributes of, and groups in, body
attribute_chains	pointer0	List of attributes, one for each attribute definition used in the body
surface	pointer0	Head of construction surface chain
curve	pointer0	Head of construction curve chain
point	pointer0	Head of construction point chain
key	pointer0	Ignore
res_size	double	Value of ‘size box’ when transmitted (normally 1000)
res_linear	double	Value of modeller linear precision when transmitted (normally 1.0e-8)
ref_instance	pointer0	Head of chain of instances referencing this part
next	pointer0	Ignore

previous	pointer0	Ignore
state	byte	Set to 1 (see below)
owner	pointer0	Ignore
body_type	byte	Body type
nom_geom_state	byte	Set to 1 (not documented)
shell	pointer0	For general bodies: null For solid bodies: the first shell in one of the solid regions For other bodies: the first shell in one of the regions This field is **obsolete**, and should be ignored by applications reading XT data. When writing XT data, it shall be set as above.
boundary_surface	pointer0	Head of chain of surfaces attached directly or indirectly to faces or edges or fins
boundary_curve	pointer0	Head of chain of curves attached directly or indirectly to edges or faces or fins
boundary_point	pointer0	Head of chain of points attached to vertices
region	pointer	Head of chain of regions in body; this is the infinite region
edge	pointer0	Head of chain of all non-wireframe edges in body
vertex	pointer0	Head of chain of all vertices in body
index_map_offset	int	Shall be set to 0
index_map	pointer0	Shall be set to null
node_id_index_map	pointer0	Shall be set to null
schema_embedding_map	pointer0	Shall be set to null

The value of the 'state' field should be ignored, as should any nodes of type 'KEY' referenced by the body. If the XT data is constructed without using the Parasolid Kernel, the state field should be set to 1, and the key to null.

The highest_node_id gives the highest node of any node in this body. Most nodes in a body have node-ids, which are non-zero integers unique to that node within the body. Applications writing XT data shall ensure that node-ids are present and distinct. The details of which nodes have node ids are given in an appendix.

The fields index_map_offset, index_map, node_id_index_map, and schema_embedding_map are used for Indexed Transmit; applications writing XT data shall ensure that these fields are set to 0 and null.

```
typedef enum
     {
     SCH_solid_body      = 1,
     SCH_wire_body       = 2,
     SCH_sheet_body      = 3,
     SCH_general_body    = 6
     }
     SCH_body_type;

typedef short short enum
        {
        SCH_nom_geom_off = 1,              --- Entirely off
        SCH_nom_geom_on  = 2               --- Entirely on
        }
```

```
        SCH_nom_geom_state_t;

struct BODY_s                                      //  Body
    {
    int                         highest_node_id;                //  $d
    union ATTRIB_GROUP_u        attributes_groups;              //  $p
    struct LIST_s              *attribute_chains;               //  $p
    union SURFACE_u             surface;                        //  $p
    union CURVE_u               curve;                          //  $p
    struct POINT_s             *point;                          //  $p
    struct KEY_s               *key;                            //  $p
    double                      res_size;                       //  $f
    double                      res_linear;                     //  $f
    struct INSTANCE_s          *ref_instance;                   //  $p
    struct BODY_s              *next;                           //  $p
    struct BODY_s              *previous;                       //  $p
    SCH_part_state              state;                          //  $u
    struct WORLD_s             *owner;                          //  $p
    SCH_body_type               body_type;                      //  $u
    SCH_nom_geom_state_t       nom_geom_state;                  //  $u
    struct SHELL_s             *shell;                          //  $p
    union SURFACE_u             boundary_surface;               //  $p
    union CURVE_u               boundary_curve;                 //  $p
    struct POINT_s             *boundary_point;                 //  $p
    struct REGION_s            *region;                         //  $p
    struct EDGE_s              *edge;                           //  $p
    struct VERTEX_s            *vertex;                         //  $p
    int                        index_map_offset;                //  $d
    struct INT_VALUES_s        *index_map;                      //  $p
    struct INT_VALUES_s        *node_id_index_map;              //  $p
    struct INT_VALUES_s        *schema_embedding_map;           //  $p
    };
typedef struct BODY_s     *BODY;
```

Attaching Geometry to Topology

The faces which reference a surface are chained together, surface->owner is the head of this chain. Similarly the edges which reference the same curve are chained together. Fins do not share curves.

Geometry in parts may be chained into one of the three boundary geometry chains, or one of the three construction geometry chains. A geometric node will fall into one of the following cases:

Table 118 — Geometry to Topology attachment

Geometry	Owner	Whether chained
Attached to face	face	In boundary_surface chain
Attached to edge or fin	edge or fin	In boundary_curve chain
Attached to vertex	vertex	In boundary_point chain
Indirectly attached to face or edge or fin	body	In boundary_surface chain or boundary_curve chain
Construction geometry	body or assembly	In surface, curve or point chain
2D B-curve in SP-curve	null	Not chained

Here ‘indirectly attached’ means geometry which is a dependent of a dependent of (... etc) of geometry attached to an edge, face or fin.

Geometry in a construction chain may reference geometry in a boundary chain, but not vice-versa.

REGION

Table 119 — Region fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of region and member_of_groups of region
body	pointer	Body of region
next	pointer0	Next region in body
prev	pointer0	Previous region in body
shell	pointer0	Head of singly-linked chain of shells in region
type	char	Region type – solid (‘S’) or void (‘V’)

```
struct REGION_s                                            //  Region
    {
    int                          node_id;                          //  $d
    union  ATTRIB_GROUP_u        attributes_groups;                //  $p
    struct BODY_s               *body;                             //  $p
    struct REGION_s             *next;                             //  $p
    struct REGION_s             *previous;                         //  $p
    struct SHELL_s              *shell;                            //  $p
    char                         type;                             //  $c
    };
typedef struct REGION_s   *REGION;
```

SHELL

Table 120 — Shell fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of shell
body	pointer0	For shells in wire and sheet bodies, and for shells bounding a solid region of a solid body, this is set to the body of the shell. For shells in general bodies, or void shells in solid bodies, it is null. This field is **obsolete**, and should be ignored by applications reading XT data. When writing XT data, it shall be set as above.
next	pointer0	Next shell in region
face	pointer0	Head of chain of back-faces of shell (i.e. faces with face normal pointing out of region of shell).
edge	pointer0	Head of chain of wire-frame edges of shell
vertex	pointer0	If shell consists of a single vertex, this is it; else null
region	pointer	Region of shell
front_face	pointer0	Head of chain of front-faces of shell (i.e. faces with face normal pointing into region of shell)

```
struct SHELL_s                                             //  Shell
    {
    int                          node_id;                          //  $d
    union  ATTRIB_GROUP_u        attributes_groups;                //  $p
    struct BODY_s               *body;                             //  $p
    struct SHELL_s              *next;                             //  $p
```

```
    struct FACE_s               *face;                      //  $p
    struct EDGE_s               *edge;                      //  $p
    struct VERTEX_s             *vertex;                    //  $p
    struct REGION_s             *region;                    //  $p
    struct FACE_s               *front_face;                //  $p
    };
typedef struct SHELL_s    *SHELL;
```

FACE

Table 121 — Face fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of face and member_of_groups of face
tolerance	double	Not used (null double)
next	pointer0	Next back-face in shell
previous	pointer0	Previous back-face in shell
loop	pointer0	Head of singly-linked chain of loops
shell	pointer	Shell of which this is a back-face
surface	pointer0	Surface of face
sense	char	Face sense – positive ('+') or negative ('-')
next_on_surface	pointer0	Next in chain of faces sharing the surface of this face
previous_on_surface	pointer0	Previous in chain of faces sharing the surface of this face
next_front	pointer0	Next front-face in shell
previous_front	pointer0	Previous front-face in shell
front_shell	pointer	Shell of which this is a front-face

```
struct FACE_s                                         //  Face
    {
    int                          node_id;                   //  $d
    union  ATTRIB_GROUP_u        attributes_groups;         //  $p
    double                       tolerance;                 //  $f
    struct FACE_s               *next;                      //  $p
    struct FACE_s               *previous;                  //  $p
    struct LOOP_s               *loop;                      //  $p
    struct SHELL_s              *shell;                     //  $p
    union  SURFACE_u             surface;                   //  $p
    char                         sense;                     //  $c
    struct FACE_s               *next_on_surface;           //  $p
    struct FACE_s               *previous_on_surface;       //  $p
    struct FACE_s               *next_front;                //  $p
    struct FACE_s               *previous_front;            //  $p
    struct SHELL_s              *front_shell;               //  $p
    };
typedef struct FACE_s     *FACE;
```

LOOP

Table 122 — Loop fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of loop

fin	pointer	One of ring of fins of loop
face	pointer	Face of loop
next	pointer0	Next loop in face

Isolated loops

An isolated loop (one consisting of a single vertex) does not refer directly to a vertex, but points to a fin which refers to that vertex. This isolated fin has fin->forward = fin->backward = fin, and fin->other = fin->curve = fin->edge = null. Its sense is not significant. The fin is chained into the chain of fins referencing the isolated vertex.

```
struct LOOP_s                                    //  Loop
    {
    int                          node_id;                      //  $d
    union  ATTRIB_GROUP_u        attributes_groups;            //  $p
    struct FIN_s                *fin;                          //  $p
    struct FACE_s               *face;                         //  $p
    struct LOOP_s               *next;                         //  $p
    };
typedef struct LOOP_s      *LOOP;
```

FIN

Table 123 — Fin fields

Field name	Type	Description
attributes_groups	pointer0	Head of chain of attributes of fin
loop	pointer0	Loop of fin
forward	pointer0	Next fin around loop
backward	pointer0	Previous fin around loop
vertex	pointer0	Forward vertex of fin
other	pointer0	Next fin around edge, clockwise looking along edge
edge	pointer0	Edge of fin
curve	pointer0	For a non-dummy fin of a tolerant edge, this will be a trimmed SP-curve, otherwise null.
next_at_vx	pointer0	Next fin referencing the vertex of this fin
sense	char	Positive (‘+’) if the fin direction is parallel to that of its edge, else negative (‘-’)

Dummy fins

An application will see edges as having any number of fins, including zero. However internally, they have at least two. This is so that the forward and backward vertices of an edge can always be found as edge->fin->vertex and edge->fin->other->vertex respectively - the first one being a positive fin, the second a negative fin. If an edge does not have both a positive and a negative externally-visible fin, **dummy** fins will exist for this purpose. Dummy fins have fin->loop = fin->forward = fin->backward = fin->curve = fin->next_at_vx = null. For example the boundaries of a sheet always have one dummy fin.

```
struct FIN_s                                       //  Fin
    {
    union  ATTRIB_GROUP_u        attributes_groups;             //  $p
    struct LOOP_s               *loop;                          //  $p
    struct FIN_s                *forward;                       //  $p
    struct FIN_s                *backward;                      //  $p
    struct VERTEX_s             *vertex;                        //  $p
```

```
    struct FIN_s               *other;                        //  $p
    struct EDGE_s              *edge;                         //  $p
    union  CURVE_u              curve;                        //  $p
    struct FIN_s               *next_at_vx;                   //  $p
    char                        sense;                        //  $c
    };
typedef struct FIN_s *FIN;
```

VERTEX

Table 124 — Vertex fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of vertex and member_of_groups of vertex
fin	pointer0	Head of singly-linked chain of fins referencing this vertex
previous	pointer0	Previous vertex in body
next	pointer0	Next vertex in body
point	pointer	Point of vertex
tolerance	double	Tolerance of vertex (null-double for accurate vertex)
owner	pointer	Owning body (for non-acorn vertices) or shell (for acorn vertices)

```
union SHELL_OR_BODY_u
    (
    struct BODY_s              *body;
    struct SHELL_s             *shell;
    };
typedef union SHELL_OR_BODY_u SHELL_OR_BODY;

struct VERTEX_s                                  //  Vertex
    {
    int                         node_id;                      //  $d
    union  ATTRIB_GROUP_u       attributes_groups;            //  $p
    struct FIN_s               *fin;                          //  $p
    struct VERTEX_s            *previous;                     //  $p
    struct VERTEX_s            *next;                         //  $p
    struct POINT_s             *point;                        //  $p
    double                      tolerance;                    //  $f
    union  SHELL_OR_BODY_u      owner;                        //  $p
    };
typedef struct VERTEX_s   *VERTEX;
```

EDGE

Table 125 — Edge fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of edge and member_of_groups of edge
tolerance	double	Tolerance of edge (null-double for accurate edges)
fin	pointer	One of singly-linked ring of fins around edge
previous	pointer0	Previous edge in body or shell

next	pointer0	Next edge in body or shell
curve	pointer0	Curve of edge, zero for tolerant edge. If edge is accurate, but any of its vertices are tolerant, this will be a trimmed curve
next_on_curve	pointer0	Next in chain of edges sharing the curve of this edge
previous_on_curve	pointer0	Previous in chain of edges sharing the curve of this edge
owner	pointer	Owning body (for non-wireframe edges) or shell (for wireframe edges)

```
struct EDGE_s                                          //  Edge
    {
    int                          node_id;                       //  $d
    union  ATTRIB_GROUP_u        attributes_groups;             //  $p
    double                       tolerance;                     //  $f
    struct FIN_s                *fin;                           //  $p
    struct EDGE_s               *previous;                      //  $p
    struct EDGE_s               *next;                          //  $p
    union  CURVE_u               curve;                         //  $p
    struct EDGE_s;              *next_on_curve                  //  $p
    struct EDGE_s               *previous_on_curve;             //  $p
    union  SHELL_OR_BODY_u       owner;                         //  $p
    };
typedef struct EDGE_s     *EDGE;
```

F.2.4.3.5 Associated Data

LIST

Table 126 — Associated List

Field name	Type	Description
node_id	int	Zero
list_type	byte	Always 4
notransmit	logical	Ignore
owner	pointer	Owning part
next	pointer0	Ignore
previous	pointer0	Ignore
list_length	int	Length of list (>= 0)
block_length	int	Length of each block of list. Always 20
size_of_entry	int	Ignore
finger_index	int	Any integer between 1 and list->list_length (set to 1 if length is zero). Ignore
finger_block	pointer	Any block e.g. the first one. Ignore
list_block	pointer	Head of singly-linked chain of pointer list blocks

Lists only occur in part data as the list of attributes referenced by a part.

```
[13] typedef enum
    [14] {
    [15] LIS_pointer    = 4
    [16] }
[17] LIS_type_t;
```

```
union LIS_BLOCK_u
     {
     struct POINTER_LIS_BLOCK_s      *pointer_block;
     };
typedef union LIS_BLOCK_u       LIS_BLOCK;

union LIST_OWNER_u
     {
     struct BODY_s                   *body;
     struct ASSEMBLY_s               *assembly;
     struct WORLD_s                  *world;
     };
typedef union LIST_OWNER_u LIST_OWNER;

struct LIST_s                                 //  List Header
     {
     int                              node_id;                    //  $d
     LIS_type_t                       list_type;                  //  $u
     logical                          notransmit;                 //  $l
     union LIST_OWNER_u               owner;                      //  $p
     struct LIST_s                   *next;                       //  $p
     struct LIST_s                   *previous;                   //  $p
     int                              list_length;                //  $d
     int                              block_length;               //  $d
     int                              size_of_entry;              //  $d
     int                              finger_index;               //  $d
     union LIS_BLOCK_u                finger_block;               //  $p
     union LIS_BLOCK_u                list_block;                 //  $p
     };
typedef struct LIST_s *LIST;
```

POINTER_LIS_BLOCK:

Table 127 — Pointer List Block

Field name	Type	Description
n_entries	int	Number of entries in this block (0 <= n_entries <= 20). Only the first block may have n_entries = 0.
index_map_offset	int	Shall be set to 0
next_block	pointer0	Next pointer list block in chain
Entries[20]	pointer0	Pointers in block, those beyond n_entries shall be zero

When the pointer_lis_block is used as the root node in XT data containing more than one part, the restriction n_entries <= 20 does not apply.

The index_map_offset field is used for Indexed Transmit; applications writing XT data shall ensure this field is set to 0.

```
struct POINTER_LIS_BLOCK_s                           //  Pointer List
     {
     int                               n_entries;              //  $d
     int                              index_map_offset         // $d
     struct POINTER_LIS_BLOCK_s       *next_block;             //  $p
     void                             *entries[ 1 ];           //  $p[]
     };
typedef struct POINTER_LIS_BLOCK_s *POINTER_LIS_BLOCK;
```

ATT_DEF_ID

Table 128 — Attribute Definition ID

Field name	Type	Description
string[]	char	String name e.g. "SDL/TYSA_COLOUR"

```
struct ATT_DEF_ID_s            //  name field type for attrib def.
     {
     char                            String[1];                //  $c[]
     };
typedef struct ATT_DEF_ID_s *ATT_DEF_ID;
```

FIELD_NAMES

Table 129 — Field Names

Field name	Type	Description
names[]	pointer	Array of field names – unicode or char

```
typedef union FIELD_NAME_u
     {
     struct CHAR_VALUES_s         *name
     struct UNICODE_VALUES_s      *uname
     };
       FIELD_NAME_t;

struct FIELD_NAME_s            //  attribute field name
     {
     union FIELD_NAME_u              names[1];                 //  $p[]
     };
typedef struct FIELD_NAME_s *FIELD_NAME;
```

ATTRIB_DEF

Table 130 — Attribut definition

Field name	Type	Description
next	pointer0	Next attribute definition. This can be ignored, except in partition data.
identifier	pointer	Pointer to string name
type_id	int	Numeric id, e.g. 8001 for colour. 9000 for user-defined attribute definitions
actions[8]	byte	Required actions on various events
field_names	pointer0	Names of fields (unicode or char)
legal_owners[14]	logical	Allowed owner types
fields[]	byte	Array of field types. Note that the number of fields is given by the length of the variable length part of this node, i.e. the integer following the node type in the XT data.

The legal_owners array is an array of logicals determining which node types may own this type of attribute.

E.g. if faces are allowed attrib_def -> legal_owners [SCH_fa_owner] = true.

Note that if the XT data contains user fields, the ‘fields’ field of an attribute definition may contain extra values, set to zero. These are to be ignored.

The ‘actions’ field in an attribute definition defines the behaviour of the attribute when an event (rotate, scale, translate, reflect, split, merge, transfer, change) occurs. The actions are in Table 49:

Table 131 — Attribute definition action fields

do_nothing	Leave attribute as it is
delete	Delete the attribute
transform	Transform the transformable fields (point, vector, direction, axis) by appropriate part of transformation
propagate	Copy attribute onto split-off node
keep_sub_dominant	Move attribute(s) from deleted node onto surviving node in a merge, but any such attributes already on the surviving node are deleted.
keep_if_equal	Keep attribute if present on both nodes being merged, with the same field values.
combine	Move attribute(s) from deleted node onto surviving node, in a merge

The XT attribute classes 1-7 correspond as follows:

Table 132 — Corresponding attribute classes

	split	merge	transfer	change	Rotate	scale	translate	reflect
class 1	propagate	keep_equal	do_nothing	do_nothing	do_nothing	do_nothing	do_nothing	do_nothing
class 2	delete	delete	delete	delete	do_nothing	delete	do_nothing	do_nothing
class 3	delete	delete	delete	delete	Delete	delete	delete	delete
class 4	propagate	keep_equal	do_nothing	do_nothing	Transform	transform	transform	transform
class 5	delete	delete	delete	delete	Transform	transform	transform	transform
class 6	propagate	combine	do_nothing	do_nothing	do_nothing	do_nothing	do_nothing	do_nothing
class 7	propagate	combine	do_nothing	do_nothing	Transform	transform	transform	transform

```
typedef enum
    {
    SCH_rotate       = 0,
    SCH_scale        = 1,
    SCH_translate    = 2,
    SCH_reflect      = 3,
    SCH_split        = 4,
    SCH_merge        = 5,
    SCH_transfer     = 6,
    SCH_change       = 7,
```

```
    SCH_max_logged_event     //  last entry; value in $d[] code for
                             actions
    }
    SCH_logged_event_t;

typedef enum
    {
    SCH_do_nothing   = 0,
    SCH_delete       = 1,
    SCH_transform    = 2,
    SCH_propagate              = 3,
    SCH_keep_sub_dominant = 4,
    SCH_keep_if_equal   = 5,
    SCH_combine      = 6
    }
    SCH_action_on_fields_t;

typedef enum
    {
    SCH_as_owner  = 0,
    SCH_in_owner  = 1,
    SCH_by_owner  = 2,
    SCH_sh_owner  = 3,
    SCH_fa_owner  = 4,
    SCH_lo_owner  = 5,
    SCH_ed_owner  = 6,
    SCH_vx_owner  = 7,
    SCH_fe_owner  = 8,
    SCH_sf_owner  = 9,
    SCH_cu_owner  = 10,
    SCH_pt_owner  = 11,
    SCH_rg_owner  = 12,
    SCH_fn_owner  = 13,
    SCH_max_owner            //  last entry; value in $l[] for
                             .legal_owners
    } SCH_attrib_owners_t;

typedef enum
    {
    SCH_int_field            = 1,
    SCH_real_field          = 2,
    SCH_char_field         = 3,
    SCH_point_field        = 4,
    SCH_vector_field      = 5,
    SCH_direction_field  = 6,
    SCH_axis_field         = 7,
    SCH_tag_field          = 8,
    SCH_pointer_field    = 9,
    SCH_unicode_field   = 10
    } SCH_field_type_t;

struct ATTRIB_DEF_s           //  attribute definition
    {
    struct ATTRIB_DEF_s        *next;                        //  $p
    struct ATT_DEF_ID_s        *identifier;                  //  $p
    int                         type_id;                     //  $d
    SCH_action_on_fields_t      actions                      //  $u[8]
                               [(int)SCH_max_logged_event]
                               ;
    struct FIELD_NAMES_s       *field_names                  //  $p
    logical                     legal_owners                 //  $l[14]
                                [(int)SCH_max_owner];
    SCH_field_type_t            fields[1];                   //  $u[]
    };
typedef struct ATTRIB_DEF_s    *ATTRIB_DEF;
```

ATTRIBUTE

Table 133 — Attribute fields

Field name	Type	Description
node_id	int	Node-id
definition	pointer	Attribute definition
owner	pointer	Attribute owner
next	pointer0	Next attribute, group, or member_of_group
previous	pointer0	Previous ditto
next_of_type	pointer0	Next attribute of this type in this part
previous_of_type	pointer0	Previous attribute of this type in this part
fields[]	pointer	Fields, of type int_values etc. The number of fields is given by the length of the variable part of the node. There may be no fields.

The attributes of a node are chained using the next and previous pointers in the attribute. The attribute_groups pointer in the node points to the head of this chain. This chain also contains the member_of_groups of the node.

Attributes within the same part, with the same attribute definition, are chained together by the next_of_type and previous_of_type pointers. The part points to the head of this chain as follows. The attribute_chains pointer in the part points to a list which contains the heads of these attribute chains, one for each attribute definition which has attributes in the part. The list may be null.

Note that the attributes_groups chains in parts, groups and nodes contain the following types of node:

Part:	attributes and groups
Group:	attributes
Node:	attributes and member_of_groups

```
union ATTRIBUTE_OWNER_u
      {
    struct ASSEMBLY_s        *assembly;
    struct INSTANCE_s        *instance;
    struct BODY_s            *body;
    struct SHELL_s           *shell;
    struct REGION_s          *region;
    struct FACE_s            *face;
    struct LOOP_s            *loop;
    struct EDGE_s            *edge;
    struct FIN_s             *fin;
    struct VERTEX_s          *vertex;
    union  SURFACE_u          Surface;
    union  CURVE_u            Curve;
    struct POINT_s           *point;
    struct GROUP_s           *group;
    };
typedef union ATTRIBUTE_OWNER_u ATTRIBUTE_OWNER;

union FIELD_VALUES_u
    {
    struct INT_VALUES_s          *int_values;
    struct REAL_VALUES_s         *real_values;
    struct CHAR_VALUES_s         *char_values;
    struct POINT_VALUES_s        *point_values;
    struct VECTOR_VALUES_s       *vector_values;
    struct DIRECTION_VALUES_s    *direction_values;
```

```
    struct AXIS_VALUES_s          *axis_values;
    struct TAG_VALUES_s           *tag_values;
    struct UNICODE_VALUES_s       *unicode_values;
    };
typedef union FIELD_VALUES_u FIELD_VALUES;

struct ATTRIBUTE_s                             //  Attribute
    {
    int                                 node_id;                //  $d
    struct ATTRIB_DEF_s                *definition;             //  $p
    union  ATTRIBUTE_OWNER_u            owner;                  //  $p
    union  ATTRIB_GROUP_u               next;                   //  $p
    union  ATTRIB_GROUP_u               previous;               //  $p
    struct ATTRIBUTE_s                 *next_of_type;           //  $p
    struct ATTRIBUTE_s                 *previous_of_type;       //  $p
    union  FIELD_VALUES_u               fields[1];              //  $p[]
    };
typedef struct ATTRIBUTE_s *ATTRIBUTE;
```

INT_VALUES

Table 134 — Integer values

values[]	int	Integer values

```
struct INT_VALUES_s                        //  Int values
    {
    int                        values[1];                    //  $d[]
    };
typedef struct INT_VALUES_s *INT_VALUES;
```

REAL_VALUES

Table 135 — Real values

values[]	double	Real values

```
struct REAL_VALUES_s                       //  Real values
    {
    double                     values[1];                    //  $f[]
    };
typedef struct REAL_VALUES_s *REAL_VALUES;
```

CHAR_VALUES

Table 136 — Character values

values[]	char	Character values

```
struct CHAR_VALUES_s                       //  Character values
    {
    char                       values[1];                    //  $c[]
    };
typedef struct CHAR_VALUES_s *CHAR_VALUES;
```

UNICODE_VALUES

Table 137 — Unicode values

values[]	short	Unicode character values

```
struct UNICODE_VALUES_s                    //  Unicode character values
    {
    short                    values[1];                   //  $w[]
    };
typedef struct UNICODE_VALUES_s *UNICODE_VALUES;
```

POINT_VALUES

Table 138 — Point values

values[]	vector	Point values

```
struct POINT_VALUES_s                   //  Point values
     {
    vector                 values[1];                    // $v[]
    };
typedef struct POINT_VALUES_s *POINT_VALUES;
```

VECTOR_VALUES

Table 139 — Vector values

values[]	vector	Vector values

```
struct VECTOR_VALUES_s                  //  Vector values
    {
    vector                 values[1];                    //  $v[]
    };
typedef struct VECTOR_VALUES_s *VECTOR_VALUES;
```

DIRECTION_VALUES

Table 140 — Direction values

values[]	vector	Direction values

```
struct DIRECTION_VALUES_s               //  Direction values
    {
    vector                 values[1];                    //  $v[]
    };
typedef struct DIRECTION_VALUES_s *DIRECTION_VALUES;
```

AXIS_VALUES

Table 141 — Axis values

values[]	vector	Axis values

Note that an axis takes up two vectors.

```
struct AXIS_VALUES_s                    //  Axis values
    {
```

```
    vector                  values[1];                          //  $v[]
    };
typedef struct AXIS_VALUES_s *AXIS_VALUES;
```

TAG_VALUES

Table 142 — Tag values

values[]	int	Integer tag values

The tag field type and the tag_values node are not available for use in user-defined attributes, they occur only in certain system attributes.

```
struct TAG_VALUES_s                     //  Tag values
    {
    int                     values[1];                          //  $t[]
    };
typedef struct TAG_VALUES_s *TAG_VALUES;
```

GROUP

Table 143 — Group fields

Field name	Type	Description
node_id	int	Node-id
attributes_groups	pointer0	Head of chain of attributes of this group
owner	pointer	Owning part
next	pointer0	Next group or attribute
previous	pointer0	Previous group or attribute
type	byte	Type of node allowed in group
first_member	pointer0	Head of chain of member_of_group nodes in group

The groups in a part are chained by the next and previous pointers in a group. The attributes_groups pointer in the part points to the head of the chain. This chain also contains the attributes attached directly to the part - groups and attributes are intermingled in this chain, the order is not significant.

Each group has a chain of member_of_groups. These are chained together using the next_member and previous_member pointers. The first_member pointer in the group points to the head of the chain. Each member_of_group has an owning_group pointer which points back to the group.

Each member_of_group has an owner pointer which points to a node. Thus the group references its member nodes via the member_of_groups.

The member_of_groups which refer to a particular node are chained using the next and previous pointers in the member_of_group. The attributes_groups pointer in the node points to the head of this chain. This chain also contains the attributes attached to the node.

```
typedef enum
    {
    SCH_instance_fe   = 1,
    SCH_face_fe       = 2,
    SCH_loop_fe       = 3,
    SCH_edge_fe       = 4,
    SCH_vertex_fe     = 5,
```

```
    SCH_surface_fe    = 6,
    SCH_curve_fe      = 7,
    SCH_point_fe      = 8,
    SCH_mixed_fe      = 9,
    SCH_region_fe     = 10
    }  SCH_group_type_t;

struct GROUP_s                        //  Group
    {
    int                              node_id;                   //  $d
    union   ATTRIB_GROUP_u           attributes_groups;         //  $p
    union   PART_u                   owner;                     //  $p
    union   ATTRIB_GROUP_u           next;                      //  $p
    union   ATTRIB_GROUP_u           previous;                  //  $p
    SCH_group_type_t                 type;                      //  $u
    struct  MEMBER_OF_GROUP_s       *first_member;              //  $p
    };
typedef struct GROUP_s *GROUP;
```

MEMBER_OF_GROUP

Table 144 — Group member fields

Field name	Type	Description
dummy_node_id	int	Entity label
owning_group	pointer	Owning group
owner	pointer	Referenced member of group
next	pointer0	Next attribute, group or member_of_group
previous	pointer0	Previous ditto
next_member	pointer0	Next member_of_group in this group
previous_member	pointer0	Previous ditto

```
union GROUP_MEMBER_u
     {
    struct INSTANCE_s          *instance;
    struct FACE_s              *face;
    struct REGION_s            *region;
    struct LOOP_s              *loop;
    struct EDGE_s              *edge;
    struct VERTEX_s            *vertex;
    union  SURFACE_u            surface;
    union  CURVE_u              curve;
    struct POINT_s             *point;
    };
typedef union GROUP_MEMBER_u GROUP_MEMBER;

struct MEMBER_OF_GROUP_s              //  Member of group
    {
    int                              dummy_node_id;             //  $d
    struct GROUP_s                  *owning_group;              //  $p
    union  GROUP_MEMBER_u            owner;                     //  $p
    union  ATTRIB_GROUP_u            next;                      //  $p
    union  ATTRIB_GROUP_u            previous;                  //  $p
    struct MEMBER_OF_GROUP_s        *next_member;               //  $p
    struct MEMBER_OF_GROUP_s        *previous_member;           //  $p
    };
typedef struct MEMBER_OF_GROUP_s *MEMBER_OF_GROUP;
```

Node Types

Table 145 — Node types

Node Name	Node Type	Visible at PK	Has Node-ID
ASSEMBLY	10	Yes	No
INSTANCE	11	Yes	Yes
BODY	12	Yes	No
SHELL	13	Yes	Yes
FACE	14	Yes	Yes
LOOP	15	Yes	Yes
EDGE	16	Yes	Yes
FIN	17	Yes	No
VERTEX	18	Yes	Yes
REGION	19	Yes	Yes
POINT	29	Yes	Yes
LINE	30	Yes	Yes
CIRCLE	31	Yes	Yes
ELLIPSE	32	Yes	Yes
INTERSECTION	33	Yes	Yes
CHART	40	No	
LIMIT	41	No	
BSPLINE_VERTICES	45	No	
PLANE	50	Yes	Yes
CYLINDER	51	Yes	Yes
CONE	52	Yes	Yes
SPHERE	53	Yes	Yes
TORUS	54	Yes	Yes
BLENDED_EDGE	56	Yes	Yes
BLEND_BOUND	59	No	
OFFSET_SURF	60	Yes	Yes
SWEPT_SURF	67	Yes	Yes
SPUN_SURF	68	Yes	Yes
LIST	70	Yes	Yes
POINTER_LIS_BLOCK	74	No	
ATT_DEF_ID	79	No	
ATTRIB_DEF	80	Yes	No
ATTRIBUTE	81	Yes	Yes
INT_VALUES	82	No	
REAL_VALUES	83	No	
CHAR_VALUES	84	No	
POINT_VALUES	85	No	
VECTOR_VALUES	86	No	
AXIS_VALUES	87	No	
TAG_VALUES	88	No	
DIRECTION_VALUES	89	No	
GROUP	90	Yes	Yes
MEMBER_OF_GROUP	91	No	
UNICODE_VALUES	98	No	
FIELD_NAMES	99	No	

TRANSFORM	100	Yes	No
WORLD	101	No	
KEY	102	No	
PE_SURF	120	Yes	Yes
INT_PE_DATA	121	No	
EXT_PE_DATA	122	No	
B_SURFACE	124	Yes	No
SURFACE_DATA	125	No	
NURBS_SURF	126	No	
KNOT_MULT	127	No	
KNOT_SET	128	No	
PE_CURVE	130	Yes	Yes
TRIMMED_CURVE	133	Yes	Yes
B_CURVE	134	Yes	Yes
CURVE_DATA	135	No	
NURBS_CURVE	136	No	
SP_CURVE	137	Yes	Yes
GEOMETRIC_OWNER	141	No	
HELIX_SU_FORM	163	No	
HELIX_CU_FORM	184	No	
POLYLINE	200	Yes	Yes
MESH	201	Yes	Yes
INTERSECTION_DATA	204	No	No

Node Classes

Table 146 — Node classes

Node class name	Node class
GEOMETRY	1003
PART	1005
SURFACE	1006
SURFACE_OWNER	1007
CURVE	1008
CURVE_OWNER	1010
POINT_OWNER	1011
LIS_BLOCK	1012
LIST_OWNER	1013
ATTRIBUTE_OWNER	1015
GROUP_OWNER	1016
GROUP_MEMBER	1017
FIELD_VALUES	1018
ATTRIB_GROUP	1019
TRANSFORM_OWNER	1023
PE_DATA	1027
PE_INT_GEOM	1028
SHELL_OR_BODY	1029
FIELD_NAME	1037

F.2.4.3.6 System Attribute Definitions

All system attribute definitions are of class 1.

Hatching Attributes

Hatching

Table 147 — Hatching

Identifier	SDL/TYSA_HATCHING	
Type_id	8003	
Entity types	face	
Fields	real	real 1
		real 2
		real 3
		real 4
	integer	Hatching type
Set by	Application	

For **planar hatching -** the four real values define the hatch orientation as a vector and a spacing between consecutive planes.

For **radial hatching** - the first three real values define the spacing of the hatch lines. The fourth value is not used.

For **parametric hatching** - the first two real values define the spacing in *u* and *v* respectively. The last two values are not used.

Planar Hatch

Table 148 — Planar Hatch

Identifier	SDL/TYSA_PLANAR_HATCH		
Type_id	8021		
Entity types	face		
Fields	real	x component	‘direction’ or plane normal
		y component	
		z component	
		‘pitch’ or separation	
		x component	position vector
		y component	
		z component	
Set by	Application		

For planar hatching, an attribute with this definition takes precedence over an attribute with the SDL/TYSA_HATCHING definition, if a face has both types of attribute attached.

Radial Hatch

Table 149 — Radial Hatch

Identifier	SDL/TYSA_RADIAL_HATCH	
Type_id	8027	
Entity types	face	
Fields	real	radial around
		radial along
		radial about
		radial around start
		radial along start
		radial about start
Set by	Application	

For radial hatching, an attribute with this definition takes precedence over an attribute with the SDL/TYSA_HATCHING definition, if a face has both types of attribute attached.

Parametric Hatch

Table 150 — Parametric Hatch

Identifier	SDL/TYSA_PARAM_HATCH	
Type_id	8028	
Entity types	face	
Fields	real	u spacing
		v spacing
		u start
		v start
Set by	Application	

For parametric hatching, an attribute with this definition takes precedence over an attribute with the SDL/TYSA_HATCHING definition, if a face has both types of attribute attached.

Density Attributes

There are density attributes for each of regions, faces, edges and vertices in addition to the system attribute for density of a body.

The region/face/edge/vertex attributes will be taken into account when finding the mass, centre of gravity and moment of inertia of a body or of the entity to which the attribute is attached:

- The mass of a region will not include that of any of its faces or edges, and the same applies to faces and edges and their boundaries.
- A void region will always have zero mass whatever its density and a solid region will inherit its density from the body if it does not have a density of its own.
- The default density for faces, edges and vertices is always zero.

Density (of a body)

Table 151 — Body Density

Identifier	SDL/TYSA_DENSITY	
Type_id	8004	
Entity types	body	
Fields	real	Density
	string	Units
Set by	Application	

A body without a density attribute is taken to have, by default, a density of 1.0.

The character field units can be set and read by the application.

Region Density

Table 152 — Region Density

Identifier	SDL/TYSA_REGION_DENSITY	
Type_id	8023	
Entity types	region	
Fields	real	Density of region
	string	Units
Set by	Application	

This attribute only makes sense for solid regions; void regions always have a mass of zero.

A solid region without a density attribute is taken to have, by default, the same density as its owning body.

The character field units can be set and read by the user.

Face Density

Table 153 — Face Density

Identifier	SDL/TYSA_FACE_DENSITY	
Type_id	8024	
Entity types	face	
Fields	real	Density of face
	string	Units
Set by	Application	

The value of this attribute is treated as a mass per unit area.

A mass will be calculated for a face only when a face possesses this attribute. In all other cases the mass of a face is not defined.

The character field units can be set and read by the user.

Edge Density

Table 154 — Edge Density

Identifier	SDL/TYSA_EDGE_DENSITY	
Type_id	8025	
Entity types	edge	
Fields	real	Density of edge
	string	Units
Set by	Application	

The value of this attribute is treated as a mass per unit length.

A mass will be calculated for an edge only when an edge possesses this attribute. In all other cases the mass of an edge is not defined.

The character field units can be set and read by the user.

Vertex Density

Table 155 — Vertex Density

Identifier	SDL/TYSA_VERTEX_DENSITY	
Type_id	8026	
Entity types	vertex	
Fields	real	Mass of vertex
	string	Units
Set by	Application	

The value of this attribute is treated as a point mass.

A mass will be calculated for a vertex only when a vertex possesses this attribute. In all other cases the mass of a vertex is not defined.

The character field units can be set and read by the user.

Region

Table 156 — Region

Identifier	SDL/TYSA_REGION	
Type_id	8013	
Entity types	face	
Fields	string	Unused
Set by	Application	

Regional data will allow the application to analyze a hidden-line picture for distinct regions in the 2D view.

Colour

Table 157 — Colour

Identifier	SDL/TYSA_COLOUR		
Token	8001		
Entity types	face edge		
Fields	real	Red value	These three values should be in the range 0.0 to 1.0
		Green value	
		Blue value	
Set by	Application		

Reflectivity

Table 158 — Reflectivity

Identifier	SDL/TYSA_REFLECTIVITY	
Token	8014	
Entity types	face	
Fields	real	Coefficient of specular reflection
		Proportion of coloured light in highlights
		Coefficient of diffuse reflection
		Coefficient of ambient reflection
	integer	Reflection power
Set by	Application	

The attribute types for Reflectivity and Translucency are also used by the Parasolid routine RRPIXL, but the use of this routine is not recommended.

Translucency

Table 159 — Translucency

Identifier	SDL/TYSA_TRANSLUCENCY		
Token	8015		
Entity types	face		
Fields	real	Transparency coefficient	range 0.0 to 1.0, where 0 is opaque and 1 is transparent
Set by	Application		

Name

Table 160 — Name

Identifier	SDL/TYSA_NAME
Token	8017
Entity types	assembly, body, instance, shell, face, loop, edge, vertex, group,

	surface, curve, point	
Fields	string	Name of entity
Set by	Application	

.

Incremental faceting

Table 161 — Incremental faceting

Identifier	SDL/TYSA_INCREMENTAL_FACETTING	
Token	8030	
Entity types	face	
Fields	string	Unused
Set by	XT incremental faceting/Application	

Transparency

Table 162 — Transparency

Identifier	SDL/TYSA_TRANSPARENCY	
Token	8029	
Entity types	Body, face	
Fields	integer	Non-zero transparency coefficient value is transparent
Set by	Application	

A body may be rendered transparent if it has an attached transparency attribute with a non-zero transparency coefficient.

Non-mergeable edges

Table 163 — Non-mergable edges

Identifier	SDL/TYSA_NO_MERGE	
Token	8032	
Entity types	edge	
Fields	string	Unused
Set by	Application	

If an edge has an attribute of this definition attached, it indicates that the edge should not be merged in any modelling operations.

Group merge behaviour

Table 164 — Group merge behaviour

Identifier	SDL/TYSA_GROUP_MERGE

Token	8037	
Entity types	group	
Fields	string	Unused
Set by	Application	

If a group has an attribute of this definition attached, it indicates that alternative behaviour should be used if an entity in the group is merged with an entity not in that group.

Unicode name

Table 165 — Unicode name

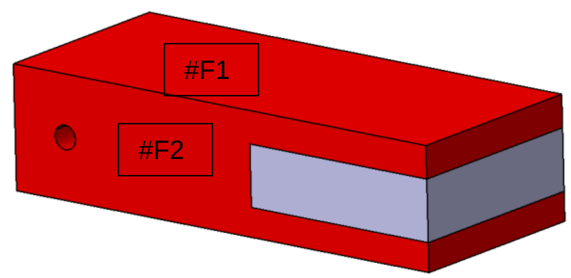
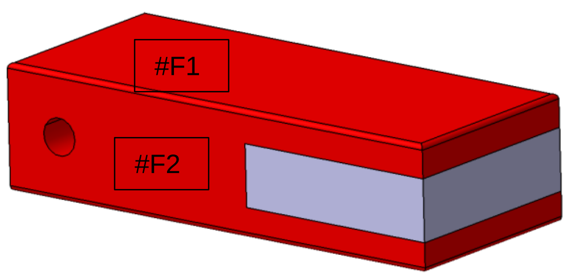
Identifier	SDL/TYSA_UNAME	
Token	8038	
Entity types	assembly, body, instance, shell, face, loop, edge, vertex, group, surf, curve, point, region	
Fields	ustring	Name of entity
Set by	Application	

If a group has an attribute of this definition attached, it indicates that alternative behaviour should be used if an entity in the group is merged with an entity not in that group.

F.2.5 XT Moniker Attributes

In order to allow design in context based on JT files, it is mandatory to provide a stable possibility to recognize a particular B-Rep element, especially faces, for external links.

Consider a design change as shown in the following pictures:

	
Original design	Design after some modifications

In case of an external reference to one of the faces, i.e. the top face #F1, it must be possible to reconsider the proper face when replacing the part. Otherwise, the external link requires an interactive reassignment.

Internally, CAD systems typically maintain names or identifiers for B-Rep entities that persist across modelling operations. These persistent names are called “Moniker IDs” in context of JT.

Note: Moniker IDs are not 100% reliable in any CAD system. If a very extensive design change is performed, some of the Moniker IDs may not be preserved.

This kind of mechanism can only work when the two JT files (original and modified designed) have been converted with the same convertor (and settings) or at least the same algorithm for adding the Moniker ID. Furthermore, it must be possible to retrieve a stable persistent ID from the original CAD system.

Moniker IDs

Moniker attributes were created to enable JT adopters with the ability to keep track of design changes made to the solid bodies stored in the XT segment of a JT file. For example, a CAD system that imports JT data that is generated from an updated version of an existing CAD model can use moniker data to determine which bodies in the XT segment of the JT file have changed. These attributes are defined to enable consistent use of JT with XT geometry for data exchange. Moniker attributes must be uniquely defined and persisted to be effective.

Moniker attributes follow the standard convention for creation of attributes in the XT segment of JT.

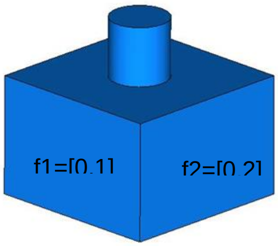
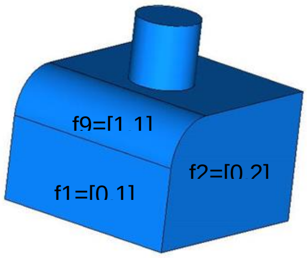
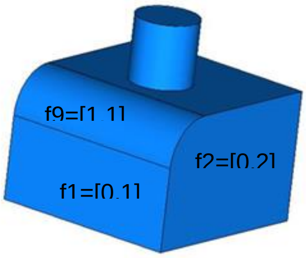
MONIKER/GUID_TABLE_ATTRIB and MONIKER/MONIKER_DATA_ATTRIB

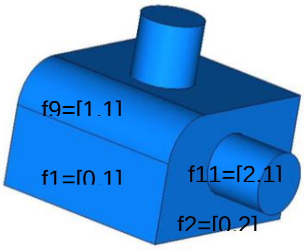
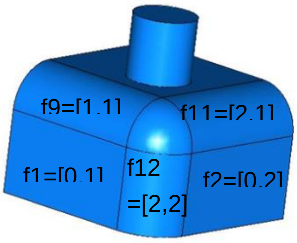
The MONIKER/GUID_TABLE_ATTRIB and MONIKER/MONIKER_DATA_ATTRIB attributes are used to store data in the XT segment of a JT file that records unique identifiers for XT faces and bodies with index values to link them together.

Unique identifiers for bodies are assigned as GUID strings for each version of a body. Unique identifiers for faces are assigned as 32-bit integers. An index field is used to tie the identifier of a body to the identifier of a face. For every version of a body, a set of MONIKER/GUID_TABLE_ATTRIB and MONIKER/MONIKER_DATA_ATTRIB attributes will be defined to record that version’s identifiers. For example, a model might start with a body that has 6 faces. That body will have one MONIKER/GUID_TABLE_ATTRIB attributes and 6 MONIKER/MONIKER_DATA_ATTRIB attributes. If a change is made that adds one face to the body, that body then will have a new version. There will be 2 MONIKER/GUID_TABLE_ATTRIB attributes and 7 MONIKER/MONIKER_DATA_ATTRIB attributes. The information in these attributes is used to form an indexed table of values for bodies and faces. The indices in the first body version will relate 6 faces to that version's GUID string. The index value in the second version of the body will tie one face ID to that version’s GUID string.

Table 166 — Moniker attribute use example shows how moniker attributes are used. Each version of a body has a unique GUID. The face identifiers reflect their GUID and index values.

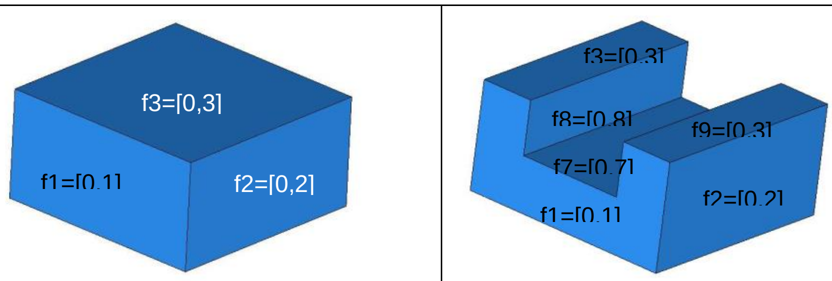
Table 166 — Moniker attribute use example

<table>
<tr><th>Operation</th><th>Application X</th><th>Application Y</th><th>Operation</th></tr>
<tr>
<td>Application X constructs initial model.
Single entry in GUID table
Part 1.0
<table><tr><th>GUID</th><th>Index</th></tr><tr><td>GUID-A</td><td>0</td></tr><tr><td></td><td></td></tr></table></td>
<td>
</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application X modifies model with blend
Adds entry in GUID Table
New f9 moniker on blend Face uses new GUID index.
Part 1.1
<table><tr><th>GUID</th><th>Index</th></tr><tr><td>GUID-A</td><td>0</td></tr><tr><td>GUID-B</td><td>1</td></tr></table></td>
<td>
</td>
<td>
</td>
<td>Part imported into Application Y
Monikers are unchanged on imported part.

Part 2.0
<table><tr><th>GUID</th><th>Index</th></tr><tr><td>GUID-A</td><td>0</td></tr><tr><td>GUID-B</td><td>1</td></tr></table></td>
</tr>
<tr>
<td>Application X adds a boss.
New entry in GUID table is unique. New boss faces reference new GUID index
Part 1.2
<table><tr><th>GUID</th><th>Index</th></tr><tr><td>GUID-A</td><td>0</td></tr><tr><td>GUID-B</td><td>1</td></tr><tr><td>GUID-C</td><td>2</td></tr></table></td>
<td>
</td>
<td>
</td>
<td>Application Y adds 2 blends.
New entry in GUID table is unique.
New blend face monikers reference new GUID
Part 2.1
<table><tr><th>GUID</th><th>Index</th></tr><tr><td>GUID-A</td><td>0</td></tr><tr><td>GUID-B</td><td>1</td></tr><tr><td>GUID-D</td><td>2</td></tr></table></td>
</tr>
</table>

If during a modelling operation a moniker bearing face is split into multiple faces then the moniker attribute is carried over to all the resulting faces (see Figure 160 — Split a face).

Figure 160 — Split a face

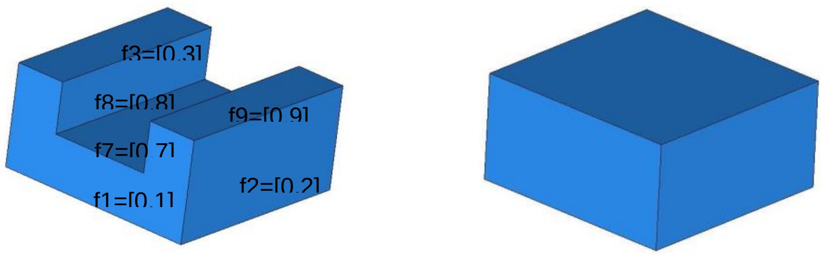


Guid	Index
Guid-A	0

Guid	Index
Guid-A	0

When multiple entities get merged in a modelling operation, the moniker attributes from both entities are copied to the new entity. The moniker data from the input faces should not match (see Figure 161 — Merge faces)

Figure 161 — Merge faces



Guid	Index
Guid-A	0

Guid	Index
Guid-A	0

Table 167 — MONIKER/GUID_TABLE_ATTRIB and MONIKER/MONIKER_DATA_ATTRIB provides attribute data descriptions.

Table 167 — MONIKER/GUID_TABLE_ATTRIB and MONIKER/MONIKER_DATA_ATTRIB

XT Attribute name	Description
MONIKER/GUID_TABLE_ATTRIB	An XT body can have multiple occurrences of this attribute. This attribute contains 3 fields: 1. GUID string – GUID (Globally Unique IDentifier). Each version of an XT body will have a GUID string. The string consists of a series of hexadecimal values separated by a “-“. The GUID string syntax appears this way when using the C printf convention for strings; “{%08x-%04x-%04x-%02x%02x-%02x%02x%02x%02x%02x%02x}” 2. Index – a 32 bit integer. Corresponds with the index field value in MONIKER/MONIKER_DATA_ATTRIB. 3. Application name string – optional string used to record the application that authored the attribute
MONIKER/MONIKER_DATA_ATTRIB	Each face can have in an XT segment can have this attribute. Each attribute contains 3 fields: 1. Index – a 32 bit integer. Correspond with the index field value in MONIKER/GUID_TABLE_ATTRIB. 2. Entity Id - a 32 bit integer face identifier. 3. Label string – optional string for face specific information.

MONIKER/BODY_ID_ATTRIB

MONIKER/BODY_ID_ATTRIB is a standalone attribute with no dependencies to the MONIKER/GUID_TABLE_ATTRIB or MONIKER/MONIKER_DATA_ATTRIB attributes. It is created for each version of an XT body in the XT segment of a JT file. An XT body can have multiple occurrences of this attribute, each with an id for that version of the body. It is valid for an XT segment of a JT file to contain all three moniker attributes as different CAD systems will use one or the other constructs.

Table 168 — MONIKER/BODY_ID_ATTRIB

XT Attribute name	Description
ONIKER/BODY_ID_ATTRIB	This attribute is created for each version of an XT body in the XT segment of a JT file. It is used to uniquely identify an XT body in the XT segment of a JT file. The Entity Id field is populated with an incrementing integer typically utilized by CAD authoring systems. This attribute consists of 2 fields: 1. Entity Id - 32 bit integer 2. Application name string – optional string used to record the application that authored the attribute

Annex G
JT ULP Segment

JT ULP Segment contains an Element that defines the semi-precise geometric Boundary Representation data for a particular Part in JT ULP format.

JT ULP Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT ULP Segment type supports compression on all element data, so all elements in JT ULP Segment use the Logical Element Header Compressed. form of element header data.

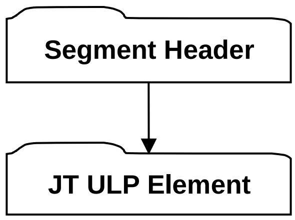


Figure 162 — JT ULP Segment data collection

Complete description for Segment Header can be found in the File Header section of Base Format Description under Data Segment.

G.1 JT ULP Element

Object Type ID: 0xf338a4af, 0xd7d2, 0x41c5, 0xbc, 0xf2, 0xc5, 0x5a, 0x88, 0xb2, 0x1e, 0x73

JT ULP Element represents a particular Part's ultra-lightweight semi-precise B-Rep data. Like JT B-Rep Element or XT B-Rep Element, JT ULP Element contains all the topological and geometric information that describes the shape of a part. The difference is that the size of JT ULP Element is typically around 10% of a typical JT file with B-Rep and LODs, and this is achieved by sophisticated compression techniques. In addition, JT ULP Element is semi-precise meaning that its geometric description is not as precise as either JT B-Rep Element or XT B-Rep Element. The precision loss of JT ULP Element, however, is carefully controlled to be equal to or better than 0.01% of the part size or 0.1mm, whichever is smaller.

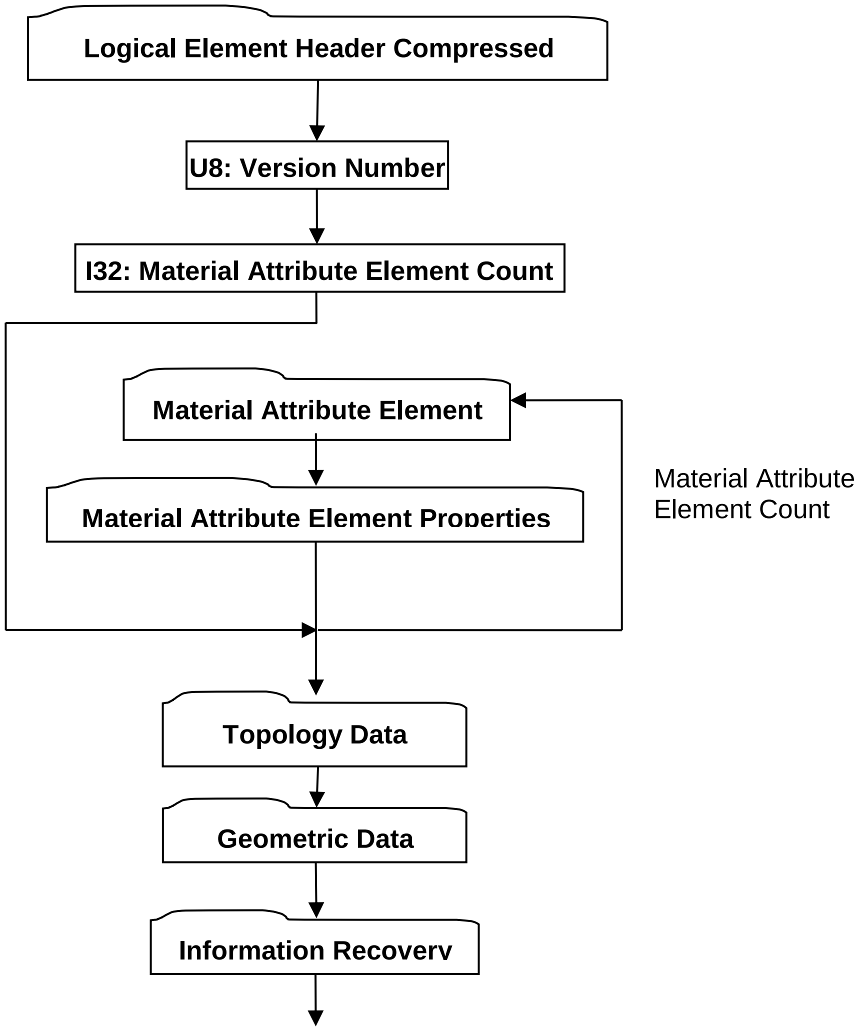


Figure 163 — JT ULP Element data collection

Complete description for Logical Element Header Compressed can be found in the File Header section of Base Format Description under Data Segment, Data.

U8: Version Number

Version Number is the version identifier for this JT ULP Element. Information on local version numbers can be found in the Base Format Description under Common Data Conventions and Constructs Local version numbers.

I32: Material Attribute Element Count

Material Attribute Element Count is the number of material attribute elements.

Complete description for Material Attribute Element can be found in Material Attribute Element.

G.1.1 Topology Data

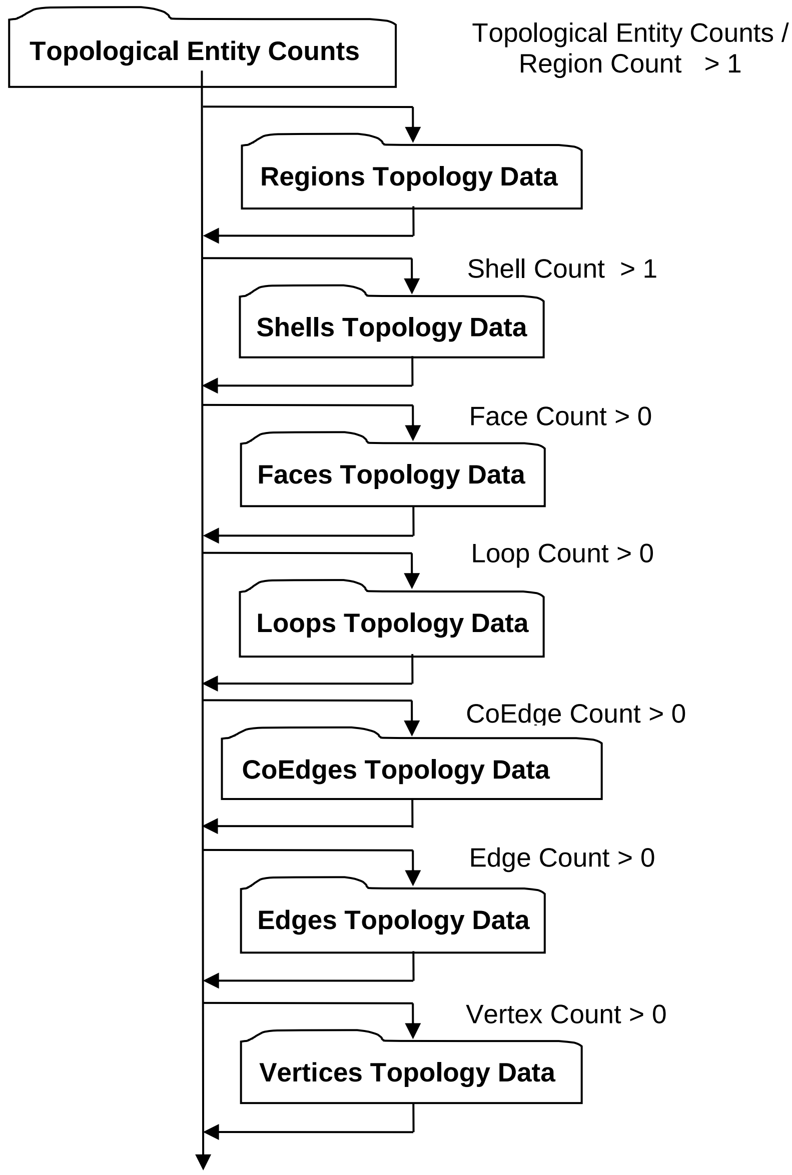


Figure 164 — Topology Data collection

Topological Entity Counts

Topological Entity Counts data collection defines the counts for each of the various topological entities within a ULP.

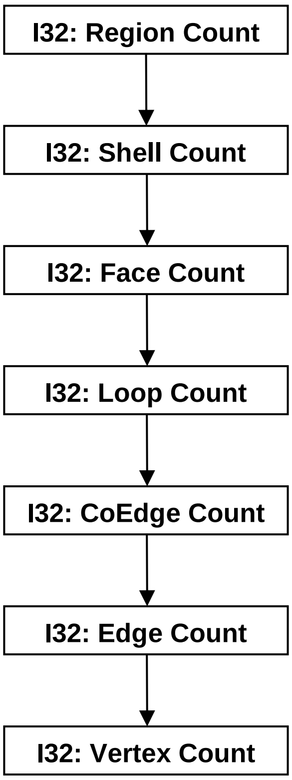


Figure 165 — Topological Entity Counts data collection

I32: Region Count

Region Count indicates the number of topological region entities in the ULP.

I32: Shell Count

Shell Count indicates the number of topological shell entities in the ULP.

I32: Face Count

Face Count indicates the number of topological face entities in the ULP.

I32: Loop Count

Loop Count indicates the number of topological loop entities in the ULP.

I32: CoEdge Count

CoEdge Count indicates the number of topological coedge entities in the ULP.

I32: Edge Count

Edge Count indicates the number of topological edge entities in the ULP.

I32: Vertex Count

Vertex Count indicates the number of topological vertex entities in the ULP.

Combined Predictor Type

A predictor type may be combined with additional processing. When Combined Predictor Type is used, the additional processing step is encoded. For example, combined predictor type *Combined:NULL* means that the data collection follows the logical diagram in the Figure 167 with ePredictorType set to be NULL.

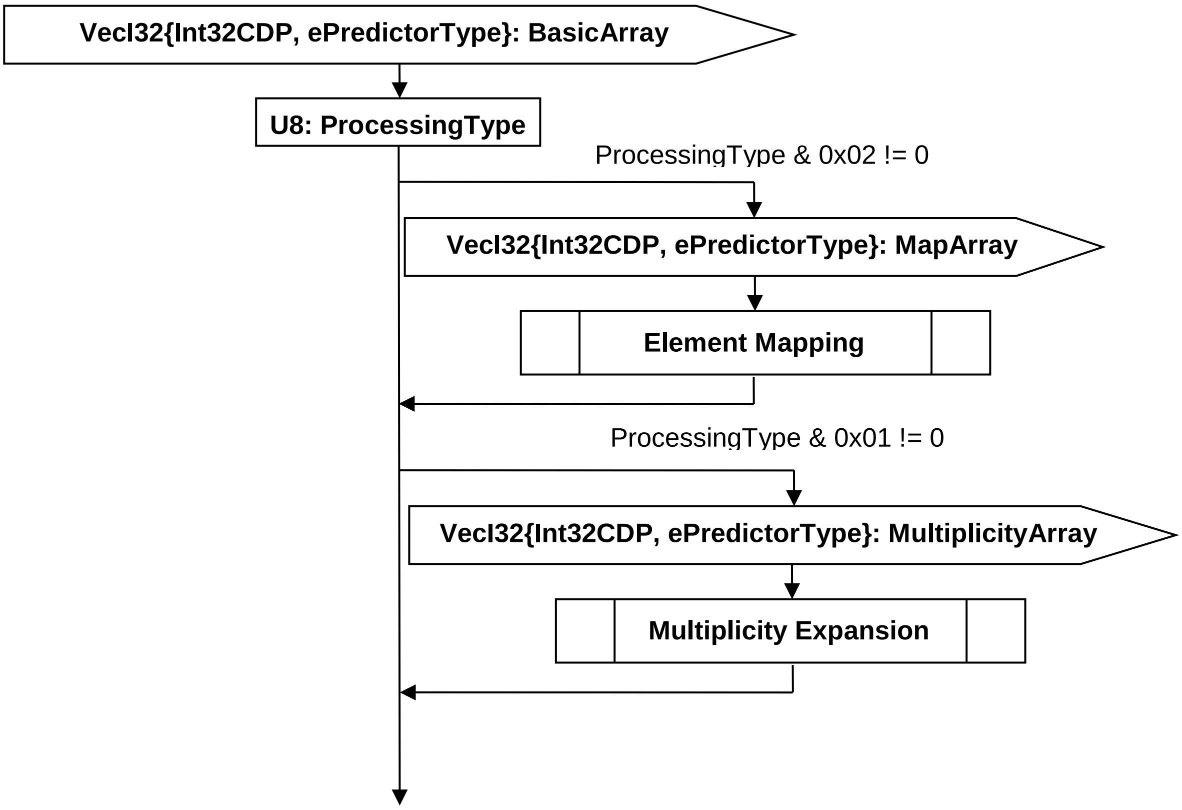


Figure 166 — Combined Predictor Type data collection

VecI32{Int32CDP, ePredictorType}: BasicArray

BasicArray is an integer array, compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U8: ProcessingType

Two bits of this value are currently used. If bit 0x02 is set, then the integer array is a list of elements with unique values and Element Mapping step is needed to recover the original values. If bit 0x01 is set, then the some elements in the integer array may be repeated, and Multiplicity Expansion is used to recover the original values.

VecI32{Int32CDP, ePredictorType}: MapArray

MapArray is an integer array, where each element represents the index mapping information. MapArray is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Element Mapping

Element Mapping recovers the original array from BasicArray and MapArray, using relationship$\mathrm{OriginalArray}[i] = \mathrm{BasicArray}[\mathrm{MapArray}[i]]$.

After Element Mapping, the value of $\mathrm{BasicArray}$ is updated with $\mathrm{OriginalArray}$.

VecI32{Int32CDP, ePredictorType}: MultiplicityArray

MultiplicityArray is an integer array, where each element represents the multiplicity of each element in BasicArray. MultiplicityArray is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Multiplicity Expansion

Multiplicity Expansion recovers the original array from BasicArray and MultiplicityArray. The original array is an expansion of the BasicArray. If the corresponding multiplicity value is greater than 1, the element in BasicArray is contiguously repeated in the original array according to multiplicity value.

Regions Topology Data

Regions Topology Data defines the disjoint set of non-overlapping Shells making up each Region. Each Region is defined by one or more non-overlapping Shells. The volume of a Region is that volume lying inside each “anti-hole Shell” and outside each simply-contained “hole Shell” belonging to the particular Region. A Region is analogous to a dimensionally elevated face where Region corresponds to Face and Shell corresponds to Trim Loop.

Each Region’s defining Shells are identified in a list of Shells by an index for both the first Shell and the last Shell in each Region (i.e. all Shells inclusive between the specified first and last Shell list index define the particular Region). In addition, the indices of all the shells in a single Region are contiguous. The first shell index of the first region is 0, and the first shell index of other regions is one greater than the last shell index of the previous region. Therefore only the number of shells of each region is stored. In the special case when the number of regions is 1, no information needs be stored since its last Shell index is known to be Shell Count-1.

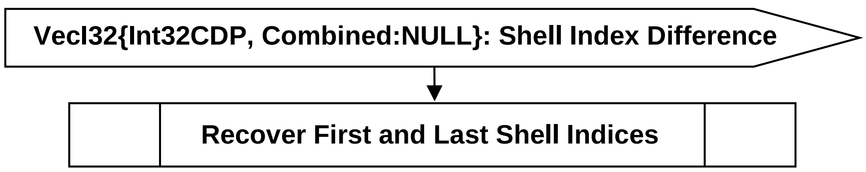


Figure 167 — Regions Topology Data collection

VecI32{Int32CDP, Combined:NULL}: Shell Index Difference

Shell Index Difference is a vector of indices representing the integer value by subtracting first shell index from last shell index in each region, encoded using Combined Predictor Type. Shell Index Difference is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover First and Last Shell Indices

The first shell index of the first region is 0, and the last shell index of the first region is element 0 of Shell Index Difference. The first shell index of region $k, k \geq 1$ equals to the last shell index of region $k-1$ plus 1. The last shell index of region $k, k \geq 1$ equals to the first shell index of region k plus element k of Shell Index Difference array.

Shells Topology Data

Shells Topology Data defines the set of topological adjacent Faces making up each Shell. A Shell’s set of topological adjacent Faces define a single (usually closed) two manifold solid that in turn defines the boundary between the finite volume of space enclosed within the Shell and the infinite volume of space outside the Shell. In addition, each Shell has a flag that denotes whether the Shell refers to the finite interior volume (i.e. a “hole Shell”) or the infinite exterior volume (i.e. an “anti-hole Shell”).

Each Shell’s defining Faces are identified in a list of Faces by an index for both the first Face and the last Face in each Shell (i.e. all Faces inclusive between the specified first and last Face list index define the particular Shell). In addition, the indices of all the faces in a single Shell are contiguous. The first face index of the first shell is 0, and the first face index of other shells is one greater than the last face index of the previous shell. Therefore only the number of faces of each shell is stored. In the special case when the number of shells is 1, no information needs be stored since its last face index is known to be Face Count-1.

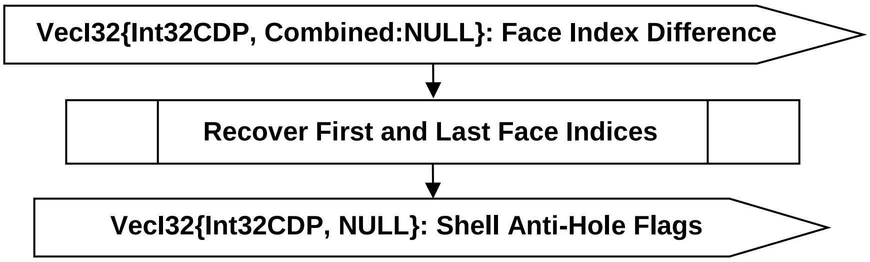


Figure 168 — Shells Topology Data collection

VecI32{Int32CDP, Combined:NULL}: Face Index Difference

Face Index Difference is a vector of indices representing the integer value by subtracting first face index from last face index in each shell, encoded using Combined Predictor Type. Face Index Difference is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover First and Last Face Indices

The first face index of the first shell is 0, and the last face index of the first shell is element 0 of Face Index Difference. The first face index of shell $k, k \geq 1$ equals to the last face index of shell $k-1$ plus 1. The last face index of shell $k, k \geq 1$ equals to the first face index of shell k plus element k of Face Index Difference array.

VecI32{Int32CDP, NULL}: Shell Anti-Hole Flags

Each Shell has a flag identifying whether the Shell is an anti-hole Shell. Shell Anti-Hole Flags is a vector of anti-hole flags for a set of Shells.

In an uncompressed/decoded form the flag values have the following meaning:

Table 169 — JT ULP Shell Anti-Hole Flag values

= 0	Shell is not an anti-hole Shell
= 1	Shell is an anti-hole Shell

Shell Anti-Hole Flags uses the Int32 version of the CODEC to compress and encode data.

Faces Topology Data

A Face shall be trimmed with at least one “anti-hole” Trim Loop and may be trimmed with one or more “hole” Trim Loops. The complete description of face and its relation to the trim loops can be found inFaces Topology Data.

Each Face's defining Trim Loops are identified in a list of trim Loops by an index for both the first Trim Loop and the last Trim Loop in each Face (i.e. all Trim Loops inclusive between the specified first and last Trim Loop list index define the particular Face). In addition, the indices of all the loops in a single Face are contiguous. The first loop index of the first face is 0, and the first loop index of other faces is one greater than the last loop index of the previous face. Therefore only the number of loops of each face is stored. In the special case when the number of faces is 1, no information needs be stored since its last loop index is known to be Loop Count-1.

Each Face's underlying Geometric Surface is identified by an index into a list of Geometric Surfaces. Each face's material is identified by an index into the list of f Material Attribute Elements.

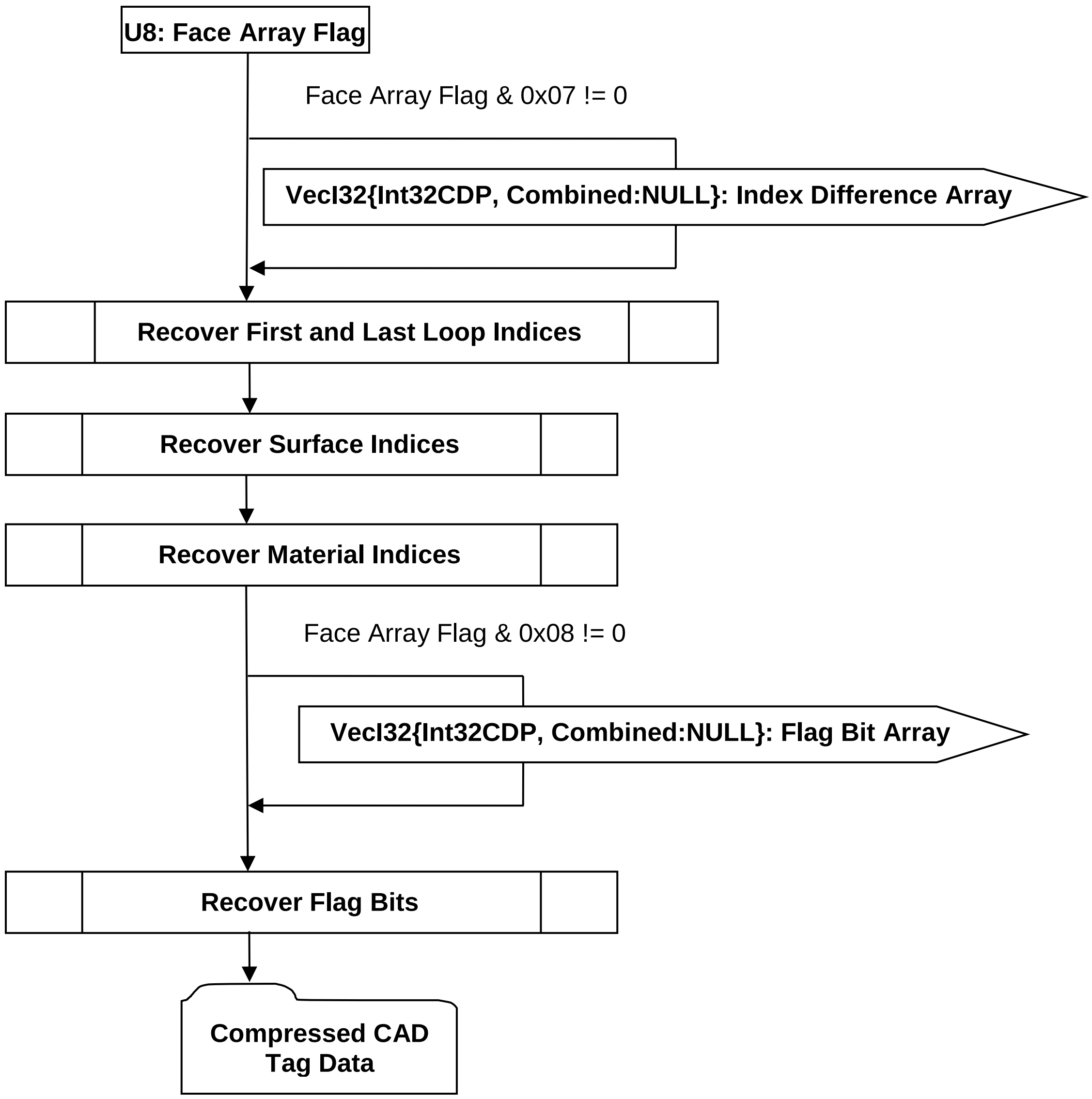


Figure 169 — Faces Topology Data collection

U8: Face Array Flag

Face Array Flag indicates which arrays of face topology data are not trivial and therefore encoded.

VecI32{Int32CDP, Combined:NULL}: Index Difference Array

Index Difference Array is a combined vector of indices encoded using Int32 version of CODEC and Combined Predictor Type, with its content decided by the value of Face Array Flag. If Face Array Flag has bit 0x01 set, then the vector of integer values obtained by subtracting first loop index from last loop index in each face is appended to the end of Index Difference Array. If Face Array Flag has bit 0x02 set, then the vector of integer values obtained by subtracting surface index from face index in each face is appended to the end of Index Difference Array. If Face Array Flag has bit 0x04 set, then the vector of integer values representing the material index of each face is appended to the end of Index Difference Array.

Recover First and Last Loop Indices

The first loop index of the first face is 0, and the last loop index of the first face is element 0 of Index Difference Array if the array is encoded, or 0 if bit 0x01 of Face Array Flag is not set. The first loop index of face $\mathrm{k}, \mathrm{k} \geq 1$ equals to the last loop index of face $\mathrm{k}-1$ plus 1. The last loop index of face $\mathrm{k}, \mathrm{k} \geq 1$ equals to the first loop index of face k plus element k of Index Difference Array, or 0 if bit 0x01 of Face Array Flag is not set.

Recover Surface Indices

The surface index of each face equals to the face index if bit 0x02 of Face Array Flag is not set. Otherwise the surface index of face k is obtained by substracting element $\mathrm{k}+\mathrm{offset}$ of Index Difference Array from face index k, where offset is equal to Face Count if bit 0x01 of Face Array Flag is set and 0 if the bit is not set.

Recover Material Indices

The material index of each face equals to 0 if bit 0x04 of Face Array Flag is not set. Otherwise the material index of face k equals to the element $\mathrm{k}+\mathrm{offset}$ of Index Difference Array, where offset is equal to twice of Face Count if both bit 0x01 and bit 0x02 of Face Array Flag are set, is equal to Face Count if either bit 0x01 or bit 0x02 of Face Array Flag is set, and is equal to 0 if neither bit is set.

VecI32{Int32CDP, Combined:NULL}: Flag Bit Array

Only the lower 24 bits of the four integer indices, namely first loop index, last loop index, surface index, and material index, are used as integer identifiers. The other bits of these integers are documented as reserved for future use.

Table 170 — JT ULP Flag Bit Array Look Index values

<table>
<tr><th></th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>First Loop Index</td><td colspan="3">Surface Type</td><td colspan="2">U Knot Type</td><td colspan="2">V Knot Type</td><td>isNormalReversed</td></tr>
<tr><td>Last Loop Index</td><td>isIsolated</td><td colspan="7">Reserved</td></tr>
<tr><td>Surface Index</td><td colspan="8">Reserved</td></tr>
<tr><td>Material Index</td><td colspan="8">Reserved</td></tr>
</table>

Each element of Flag Bit Array is a 32 bit integer obtained by combining all 32 flag bits from four different integers. More specifically:

- Bits 0~7 of Flag Bit Array are equal to bits 24~31 of First Loop Index.
- Bits 8~15 of Flag Bit Array are equal to bits 24~31 of Last Loop Index.
- Bits 16~23 of Flag Bit Array are equal to bits 24~31 of Surface Index.
- Bits 24~31 of Flag Bit Array are equal to bits 24~31 of Material Index.

Supported Surface Type

In an uncompressed/decoded form, the supported surface types are listed below.

Table 171 — JT ULP Supported Surface Type values

0	Nurbs
1	Plane
2	Cylinder
3	Cone
4	Sphere
5	Torus
6	Reserved
7	Reserved

Supported Knot Type

In an uncompressed/decoded form, the supported knot types are listed below. The knot type of the underlying surface along both U and V parameter directions are encoded.

Table 172 — JT ULP Supported Knot Type Values

0	No Pattern
1	No knot value in between the clamped end knots
2	All knot values in between the end knots increase with an even interval
3	All knot values in between the end knots repeat exactly once, and the distinct values increase with an even interval

In an uncompressed/decoded form, the Face Reverse Normal Flag has the following meaning:

Table 173 — JT ULP Face Reverse Normal Flag values

= 0	Face normal is not reversed
= 1	Face normal is reversed.

Recover Flag Bits

If Face Array Flag & 0x08 is equal to 0, then each element in Flag Bit Array is set to have value 0. The flag bits are recovered by assigning bits 0~7 of Flag Bit Array to bits 24~31 of First Loop Index, bits 8~15 of Flag Bit Array to bits 24~31 of Last Loop Index, bits 16~23 of Flag Bit Array to bits 24~31 of Surface Index, and bits 24~31 of Flag Bit Array to bits 24~31 of Material Index.

Loops Topology Data

A Loop (often called Trimming Loop) defines in parameter space a 1D boundary around which geometric surfaces are trimmed to form a Face. Loops Topology Data specifies the CoEdges making up each Loop along with an anti-hole flag and identifier tag for each Loop. The complete description of loop and its relation to the CoEdges can be found in Loops Topology Data.

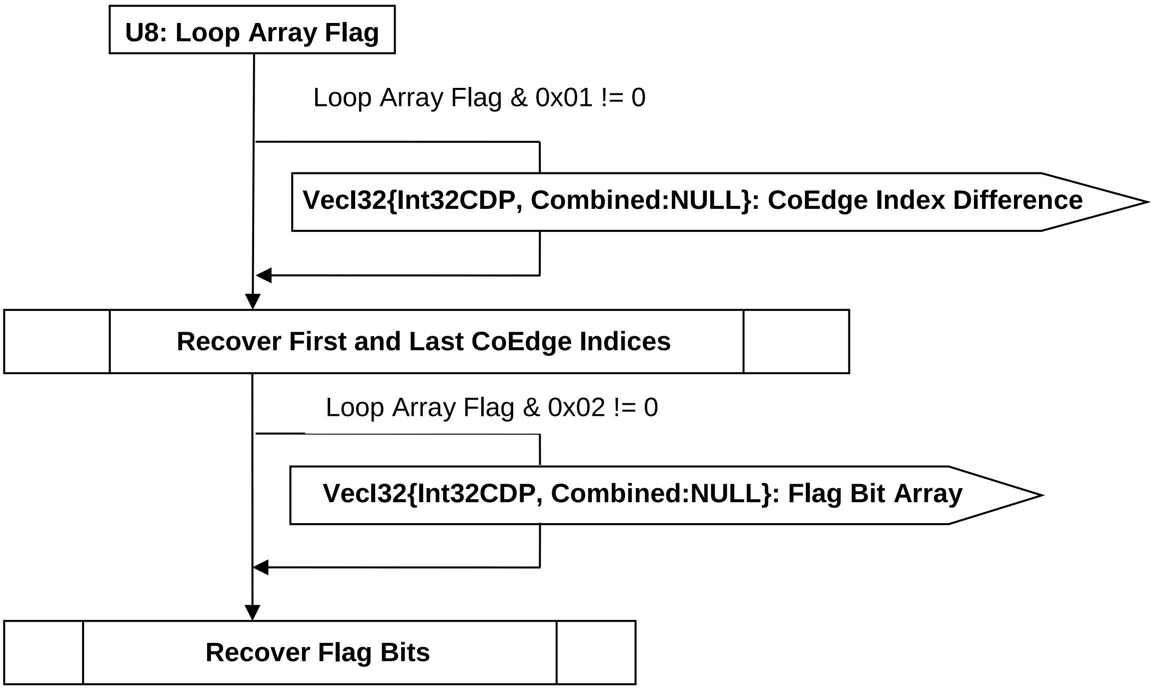


Figure 170 — Loops Topology Data collection

U8: Loop Array Flag

Loop Array Flag indicates which arrays of loop topology data are not trivial and therefore encoded.

VecI32{Int32CDP, Combined:NULL}: CoEdge Index Difference

CoEdge Index Difference is a vector of indices representing the integer value by subtracting first CoEdge index from last CoEdge index in each loop, encoded using Combined Predictor Type. CoEdge Index Difference is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover First and Last CoEdge Indices

The first CoEdge index of the first loop is 0, and the last CoEdge index of the first loop is element 0 of CoEdge Index Difference. The first CoEdge index of loop $k, k \geq 1$ equals to the last CoEdge index of loop $k-1$ plus 1. The last CoEdge index of loop $k, k \geq 1$ equals to the first CoEdge index of loop k plus element k of CoEdge Index Difference array.

VecI32{Int32CDP, Combined:NULL}: Flag Bit Array

Only the lower 24 bits of the two integer indices, namely first CoEdge index and last CoEdge index are used as integer identifiers. The other bits of these integers documented as reserved for future use.

Table 174 — JT ULP Loops Topology Flag Bit Array values

	24	25	26	27	28	29	30	31
First CoEdge Index	Reserved							isAntiHoleLoop
Last CoEdge Index	Reserved							

Bits 0~7 of Flag Bit Array are equal to bits 24~31 of First CoEdge Index

Bits 8~15 of Flag Bit Array are equal to bits 24~31 of Last CoEdge Index

Bits 16~31 of Flag Bit Array are set to be 0

In an uncompressed/decoded form, the AntiHole Loop Flag has the following meaning:

Table 175 — JT ULP Loops Topology Reverse Normal Flag values

= 0	Loop is not an anti-hole Loop
= 1	Loop is an anti-hole Loop

Recover Flag Bits

The flag bits are recovered by assigning bits 0~7 of Flag Bit Array to bits 24~31 of First CoEdge Index, and bits 8~15 of Flag Bit Array to bits 24~31 of Last CoEdge Index.

CoEdges Topology Data

A CoEdge defines a parameter space edge trim Loop segment (i.e. the projection of an Edge into the parameter space of the Face). CoEdges Topology Data specifies the underlying Edge and PCS Curve making up each CoEdge along with a MCS curve reversed flag and tag for each CoEdge. The complete description of CoEdge and its relation to the Edge can be found in CoEdges Topology Data.

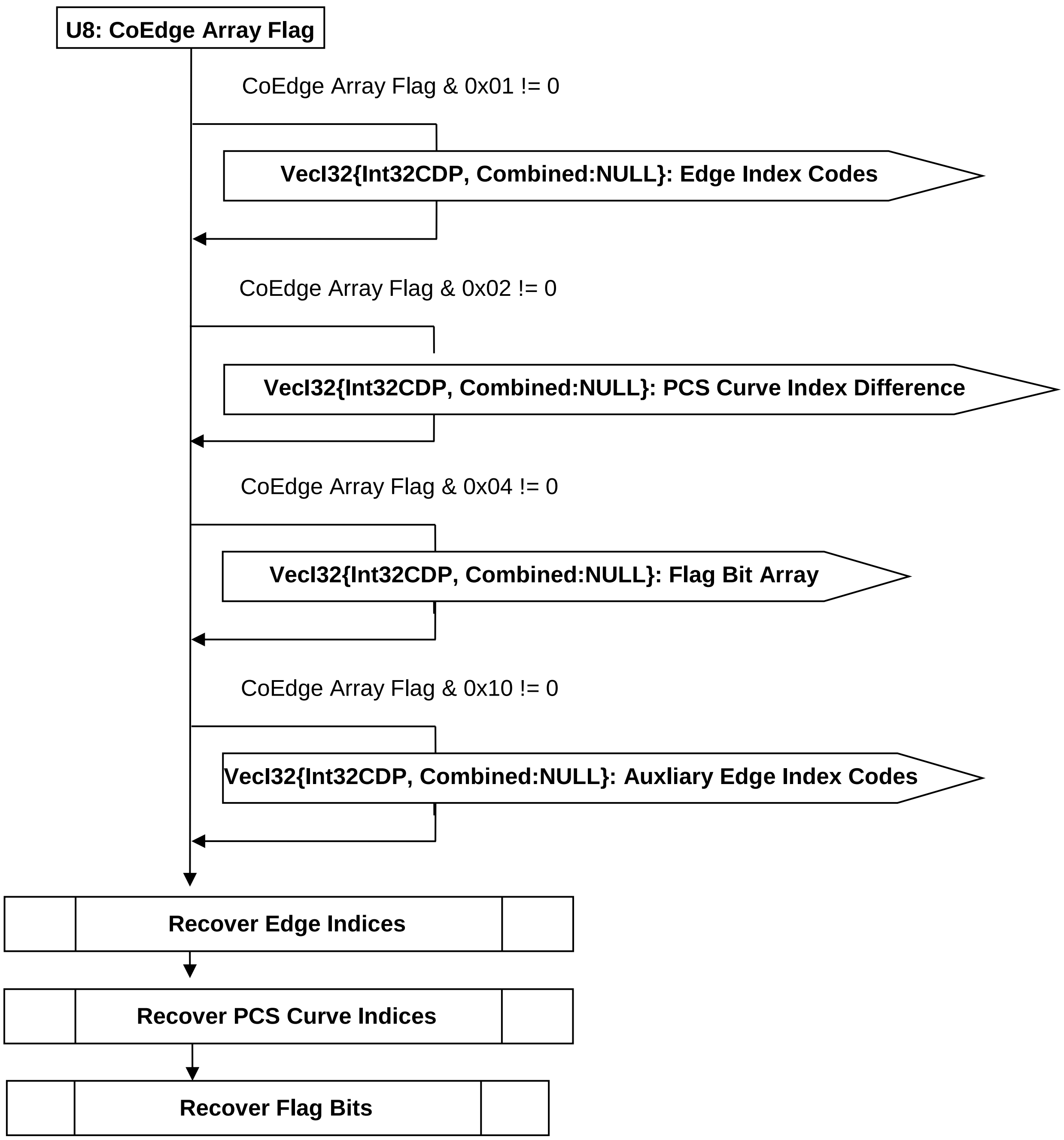


Figure 171 — CoEdges Topology Data collection

U8: CoEdge Array Flag

CoEdge Array Flag indicates which arrays of coedge topology data are not trivial and therefore encoded.

VecI32{Int32CDP, Combined:NULL}: Edge Index Codes

Edge Index Codes is a vector of integer indices representing the Edge index for each CoEdge, encoded using Combined Predictor Type. Edge Index Codes is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Depending on how the edge indices are assigned, and indicated by bit 0x08 in CoEdge Array Flag, the Edge Index Codes may represent the Edge Index information in two different ways. If the edge indices are randomly assigned, for example as shown in Figure 173, then each element in Edge Index Codes represents the integer value by subtracting the Edge index from the CoEdge index for each CoEdge. For the example shown in Figure 173, the integer values in Edge Index Codes are -7, -4, -4, 3, 3, -1, -2, 5, 5, 1, 5, 7 and bit 0x08 is turned off in CoEdge Array Flag. If the edge indices are chosen to be based on the sequence of the reference from the parent coedges when the coedges are visited sequentially, as shown in Figure 174, then each element in Edge Index Codes has value 0 if the edge is visited the first time, and value 1 if the edge is visited the second time. For the example shown in Figure 174, the interger values in Edge Index Codes are 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, and bit 0x08 is turned on in CoEdge Array Flag. Note that the edge index for all CoEdges with 0 entry can be figured out by counting number of zeros in Edge Index Codes preceding (not including) this entry. Take CoEdge 6 that has entry 0 in Edge Index Codes for example. The number of zeros before this entry is 5, which is equal to the edge index of CoEdge 6. Therefore only the edge indices of those CoEdges with entry value 1 in Edge Index Codes need be stored. These edge indices are stored in Auxliary Edge Index Codes.

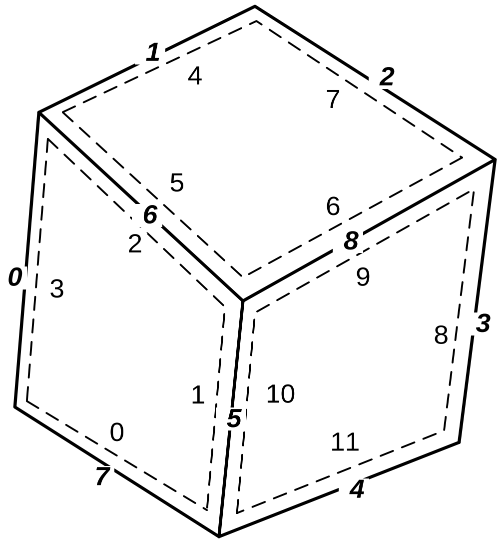


Figure 172 — Sample Model with Randomly Assigned Edge Indices

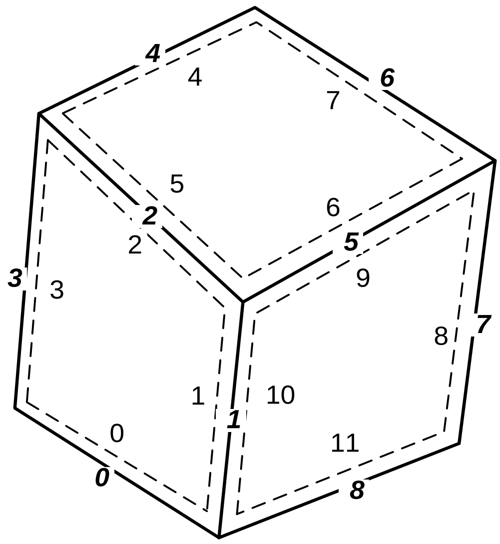


Figure 173 — Sample Model with Sequentially Assigned Edge Indices

VecI32{Int32CDP, Combined:NULL}: Auxliary Edge Index Codes

Auxliary Edge Index Codes is an optional field and only exists if CoEdge Array Flag & 0x08 is not equal to 0. It contains the Edge indices that appear again during sequential traversal of the CoEdges. For the example shown in Figure 174, the entries in Auxliary Edge Index Codes are 2, 5, 1. More detailed explanation of what these entires mean can be found in Edge Index Codes.

Recover Edge Indices

If CoEdge Array Flag & 0x08 is equal to 0, then the Edge index of each CoEdge is equal to the CoEdge index if CoEdge Array Flag & 0x01 is equal to 0. Otherwise, the Edge index of CoEdge with index k can be computed by substracting element k of Edge Index Codes array from k, the CoEdge index.

If CoEdge Array Flag & 0x08 is equal to 1, then the Edge index of all CoEdges having 0 entry in Edge Index Codes is set to be the number of zeros in Edge Index Codes preceding (not including) this entry. For all CoEdges that have 1 in Edge Index Codes, their edge indices are sequentially assigned as the corresponding value in Auxliary Edge Index Codes.

VecI32{Int32CDP, Combined:NULL}: PCS Curve Index Difference

PCS Curve Index Difference is a vector of indices representing the integer value by subtracting the PCS Curve index from the CoEdge index for each CoEdge, encoded using Combined Predictor Type. PCS Curve Index Difference is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover PCS Curve Indices

If CoEdge Array Flag & 0x02 is equal to 0, then the PCS Curve index of each CoEdge is equal to the CoEdge index. Otherwise, the PCS Curve index of CoEdge with index k can be computed by substracting element k of PCS Curve Index Difference array from k, the CoEdge index.

VecI32{Int32CDP, Combined:NULL}: Flag Bit Array

Only the lower 24 bits of the two integer indices, namely Edge index and PCS Curve index, are used as integer identifiers. The other bits of these integers are documented as reserved for future use.

Table 176 — JT ULP Recover Edge Indices Flag Bit Array values

<table>
<tr><th></th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>Edge Index</td><td colspan="2">Knot Type</td><td colspan="3">Domain Type</td><td colspan="2">PCS Curve Type</td><td>isXYZReversed</td></tr>
<tr><td>PCS Curve Index</td><td>isUvInc</td><td colspan="7">Reserved</td></tr>
</table>

Bits 0~7 of Flag Bit Array are equal to bits 24~31 of Edge Index

Bits 8~15 of Flag Bit Array are equal to bits 24~31 of PCS Curve Index

Bits 16~31 of Flag Bit Array are set to be 0

The Knot Type, defined in Supported Knot Type, is an integer with its value between 0 and 3.

Domain Type

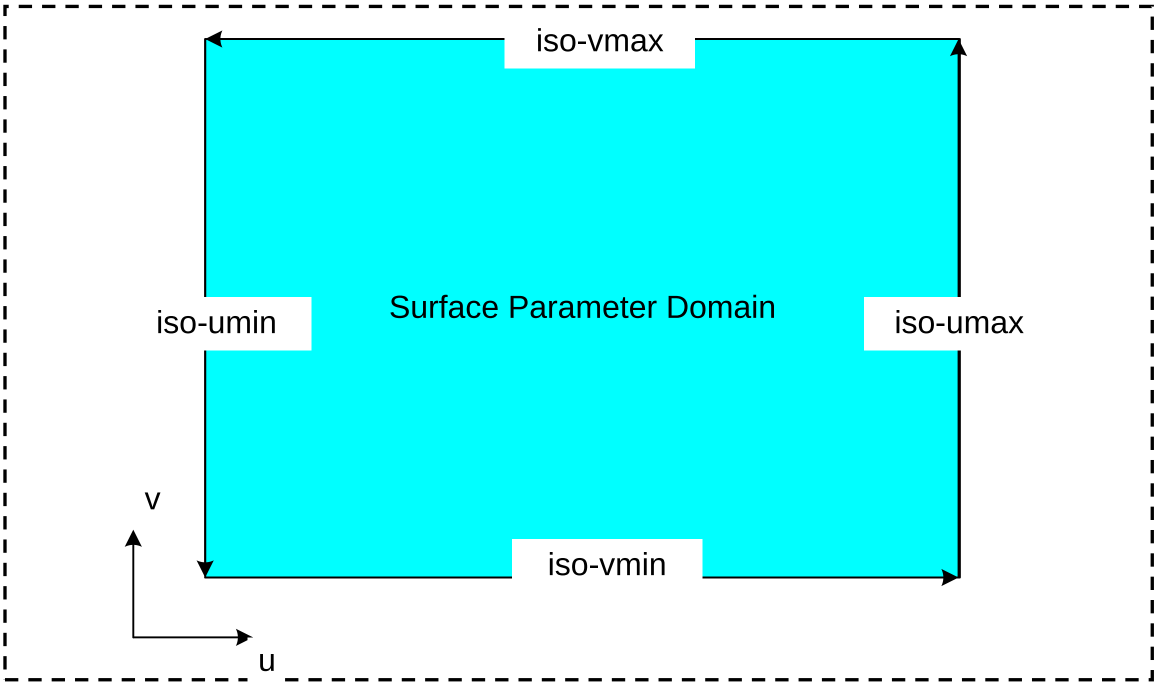


Figure 174 — Surface Domain Classification

In an uncompressed/decoded form, the supported PCS Curve types are listed below.

Table 177 — JT ULP Recover Edge Indices PCS curve type values

0	General
1	PCS curve is coincident with iso-umin curve in the surface parameter domain
2	PCS curve is coincident with iso-umax curve in the surface parameter domain
3	PCS curve is coincident with iso-vmin curve in the surface parameter domain
4	PCS curve is coincident with iso-vmax curve in the surface parameter domain
5	Reserved
6	Reserved
7	PCS curve is to be derived from MCS curve and surface geometry

PCS Curve Type

In an uncompressed/decoded form, the supported PCS Curve types are listed below.

Table 178 — JT ULP PCS Curve Type values

0	Nurbs
1	Line
2	Circle
3	Reserved

In an uncompressed/decoded form, the XYZReversed Flag has the following meaning:

Table 179 — JT ULP PCS Curve Type XYZ Reversed Flag values

= 0	Directional sense of associated edges MCS curve should not be interpreted as opposite the direction its parameterization implies.
= 1	Directional sense of associated edges MCS curve should be interpreted as opposite the direction its parameterization implies.

In an uncompressed/decoded form, the isUVInc Flag has the following meaning:

Table 180 — JT ULP PCS Curve Type isUVInc Flag values

= 0	PCS Curve is iso-parameteric in surface parameter domain in one direction and the parameter increases in the other direction
= 1	PCS Curve is iso-parameteric in surface parameter domain in one direction and the parameter decreases in the other direction

The isUVInc flag is set only if the Domain Type of this CoEdge has value between 1 and 4 inclusive.

Recover Flag Bits

If CoEdge Array Flag & 0x04 is equal to 0, then each element in Flag Bit Array is set to have value 0. The flag bits are recovered by assigning bits 0~7 of Flag Bit Array to bits 24~31 of Edge Index, and bits 8~15 of Flag Bit Array to bits 24~31 of PCS Curve Index.

Edges Topology Data

An Edge defines a model space trim Loop segment. Edges Topology Data specifies the underlying MCS Curve along with an identification tag for each Edge. The complete description of Edge can be found in Edges Topology Data. Note that the start and end vertex index information is not stored. Instead it is recovered (Information Recovery).

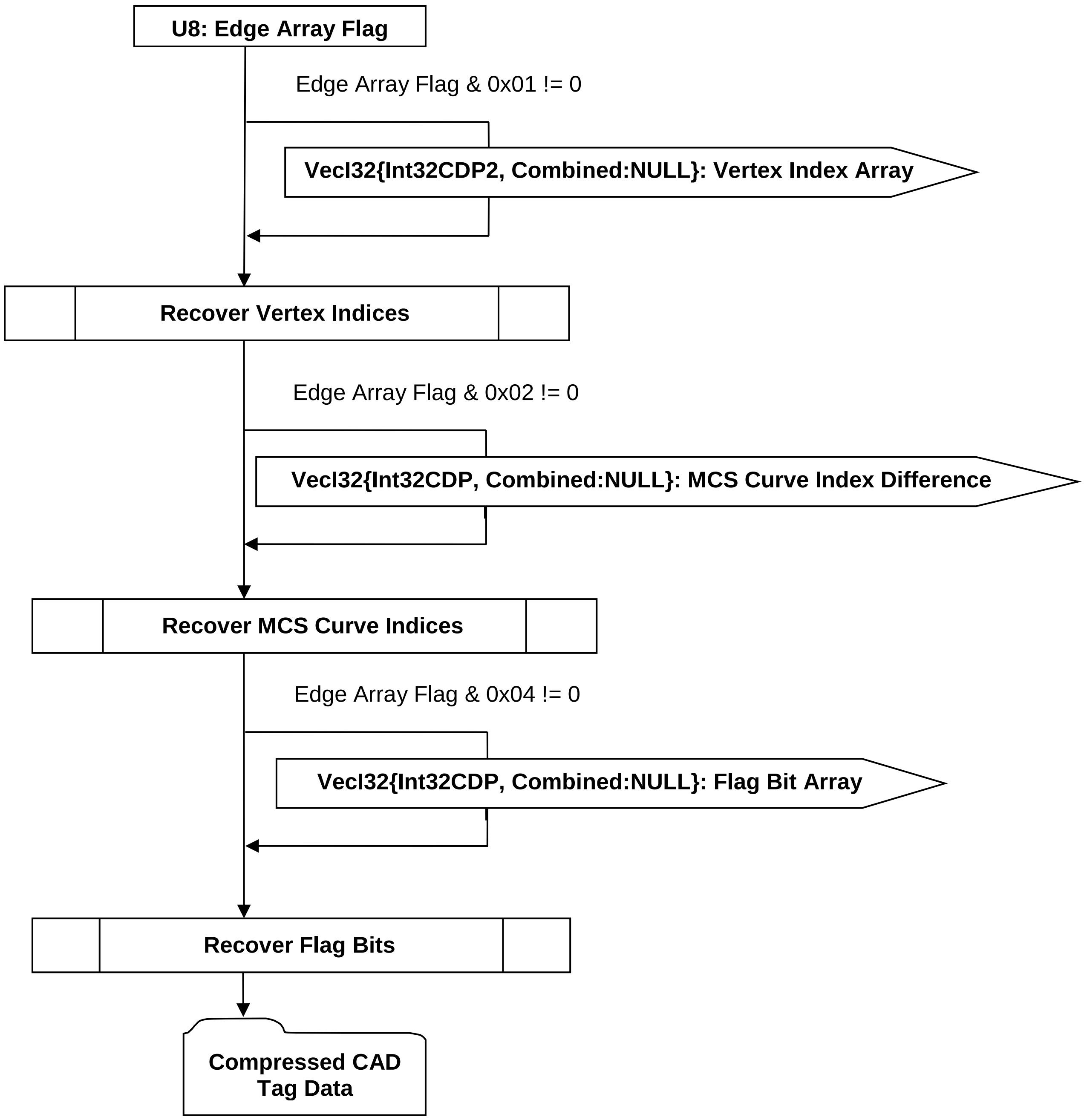


Figure 175 — Edges Topology Data collection

U8: Edge Array Flag

Edge Array Flag indicates which arrays of edge topology data are not trivial and therefore encoded.

VecI32{Int32CDP, Combined:NULL}: MCS Curve Index Difference

MCS Curve Index Difference is a vector of indices representing the integer value by subtracting the MCS Curve index from the Edge index for each Edge, encoded using Combined Predictor Type. MCS Curve Index Difference is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover MCS Curve Indices

If Edge Array Flag & 0x02 is equal to 0, then the MCS Curve index of each Edge is equal to the Edge index. Otherwise, the MCS Curve index of Edge with index k can be computed by substracting element k of MCS Curve Index Difference array from k, the Edge index.

VecI32{Int32CDP, Combined:NULL}: Flag Bit Array

Only the lower 24 bits of the three integer indices, namely MCS Curve index, Start Vertex index, and End Vertex index, are used as integer identifiers. The other bits of these integers documented as reserved for future use.

Table 181 — JT ULP Edges Topology Recover MCS Curve Indices Flag Bit Array values

	24	25	26	27	28	29	30	31
MCS Curve Index	Knot Type		MCS Curve Type		Reserved			
Start Vertex Index	Reserved							
End Vertex Index	Reserved							

The Knot Type, defined in Supported Knot Type, is an integer with its value between 0 and 3.

MCS Curve Type

In an uncompressed/decoded form, the supported MCS Curve types are listed below.

Table 182 — JT ULP Edges Topology Recover MCS Curve Type values

0	Nurbs
1	Line
2	Circle
3	Projection: MCS curve geometry is to be computed from surface geometry and/or PCS curve geometry

Recover Flag Bits

If Edge Array Flag & 0x04 is equal to 0, then each element in Flag Bit Array is set to have value 0. The flag bits are recovered by assigning bits 0~7 of Flag Bit Array to bits 24~31 of MCS Curve Index.

Vertices Topology Data

Vertices Topology Data is not stored on disk. Instead it is constructed (Information Recovery).

G.1.2 Geometric Data

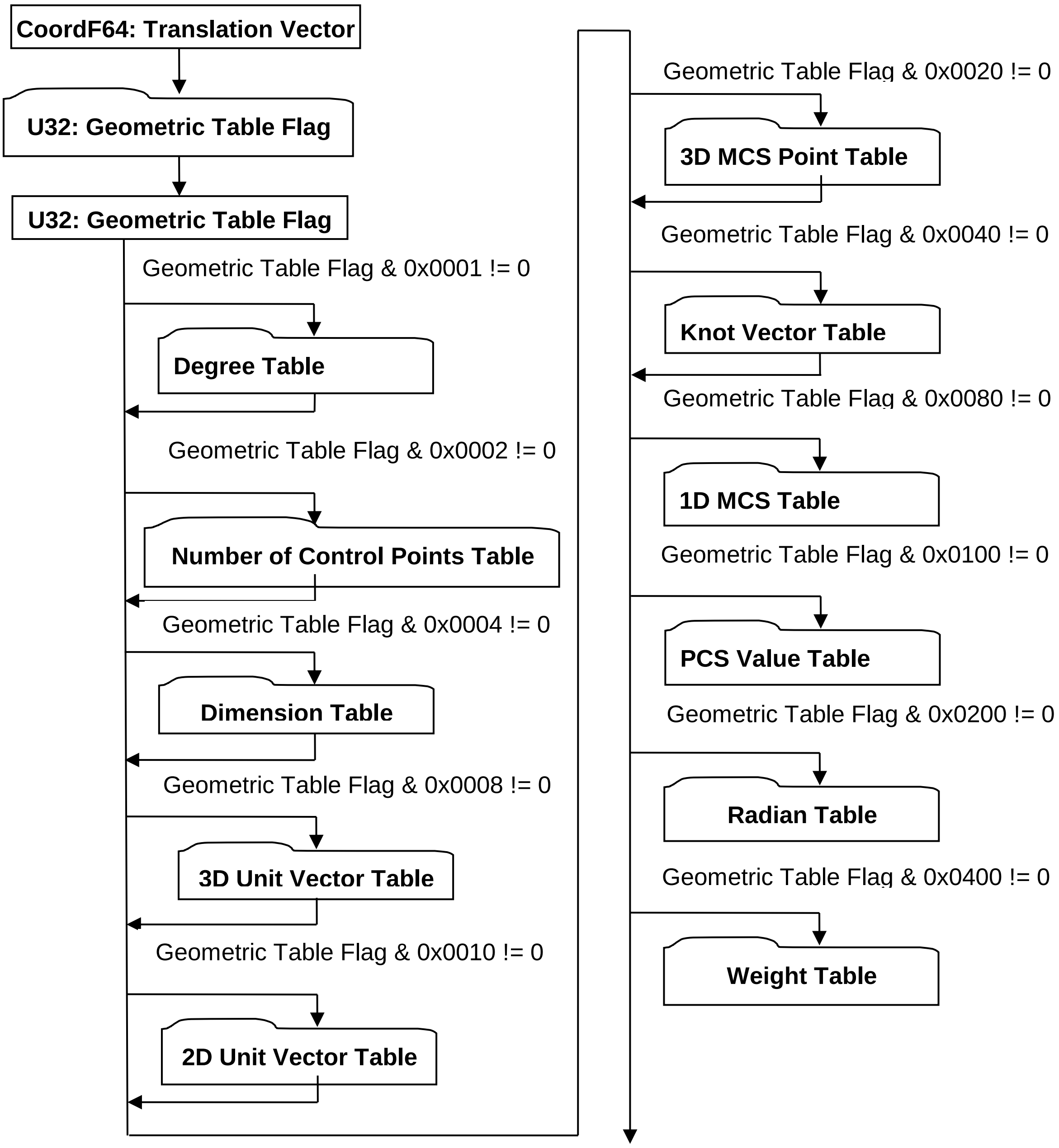


Figure 176 — Geometric Data collection

CoordF64: Translation Vector

Translation Vector is a 3-dimensional vector that represents how the ULP geometry is defined w.r.t. the original B-Rep definition from which ULP geometry is derived. If the Translation Vector is not zero vector, then the ULP geometry read from disk is translated from original B-Rep definition by the amount of Translation Vector. This is usually done by the JT writer implementation to improve numerical accuracy of floating point numbers in the ULP geometry. It is important for all the JT readers to take this Translation Vector into consideration when consuming ULP geometry. For example if a LOD is generated from ULP geometry, e.g. by tessellation, then the LOD geometry shall be translated to undo the effect of Translation Vector for it be consistent with the original B-Rep definition. In other words, if we denote the Translation Vector as v, then the LOD geometry from ULP shall be translated by – v.

U32: Geometric Table Flag

Geometric Table Flag indicates which geometric tables are not trivial and therefore encoded.

Geometric Entity Counts

Geometric Entity Counts data collection defines the counts for each of the various geometric entities within a ULP.

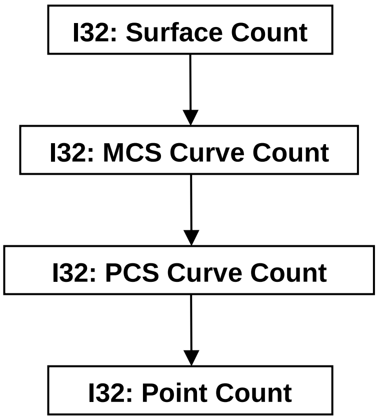


Figure 177 — Geometric Entity Counts

I32: Surface Count

Surface Count indicates the number of distinct geometric surface entities in the ULP.

I32: MCS Curve Count

MCS Curve Count indicates the number of distinct geometric (Model Coordinate Space) curves (i.e. XYZ curve) entities in the ULP.

I32: PCS Curve Count

PCS Curve Count indicates the number of distinct geometric Parameter Coordinate Space curves (i.e. UV curve) entities in the ULP.

I32: Point Count

Point Count indicates the number of distinct geometric point entities in the ULP.

Degree Table

Degree Table stores a vector of integers that represent the degree information of Nurbs surfaces and/or curves. If the ULP does not contain any Nurbs entity, then the table is empty and bit 0x0001 in Geometric Table Flag is set to be 0.

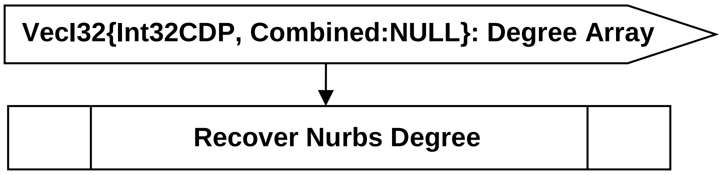


Figure 178 — Degree Table data collection

VecI32{Int32CDP, Combined:NULL}: Degree Array

Degree Array is a vector of integers that stores the degree information for all the Nurbs entities in the ULP, encoded using Combined Predictor Type. Degree Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover Nurbs Degree

The logic diagram to recover degree information for all the Nurbs entities in the ULP from the Degree Array is shown below.

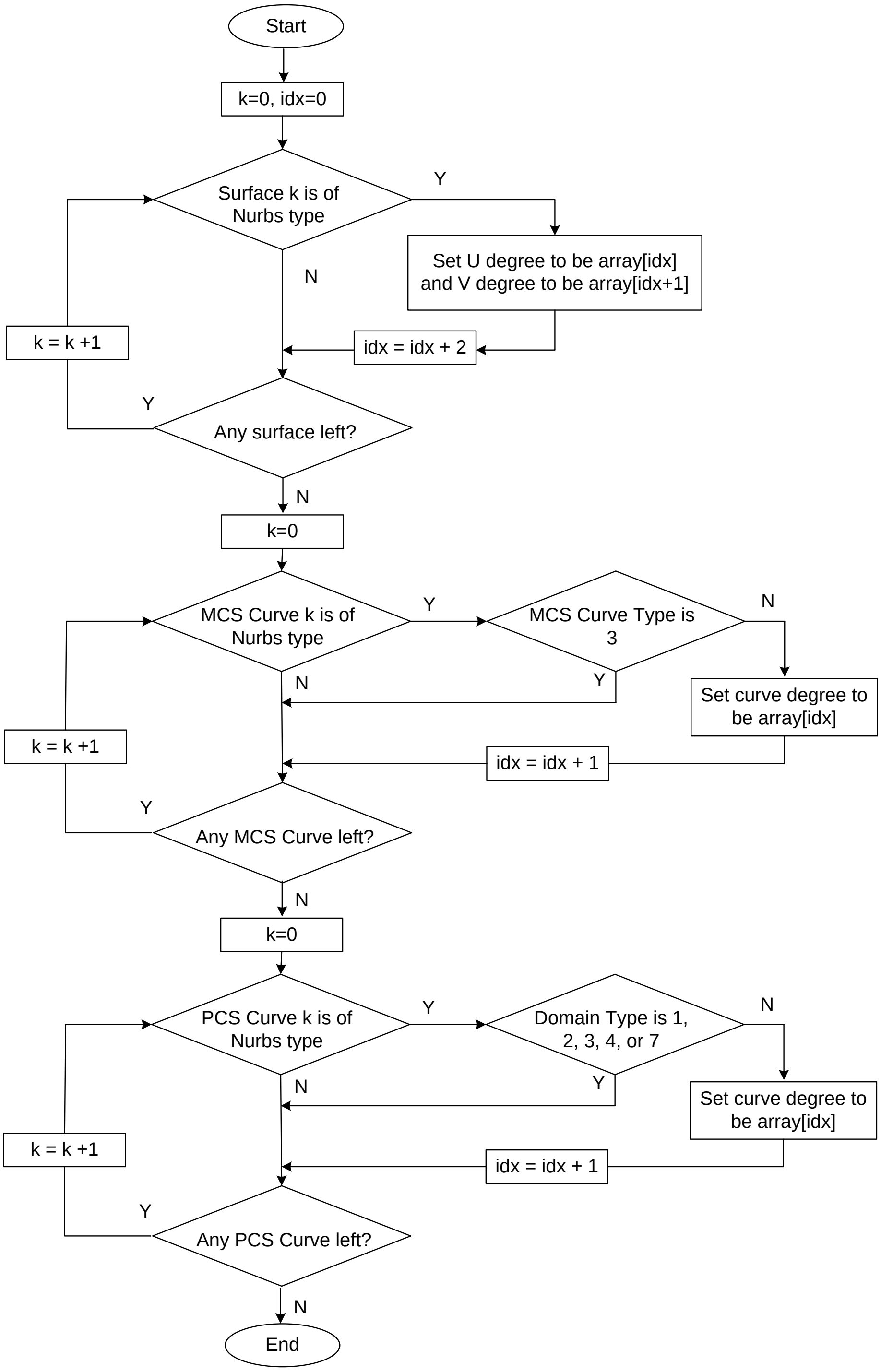


Figure 179 — Recover Nurbs Degree

Number of Control Points Table

Number of Control Points Table stores a vector of integers that represent the number of control points information of Nurbs surfaces and/or curves. If the ULP does not contain any Nurbs entity, then the table is empty and bit 0x0002 in Geometric Table Flag is set to be 0.

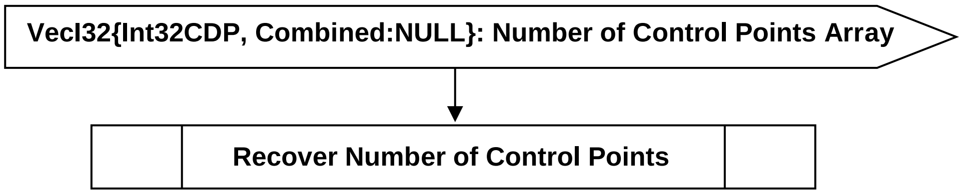


Figure 180 — Number of Control Points Table data collection

VecI32{Int32CDP, Combined:NULL}: Number of Control Points Array

Number of Control Points Array is a vector of integers that stores the number of control points information for all the Nurbs entities in the ULP, encoded using Combined Predictor Type. Number of Control Points Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover Number of Control Points

The logic diagram to recover number of control points information for all the Nurbs entities in the ULP from the Number of Control Points Array is shown below.

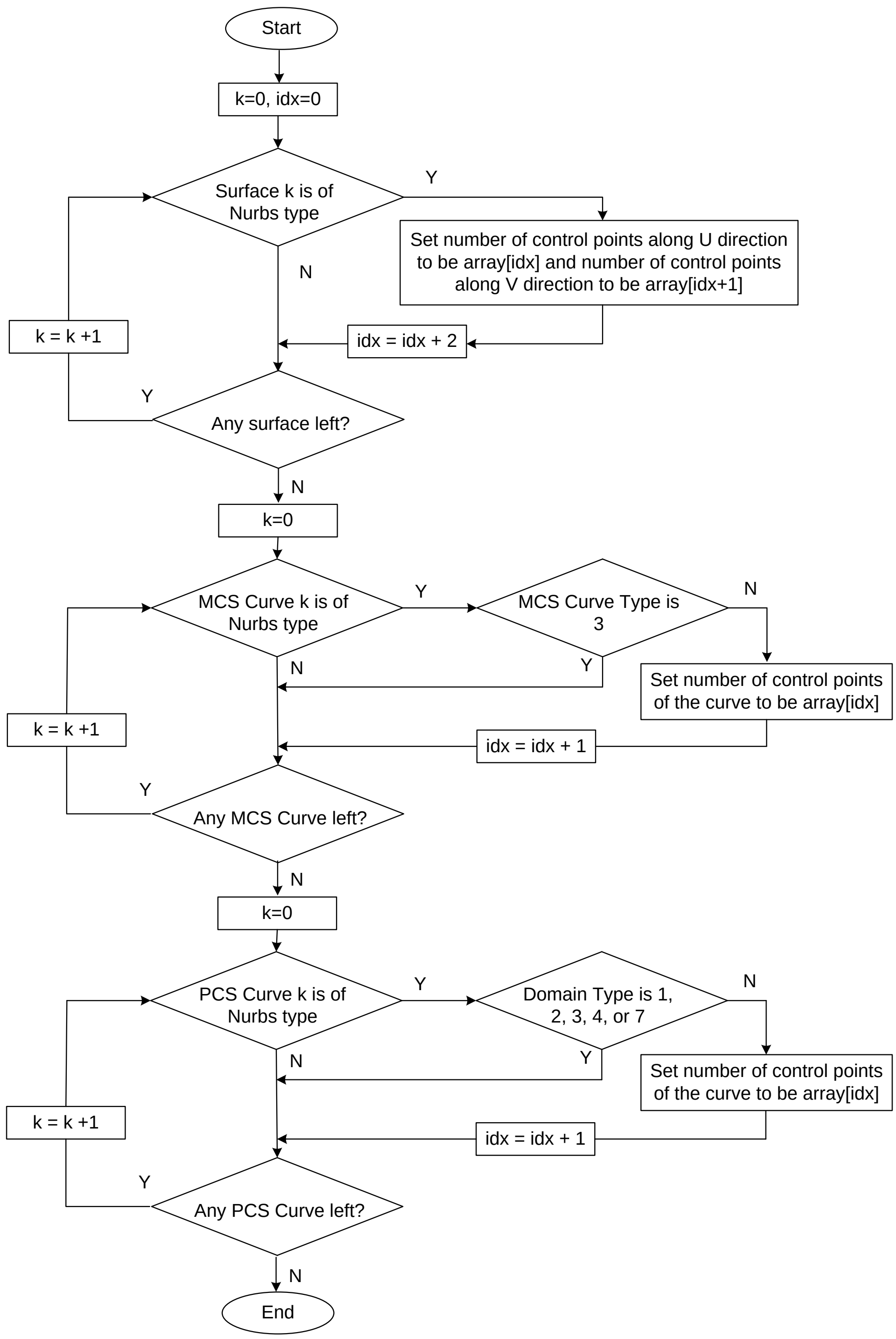


Figure 181 — Recover Number of Control Points

Dimension Table

Dimension Table stores a vector of integers that represent the dimension information of Nurbs surfaces and/or curves. If the ULP does not contain any Nurbs entity, then the table is empty and bit 0x0004 in Geometric Table Flag is set to be 0.

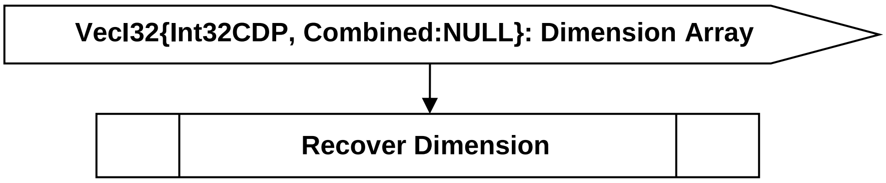


Figure 182 — Dimension Table data collection

VecI32{Int32CDP, Combined:NULL}: Dimension Array

Dimension Array is a vector of integers that stores the dimension information for all the Nurbs entities in the ULP, encoded using Combined Predictor Type. Dimension Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover Dimension

The logic diagram to recover dimension information for all the Nurbs entities in the ULP from the Dimension Array is shown below.

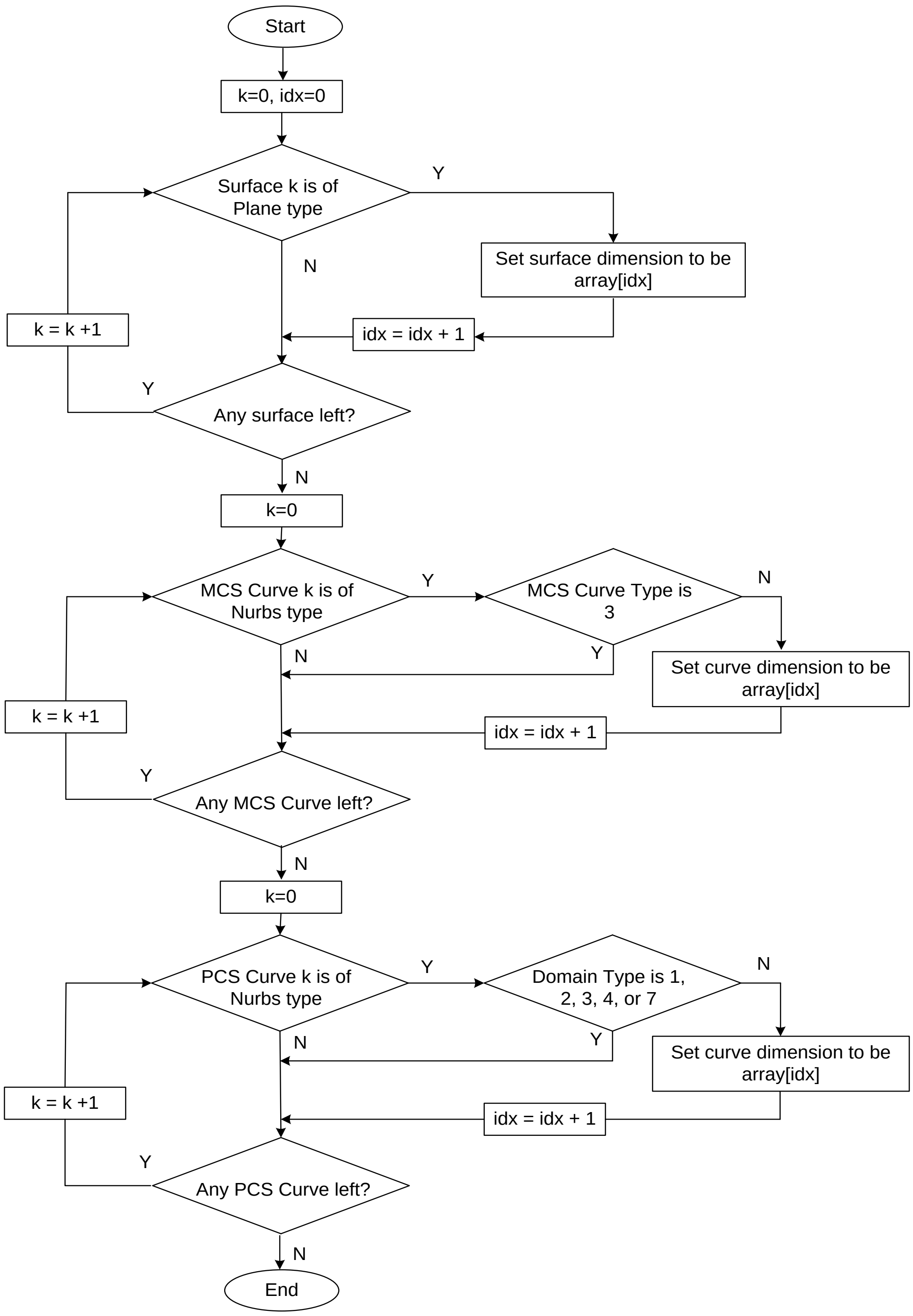


Figure 183 — Recover Dimension

3D Unit Vector Table

3D Unit Vector Table stores an array of unit vectors in 3D that form part of the analytic surface or curve representation in ULP. If the ULP does not contain any analytic entity, then the table is empty and bit 0x0008 in Geometric Table Flag is set to be 0. The supported analytic surface types include plane, cylinder, cone, sphere, and torus, and the supported analytic curve types include line and circle for both parameter space and model space curves. The analytic representation of ULP follows XT convention as detailed in the XT Annex description.

Similar to the coding of Compressed Vertex Normal Array, each 3D unit vector is encoded as a single 32 bit integer using Deering Normal CODEC.

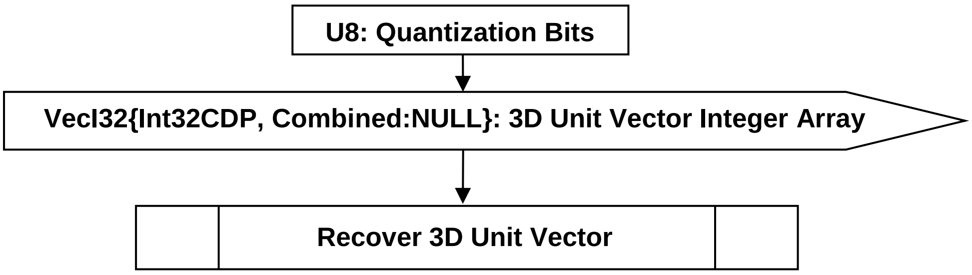


Figure 184 — 3D Unit Vector Table data collection

U8: Quantization Bits

The number of bits used for the Deering Normal CODEC if quantization is enabled. A value of 0 denotes that quantization is disabled.

VecI32{Int32CDP, Combined:NULL}: 3D Unit Vector Integer Array

3D Unit Vector Integer Array is a vector of integers that stores the encoded 3D unit vector from all analytic entities in the ULP, encoded using Combined Predictor Type. 3D Unit Vector Integer Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Recover 3D Unit Vector

The logic diagram to recover 3D unit vector information for all the analytic entities in the ULP from the 3D Unit Vector Integer Array is shown below.

The recovery of a unit vector from an element in the 3D Unit Vector Integer Array is done as part of Deering Normal CODEC.

As described in the XT B-Rep Annex, the representation of an analytic surface of types plane, cylinder, cone, sphere, or torus, includes two 3D unit vectors. One is called “axis” and the other is called “x_axis”. These two unit vectors of each analytic surface are recovered for each analytic surface. In addition, the “normal” vector to the plane containing a 3D circle is also recovered.

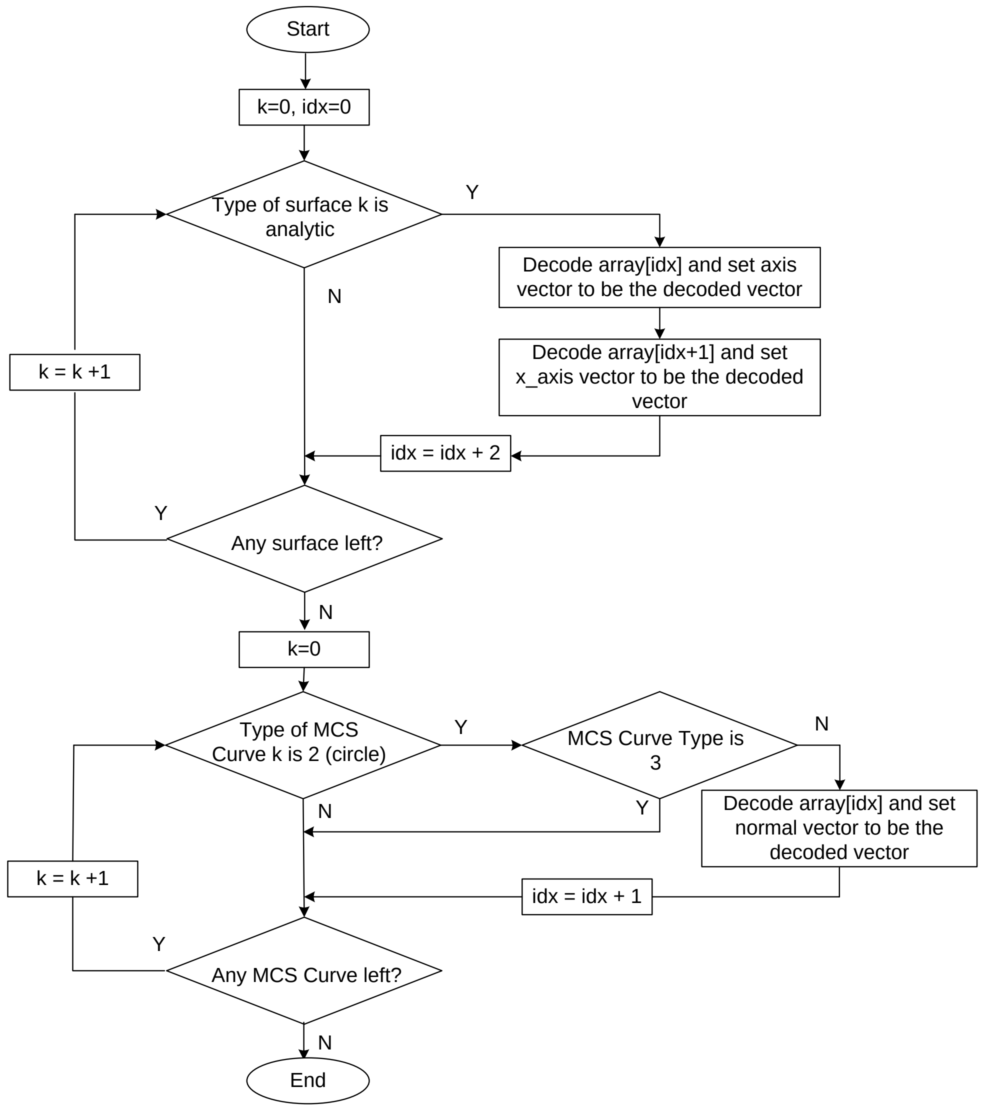


Figure 185 — Recover Dimension

2D Unit Vector Table

2D Unit Vector Table stores an array of unit vectors in 2D that form part of PCS analytic circle representation in ULP. If the ULP does not contain any analytic circle in the PCS, then the table is empty and bit 0x0010 in Geometric Table Flag is set to be 0. The analytic curve representation of ULP follows XT convention as detailed in Annex E.

Similar to the coding of Compressed Vertex Normal Array, each 2D unit vector is treated as a 3D unit vector with z component set to be 0.0, and encoded as a single 32 bit integer using Deering Normal CODEC. In addition, the Quantization Bits information of Deering Normal CODEC used to encode 2D Unit Vector Table is always the same as the one used for 3D Unit Vector Table.

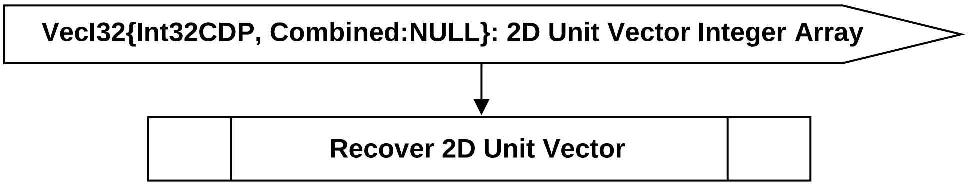


Figure 186 — 2D Unit Vector Table data collection

VecI32{Int32CDP, Combined:NULL}: 2D Unit Vector Integer Array

2D Unit Vector Integer Array is a vector of integers that stores the encoded 2D unit vector from all analytic entities in the ULP.

Recover 2D Unit Vector

The logic diagram to recover 2D unit vector information for all the analytic entities in the ULP from the 2D Unit Vector Integer Array is shown below.

The recovery of a unit vector from an element in the 2D Unit Vector Integer Array is done as part of Deering Normal CODEC.The Quantization Bits read from 3D Unit Vector Table should be used for Deering Normal CODEC to decode the vector information from each element in 2D Unit Vector Integer Array.

The “x_axis” vector to the circle in the PCS, as described in the XT B-Rep Annex, is recovered.

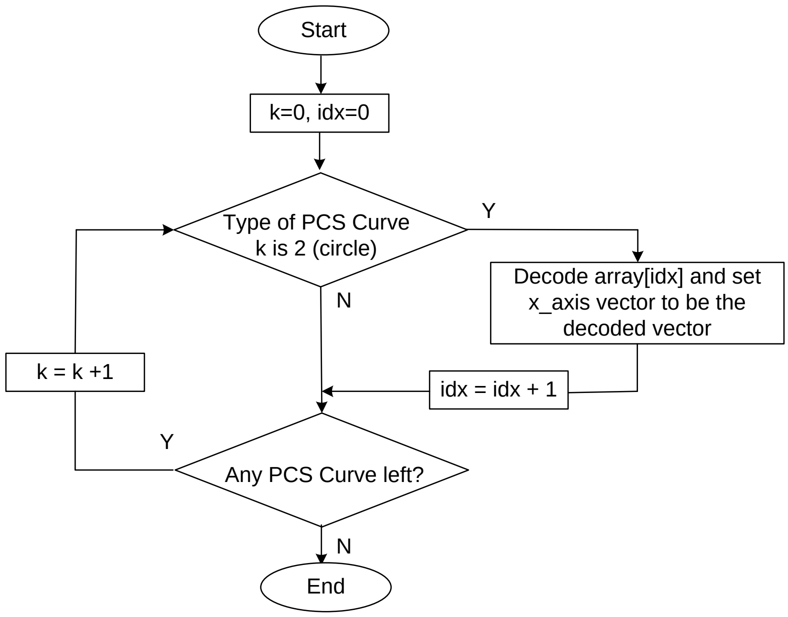


Figure 187 — Recover 2D Unit Vector

3D MCS Point Table

3D MCS Point Table stores the quantization representation of an array of 3D MCS points in ULP. If the ULP does not contain 3D MCS points, then the table is empty and bit 0x0020 in Geometric Table Flag is set to be 0.

Each point coordinate is first encoded into an integer with uniform quantizer (see Uniform Quantizer Data) and then all the integers from each coordinate are grouped into an integer array, which is then encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type.

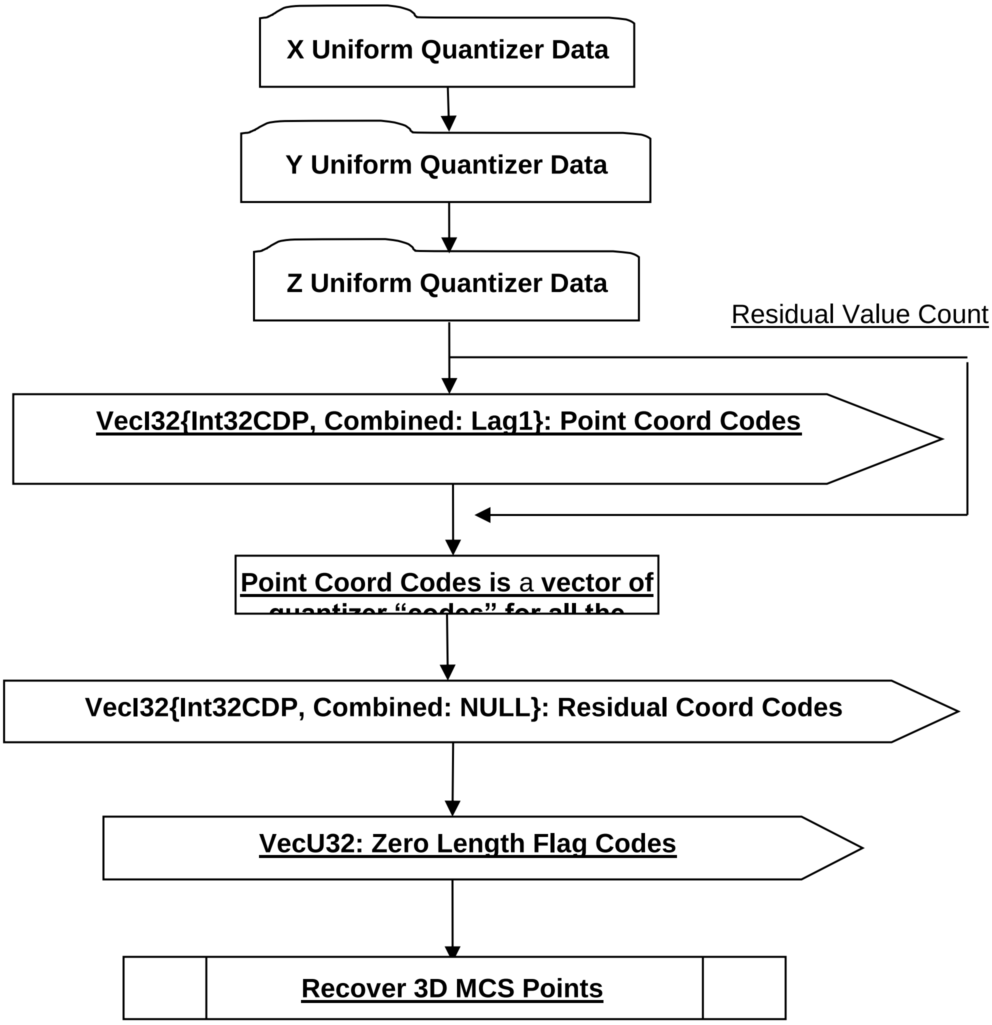


Figure 188 — 3D MCS Point Table data collection

VecI32{Int32CDP, Combined: Lag1}: Point Coord Codes

Point Coord Codes is a vector of quantizer "codes" for all the coordinate components of an array of three dimensional points, arranged in the order of X-component of point 1, Y-component of point1, Z-component of point1, X-component of point2, etc.. Point Coord Codes uses the Int32CDP CODEC described in Int32 Compressed Data Packet to compress and encode data.

I32: Residual Value Count

Residual Value Count indicates the number of residual values.

VecI32{Int32CDP, Combined: NULL}: Residual Coord Codes

Residual Coord Codes is a vector of quantizer "codes" for all the coordinate components of an array of 3-dimensional residual points, arranged in the order of X-component of residual point 1, Y-component of residual point1, Z-component of residual point1, X-component of residual point2, etc.. The residual points are computed based on Parallelogram rule for the control points of Nurbs surfaces.

Denote $P_{i,i}, P_{i+1,i}, P_{i,i+1}, P_{i+1,i+1}$ as four control points of a Nurbs surface, and $Q_{i,i}, Q_{i+1,i}, Q_{i,i+1}$ as quantized points of $P_{i,i}, P_{i+1,i}, P_{i,i+1}$ respectively, the residual point of $P_{i+1,i+1}$ is defined as $R_{i+1,i+1} = P_{i+1,i+1} + Q_{i,i} - Q_{i,i+1} - Q_{i+1,i}$.

Residual Coord Codes uses the Int32CDP CODEC described in Int32 Compressed Data Packet to compress and encode data.

VecU32: Zero Length Flag Codes

Zero Length Flag Codes is a vector of 32 bit unsigned integers, with each bit indicating whether or not a MCS curve with line geometry has zero length. The bits are arranged the same sequence as the MCS curve array. After decoding, the first N bits, where N is the total number of MCS line curves in the ULP, can be assigned to an integer array of length N with its element assigned with value 0 or 1. Each element in the decoded integer array describes whether or not the corresponding MCS line curve has zero length.

Recover 3D MCS Points

The logic diagram to recover 3D MCS points information in the ULP from the three decoded arrays, point coordinate array Pcc (with index ip) decoded from Zero Length Flag Codes, residual coordinate array Rcc (with index ir) decoded from Residual Coord Codes, and zero length flag array Zlf (with index iz) decoded from Zero Length Flag Codes, is shown below. Note that the point coordinates are decoded from the integer elements with Uniform Quantizer (see Uniform Quantizer Data).

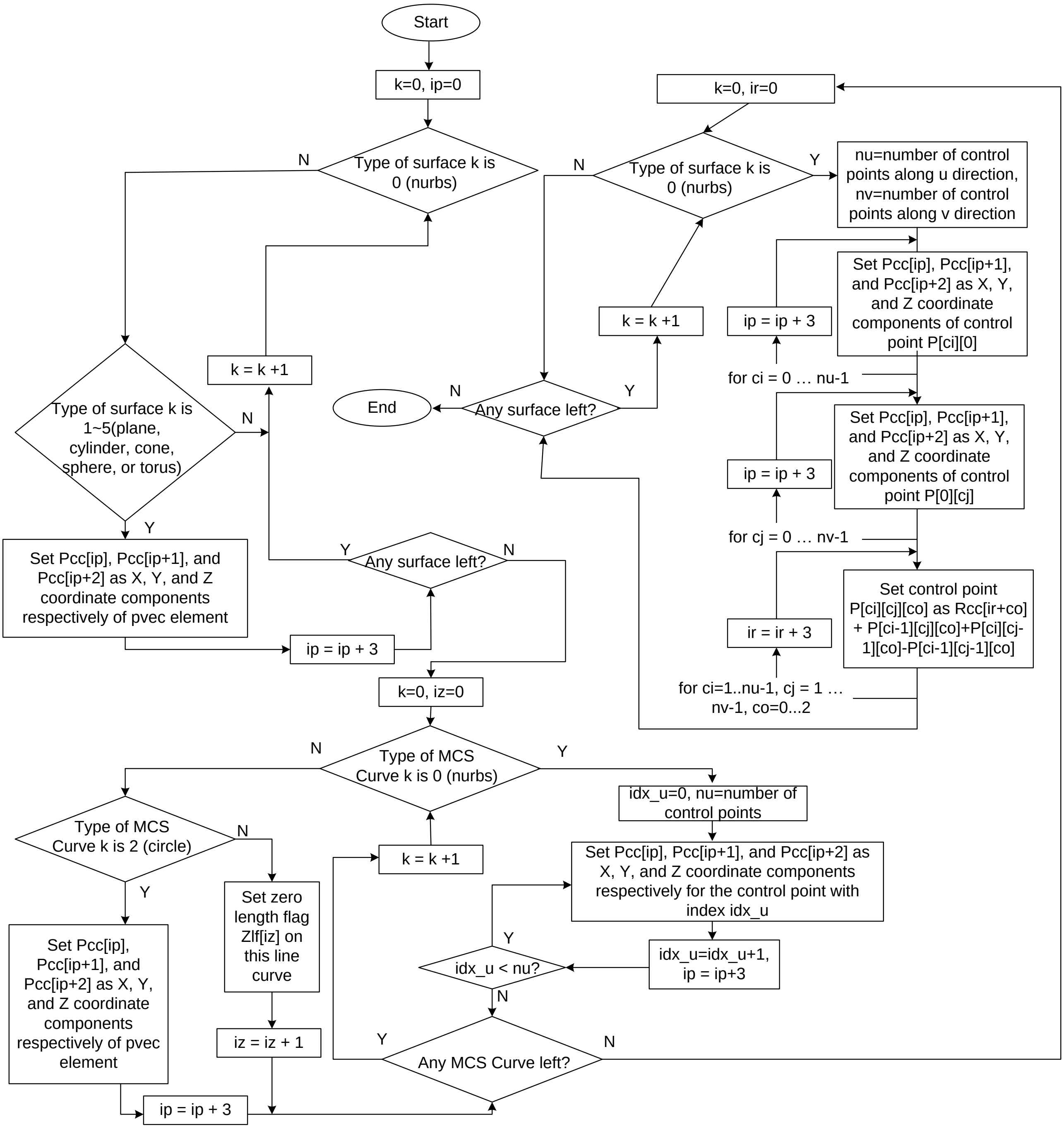


Figure 189 — Recover 3D MCS Points

Knot Vector Table

Knot Vector Table stores the quantization representation of knot vectors in ULP. If the ULP does not contain any knot vector that needs be stored, then the table is empty and bit 0x0040 in Geometric Table Flag is set to be 0.

In ULP every knot vector starts with 0.0 and ends with 1.0 and is always clamped at both ends. The encoding of knot vector depends on its classified knot type. The knot values in the middle of a knot vector need be written only if the knot type is 0 (see Supported Knot Type). For all the knot values that need be written, each of them is encoded into an integer with uniform quantizer (see Uniform Quantizer Data) and then all the integers are grouped into an integer array. The integer array is then encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type.

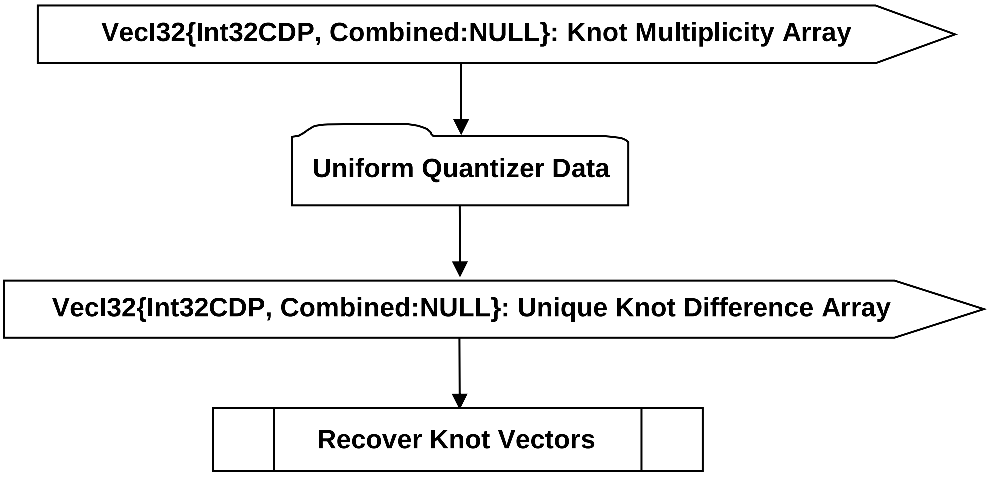


Figure 190 — Knot Vector Table data collection

VecI32{Int32CDP, Combined:NULL}: Knot Multiplicity Array

Knot Multiplicity Array is a vector of integers that describes knot multiplicity for all the knot vectors. The value of knot multiplity is set to be 0 if the knot value does not repeat itself. Knot Multiplicity Array is parallel to Knot Multiplicity Array with the same length, and uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

VecI32{Int32CDP, Combined:NULL}: Unique Knot Difference Array

Unique Knot Difference Array is a vector that represents the unique knot values. The first element has the value of the first unique knot value. Each subsequent element k represents the value difference between unique knot value k and the quantized value of unique knot value $k-1$. Unique Knot Difference Array is first quantized (Uniform Data Quantization) and then the quantized value uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

Recover Knot Vectors

The logic diagram to recover knot vector information in the ULP from the Knot Multiplicity Array (array Km) and Unique Knot Difference Array (array Uk) is shown below. Note that each integer element in the Unique Knot Difference Array is decoded with Uniform Quantizer.

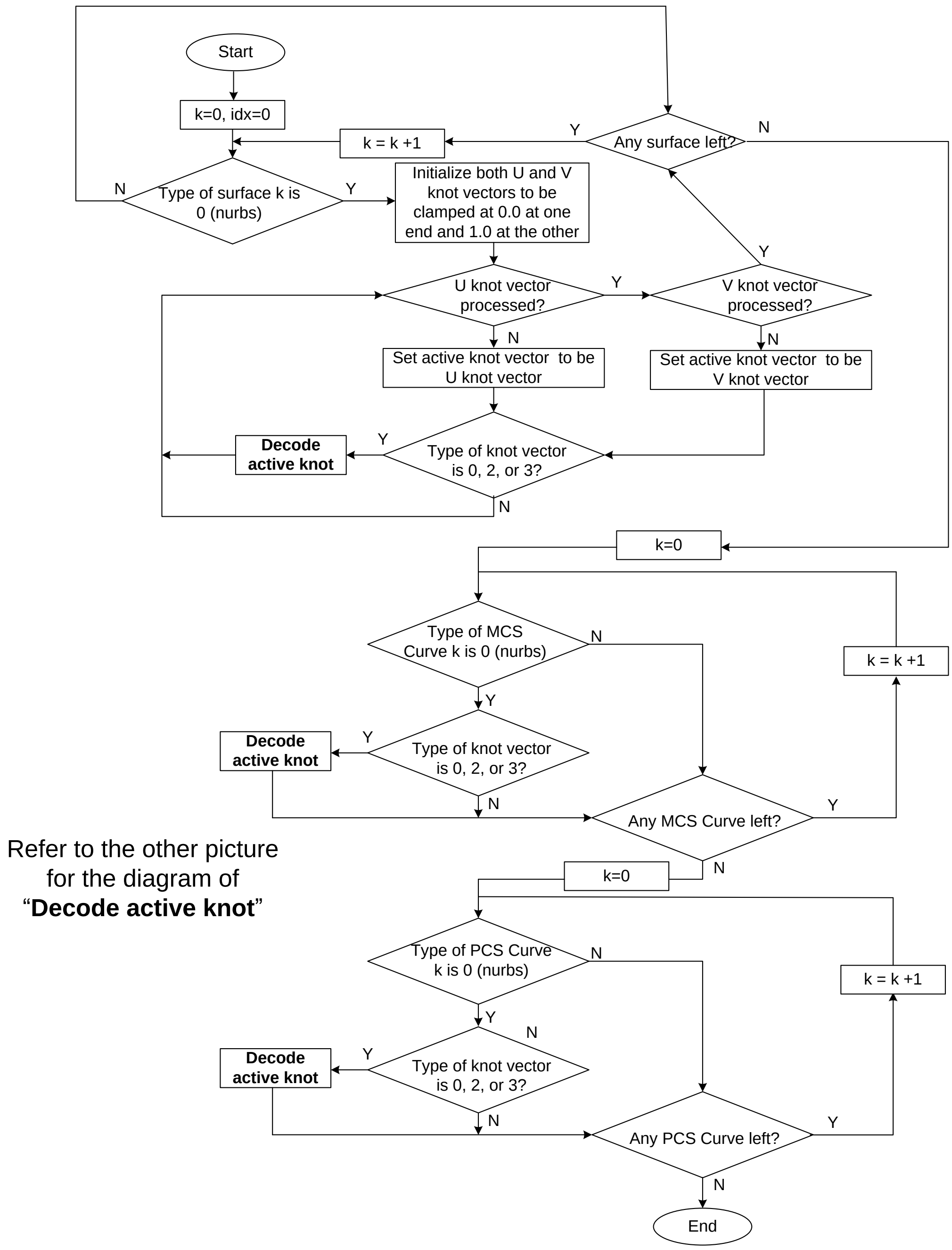
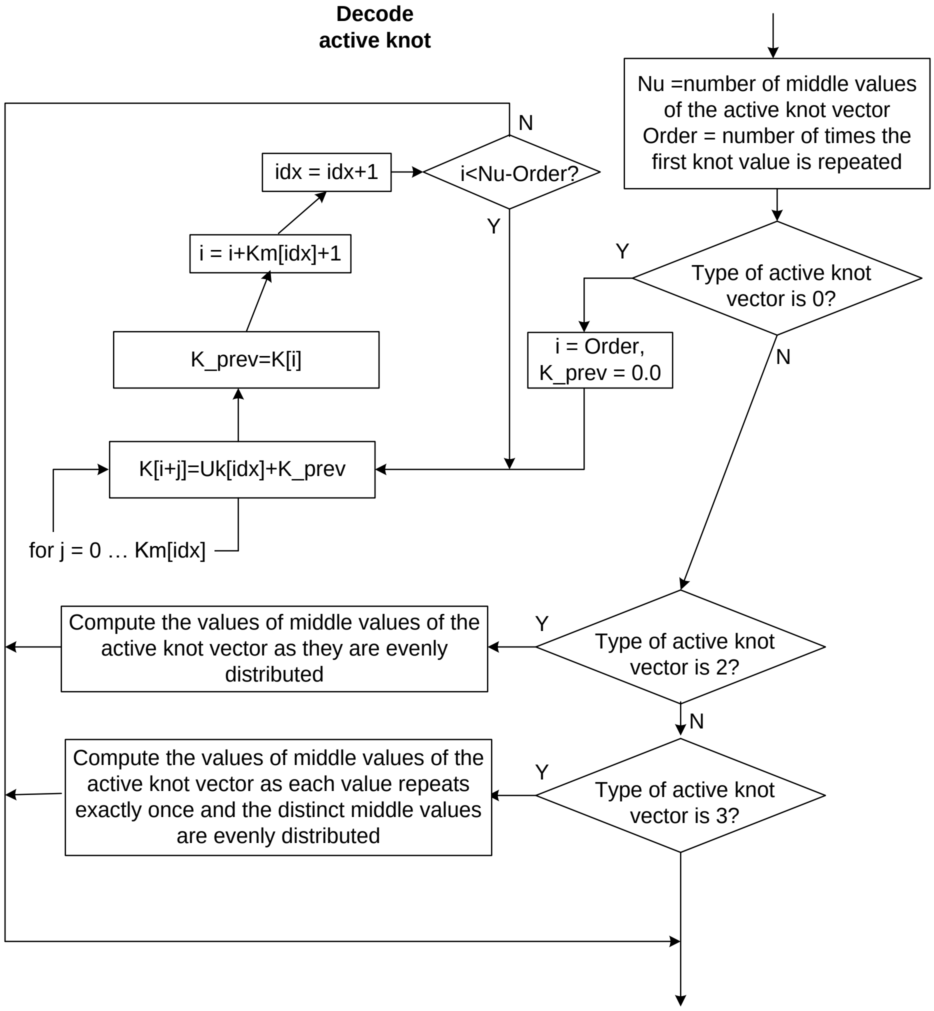


Figure 191 — Recover Knot Vectors



Recover Knot Vectors (continued)

1D MCS Table

1D MCS Table stores the quantization representation of floating point values in MCS. If the ULP does not contain any such value, then the table is empty and bit 0x0080 in Geometric Tabe Flag is set to be 0. Each floating point value is encoded into an integer with uniform quantizer (see Uniform Quantizer Data) and then all the integers are grouped into an integer array. The integer array is then encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type.

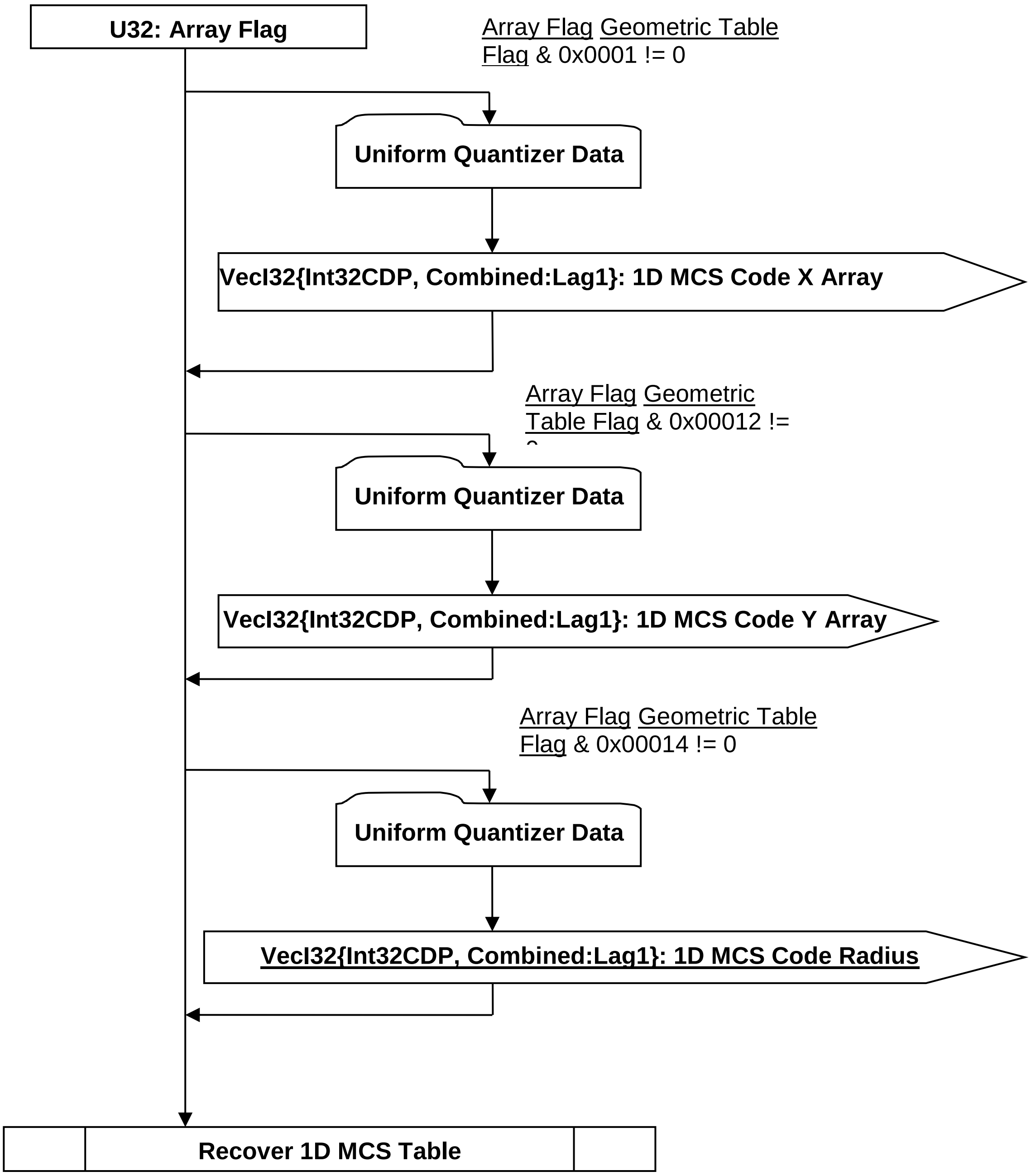


Figure 192 — 1D MCS Table data collection

U32: Array Flag

Array Flag indicates which arrays are not trivial and therefore encoded.

VecI32{Int32CDP, Combined:Lag1}: 1D MCS Code X Array

1D MCS Code X Array is a vector of quantizer "codes" for one group of 1D floating point values in MCS that represent X coordinates. 1D MCS Code X Array uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

VecI32{Int32CDP, Combined:Lag1}: 1D MCS Code Y Array

1D MCS Code Y Array is a vector of quantizer “codes” for one group of 1D floating point values in MCS that represent Y coordinates. 1D MCS Code Y Array uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

VecI32{Int32CDP, Combined:Lag1}: 1D MCS Code Radius Array

1D MCS Code Radius Array is a vector of quantizer “codes” for one group of 1D floating point values in MCS that represent radius or other MCS metric values. 1D MCS Code Radius Array uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

Recover 1D MCS Table

The representation of each surface or curve in ULP includes information that describes the extent of the surface or curve in the parameter domain. For curves the extent information is represented by two numbers, umin and umax, while for surfaces it is represented by two additional numbers for the other parametric direction, vmin and vmax. For surfaces or curves of Nurbs type such extent information is implied by the knot vector information. For surfaces or curves of other types the extent information needs be read from 1D MCS Table if the parameter value represents value in MCS, or Radian Table if the parameter value represents angle information. The detailed information about how the parameter domain information of different entities should be read is listed in the table below.

Table 183 — Parameter Domain

Entity Type	umin	umax	vmin	vmax
NURBS Surface	n/a (from knot)	n/a (from knot))	n/a (from knot)	n/a (from knot)
Plane	n/a (always 0)	1D MCS Table	n/a (always 0)	1D MCS Table
Cylinder	n/a (always 0)	Radian Table	n/a (always 0)	1D MCS Table
Cone	n/a (always 0)	Radian Table	n/a (always 0)	1D MCS Table
Sphere	n/a (always 0)	Radian Table	Radian Table	Radian Table
Torus	n/a (always 0)	Radian Table	Radian Table	Radian Table
XYZ NURBS Curve	n/a (from knot)	n/a (from knot)	n/a	n/a
XYZ Line	n/a (always 0)	n/a (from vertex geometry)	n/a	n/a
XYZ Circle	n/a (always 0)	Radian Table	n/a	n/a
UV NURBS Curve	n/a (from knot)	n/a (from knot)	n/a	n/a
UV Line	n/a (always 0)	n/a (from next uv curve)	n/a	n/a
UV Circle	Radian Table	Radian Table	n/a	n/a

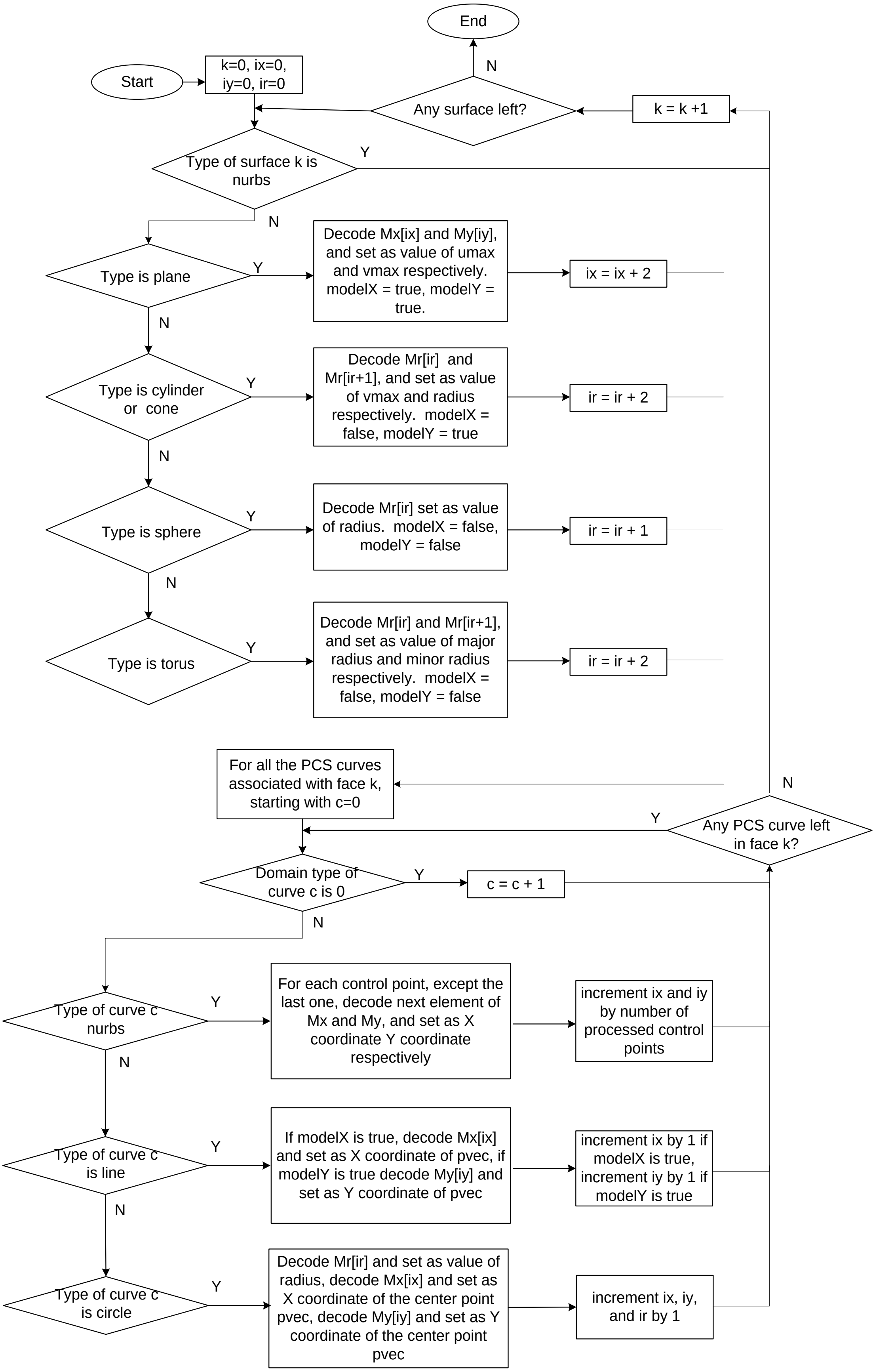


Figure 193 — Recover 1D MCS Table

The logic diagram to recover 1D MCS table information in the ULP from the 1D MCS Code X Array is shown in Figure 194- Recover 1D MCS Table 1D MCS Code X Array is denoted as Mx (with index ix), 1D MCS Code Y Array is denoted as My (with index iy), and 1D MCS Code Radius Array is denoted as Mr (with index ir). Note that each integer element in the arrays is decoded with Uniform Quantizer.

PCS Value Table

PCS Value Table stores the quantization representation of floating point values in PCS. If the ULP does not contain any such value, then the table is empty and bit 0x0100 in Geometric Tabe Flag is set to be 0. Each floating point value is encoded into an integer with uniform quantizer (see Uniform Quantizer Data) and then all the integers are grouped into an integer array. The integer array is then encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type.

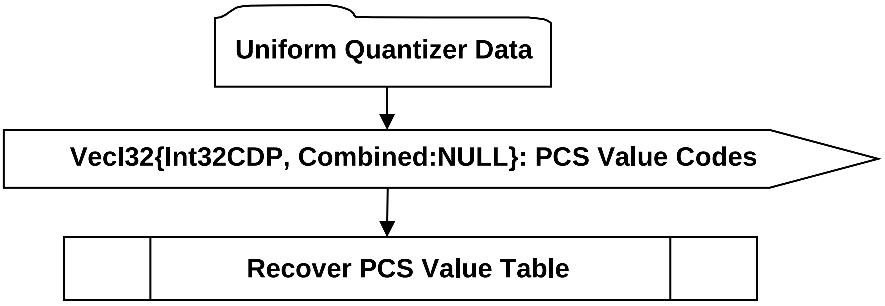


Figure 194 — PCS Value Table data collection

VecI32{Int32CDP, Combined:NULL}: PCS Value Codes

PCS Value Codes is a vector of quantizer “codes” for all the floating point values in PCS. PCS Value Codes uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

Recover PCS Value Table

The logic diagram to recover PCS Value Table information in the ULP from the PCS Value Codes is shown in Figure 119. Note that each integer element in the PCS Value Codes array is decoded with Uniform Quantizer.

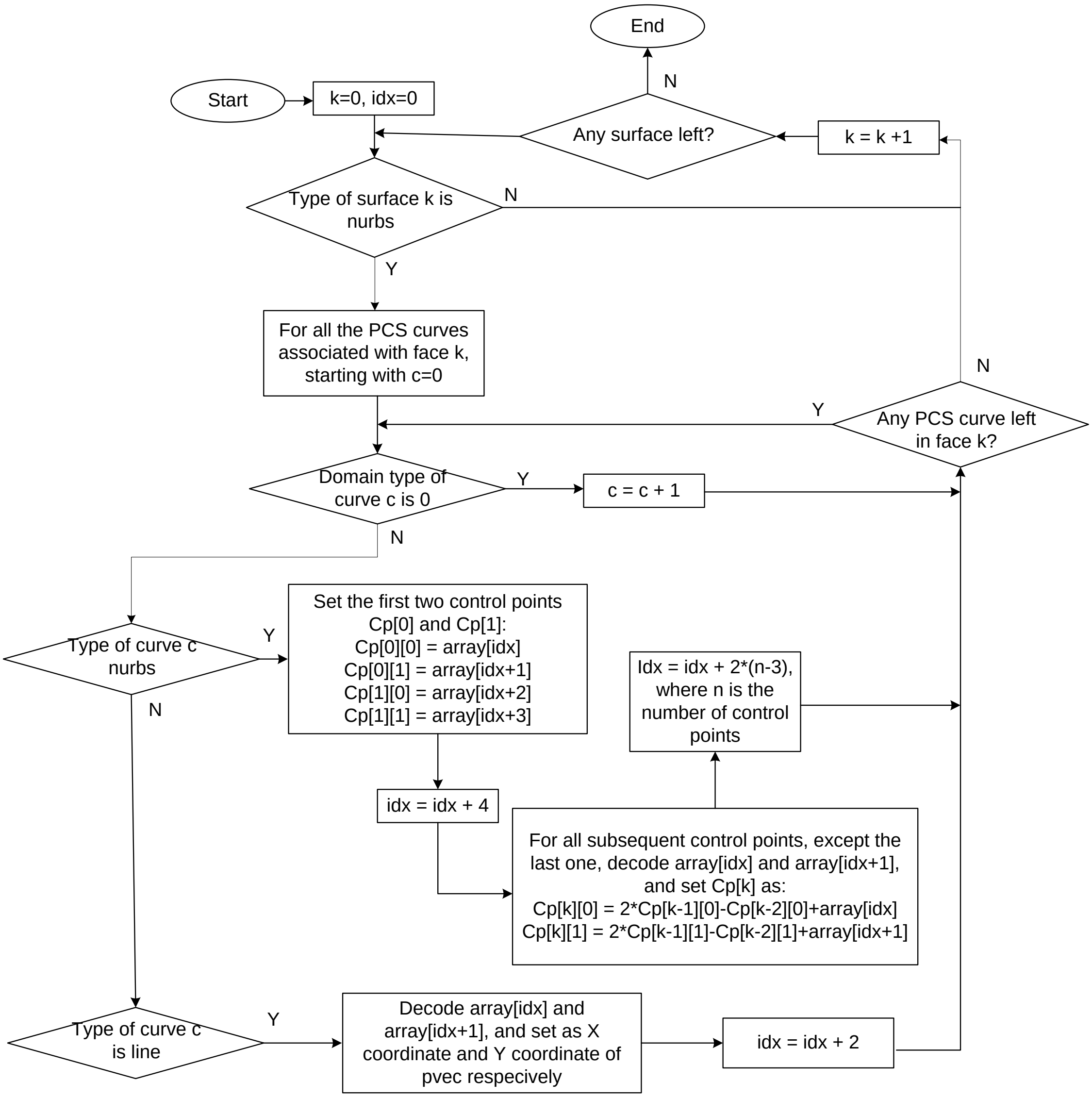


Figure 195 — Recover PCS Value Table

Radian Table

Radian Table stores the quantization representation of angular values. If the ULP does not contain any such angular value, then the table is empty and bit 0x0200 in Geometric Tabe Flag is set to be 0. Each angular value is encoded into an integer with uniform quantizer (see Uniform Quantizer Data) and then all the integers are grouped into an integer array. The integer array is then encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type.

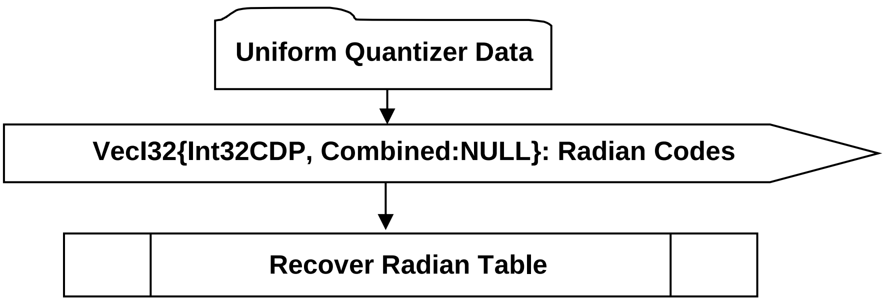


Figure 196 — Radian Table data collection

VecI32{Int32CDP, Combined:NULL}: Radian Codes

Radian Codes is a vector of quantizer “codes” for all the angular values. Radian Codes uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

Recover Radian Table

The logic diagram to recover Radian Table information in the ULP from the Radian Codes is shown in Figure 198 - Recover Radian Table. Note that each integer element in the Radian Codes array is decoded with Uniform Quantizer.

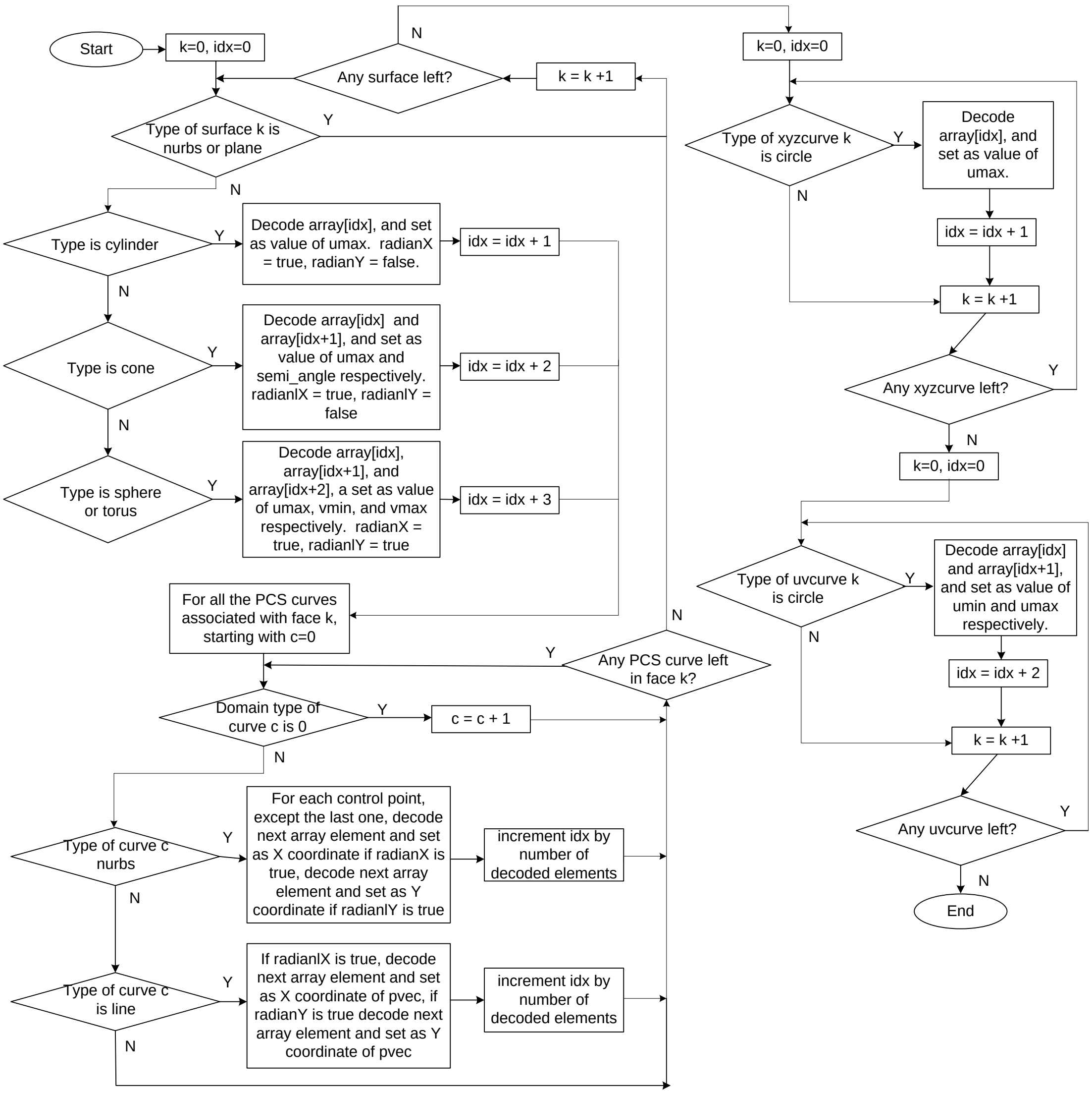


Figure 197 — Recover Radian Table

Weight Table

Weight Table stores the quantization representation of weight values. If the ULP does not contain any such weight value, then the table is empty and bit 0x0400 in Geometric Tabe Flag is set to be 0. Each weight value is encoded into an integer with uniform quantizer (see Uniform Quantizer Data) and then all the integers are grouped into an integer array. The integer array is then encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type

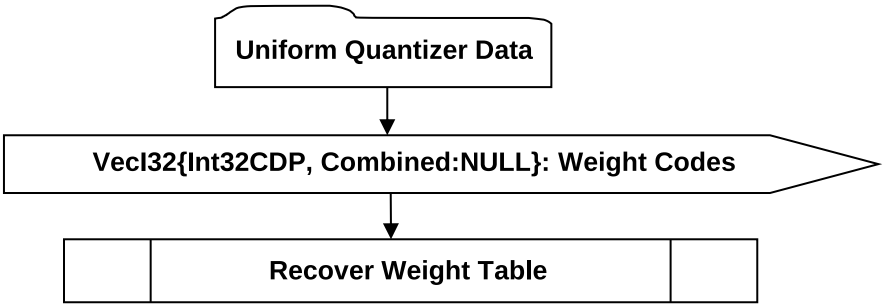


Figure 198 — Weight Table data collection

VecI32{Int32CDP, Combined:NULL}: Weight Codes

Weight Codes is a vector of quantizer “codes” for all the weight values. Weight Codes uses the Int32CDP CODEC described in Int32 Compressed Data Packet with Combined Predictor Type to compress and encode data.

Recover Weight Table

The logic diagram to recover Weight Table information in the ULP from the Weight Codes is shown in Figure - Recover Weight Table. Note that each integer element in the Weight Codes array is decoded with Uniform Quantizer.

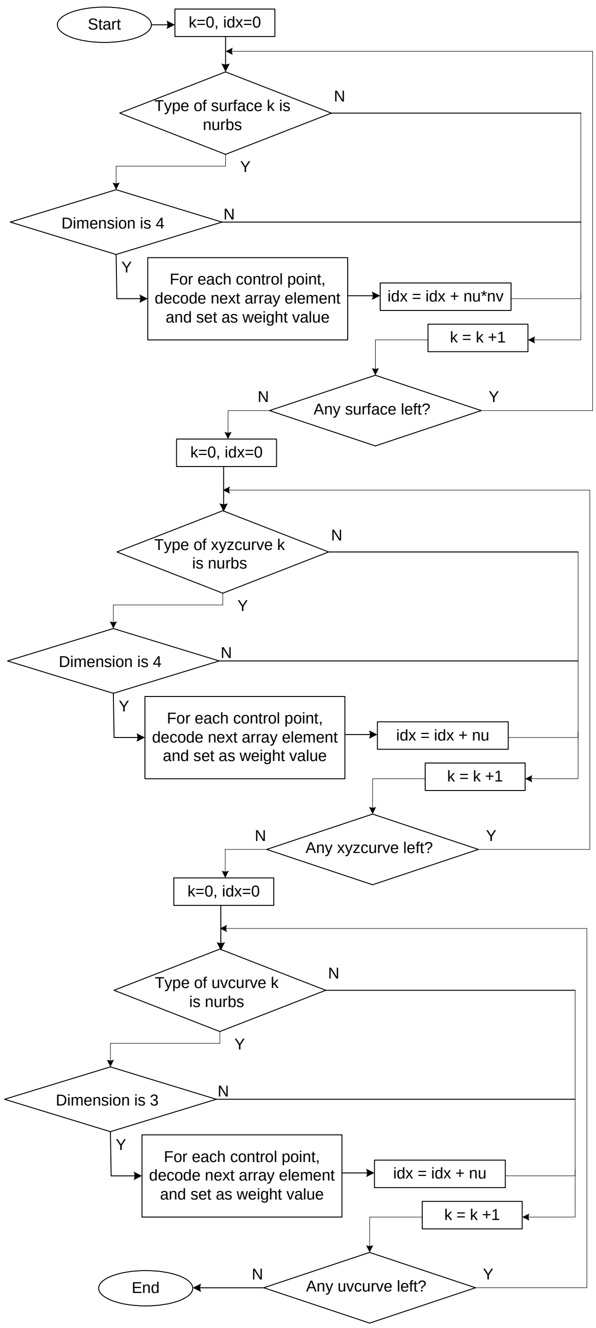


Figure 199 — Recover Weight Table

G.1.3 Material Attribute Element Properties

The properties attached to material attribute are standard JT properties, and the logic diagram to read the properties attached to a material attribute is shown in Figure 201 - Material Attribute Element Properties.

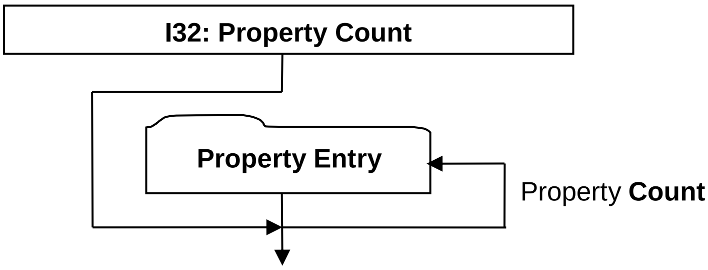


Figure 200 — Material Attribute Element Properties

I32: Property Count

Property count is the number of properties attached.

Property Entry

Standard JT property entry, consisting of key and value pair.

G.1.4 Information Recovery

The information in ULP is classified as “essential information” that is explicitly written on disk, and “derivative information” that can be computed from the “essential information”. How “essential information” of ULP can be read from disk was covered in previous sections, and this section focuses on the logic to recover “derivative information” from “essential information”.

The derivative information consists of curve information either in the parameter or model space. For example, the PCS curves associated with an untrimmed face can be inferred from the parameter domain of the surface, or an MCS curve may be computed from vertex information and/or the combination of corresponding PCS curve geometry and surface geometry, etc.. Shown in Figure Information Recovery is the high level diagram to recover “derivative information”. First, all the PCS line geometry are recovered from the associated surface domain information if the domain type of those PCS curves, stored in its associated coedge, are of value 1, 2, 3, 4 meaning that the PCS curve is identical to one of the parameter boundaries of the surface. The PCS curve geometry is then finalized by leveraging the knowledge that all the PCS curves in the same loop are joined in a head to tail fashion. The geometry of two end points of every MCS curve can then be computed from the corresponding PCS curve and surface geometry. Second, the MCS curve geometry is recovered depending on its type. If the MCS curve type is 0, 1, or 2, then the geometry of its two end points is used to compute the curve geometry. If the MCS curve type is 3, then its geometry is computed by projecting PCS curve onto the surface geometry. The logical steps that are displayed with dark colour indicate steps that will be elaborated in more detail later.

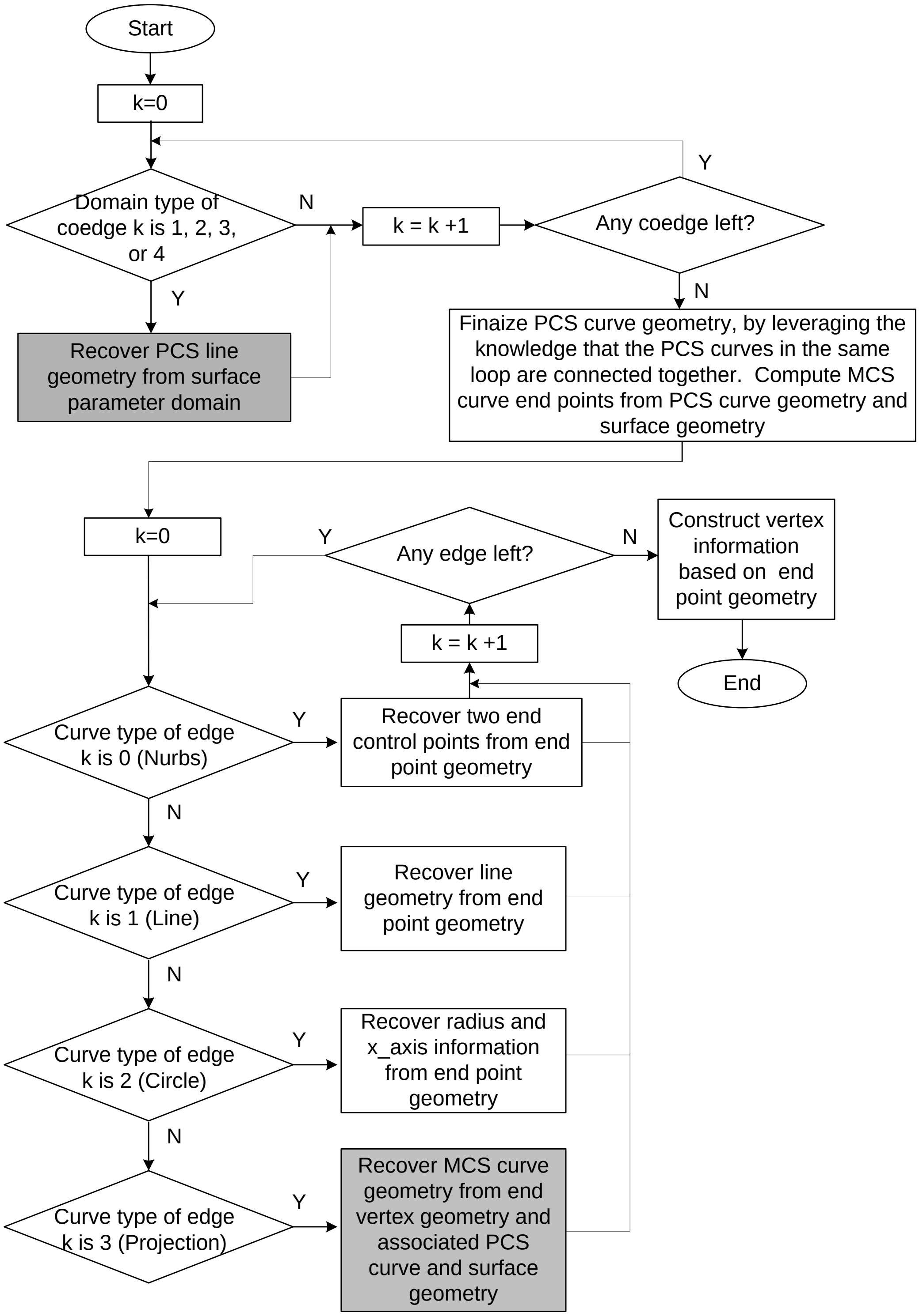


Figure 201 — Information Recovery

PCS Curve Recovery from Surface Domain

Shown in Figure 203 - PCS Curve Recovery from Surface Domain is the diagram illustrating how the PCS curve geometry is recovered from surface parameter domain information.

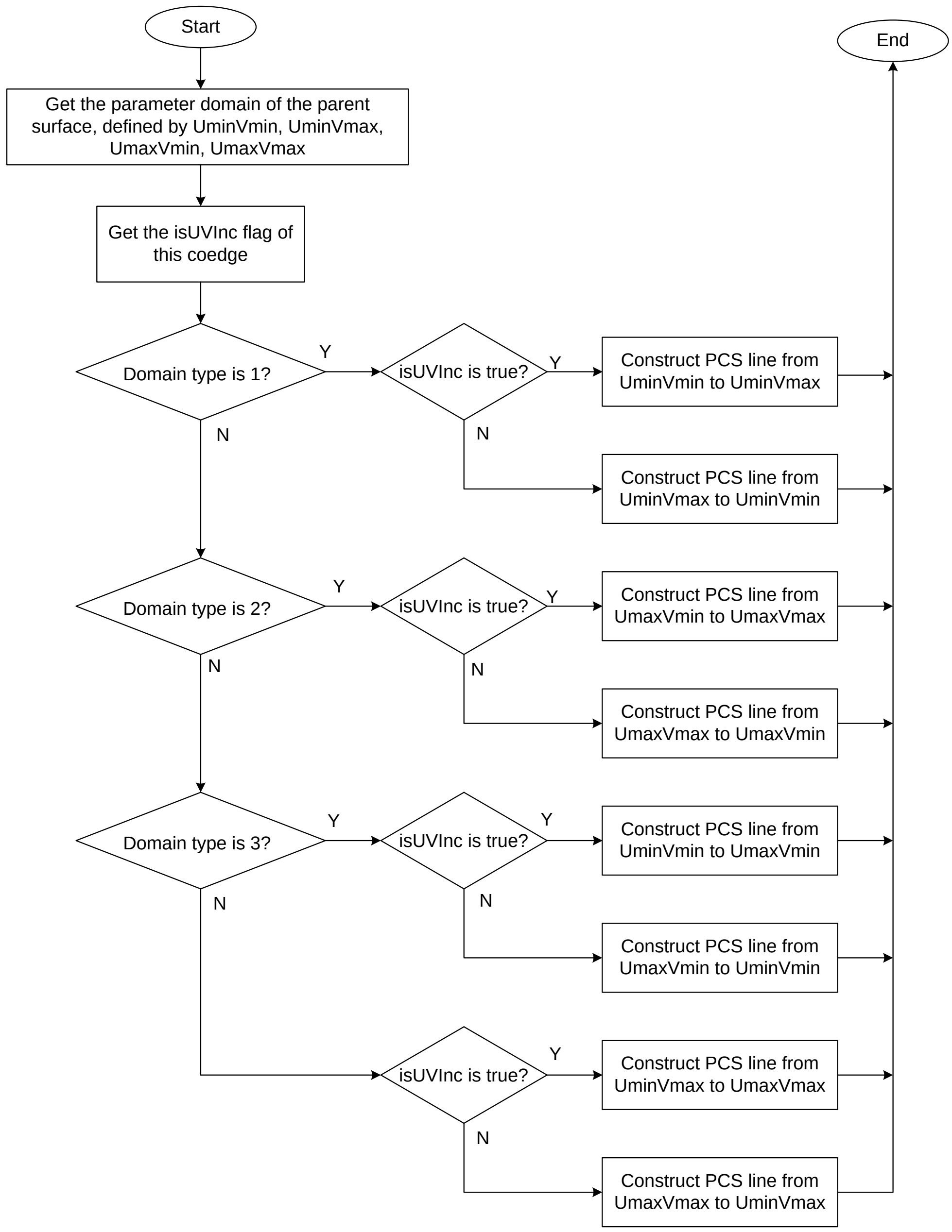


Figure 202 — PCS Curve Recovery from Surface Domain

MCS Curve Recovery

Shown in Figure 204 - MCS Curve Recovery is the diagram illustrating how MCS curve geometry is recovered from its end point geometry, and/or its associated PCS curve geometry and surface geometry. If the associated PCS curve is coincident with one of the parameter boundaries of the parent surface, then the MCS curve can be recovered from parent surface geometry. Otherwise, if the surface type is planar and PCS curve is of type Nurbs or circle, then the MCS curve geometry can be recovered by projecting the PCS curve from parameter domain to model space onto the planar surface.

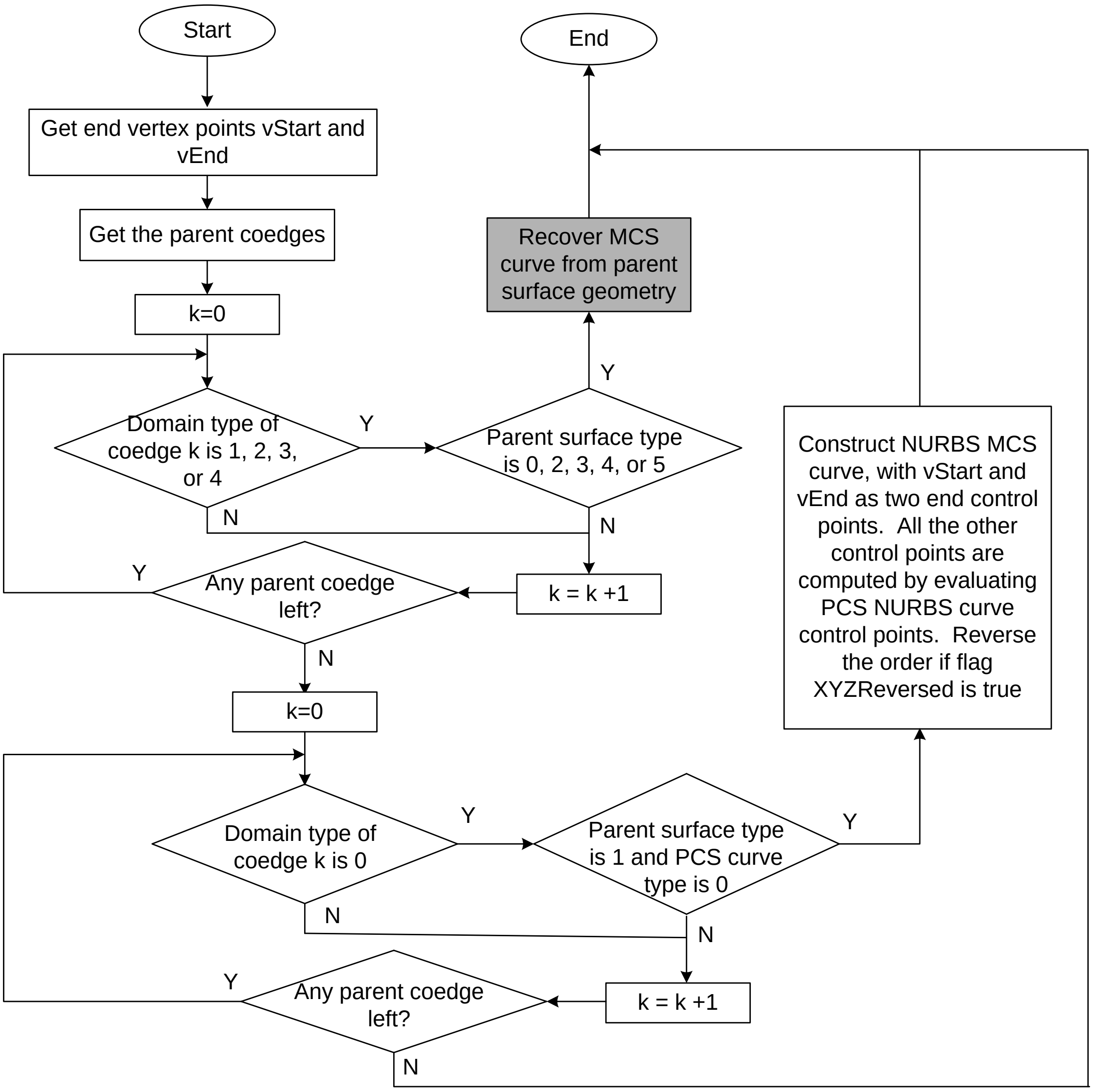


Figure 203 — MCS Curve Recovery

Shown in Figure 205 - MCS Curve Recovery from Surface Geometry is the detailed description of how MCS curve can be recovered from surface geometry.

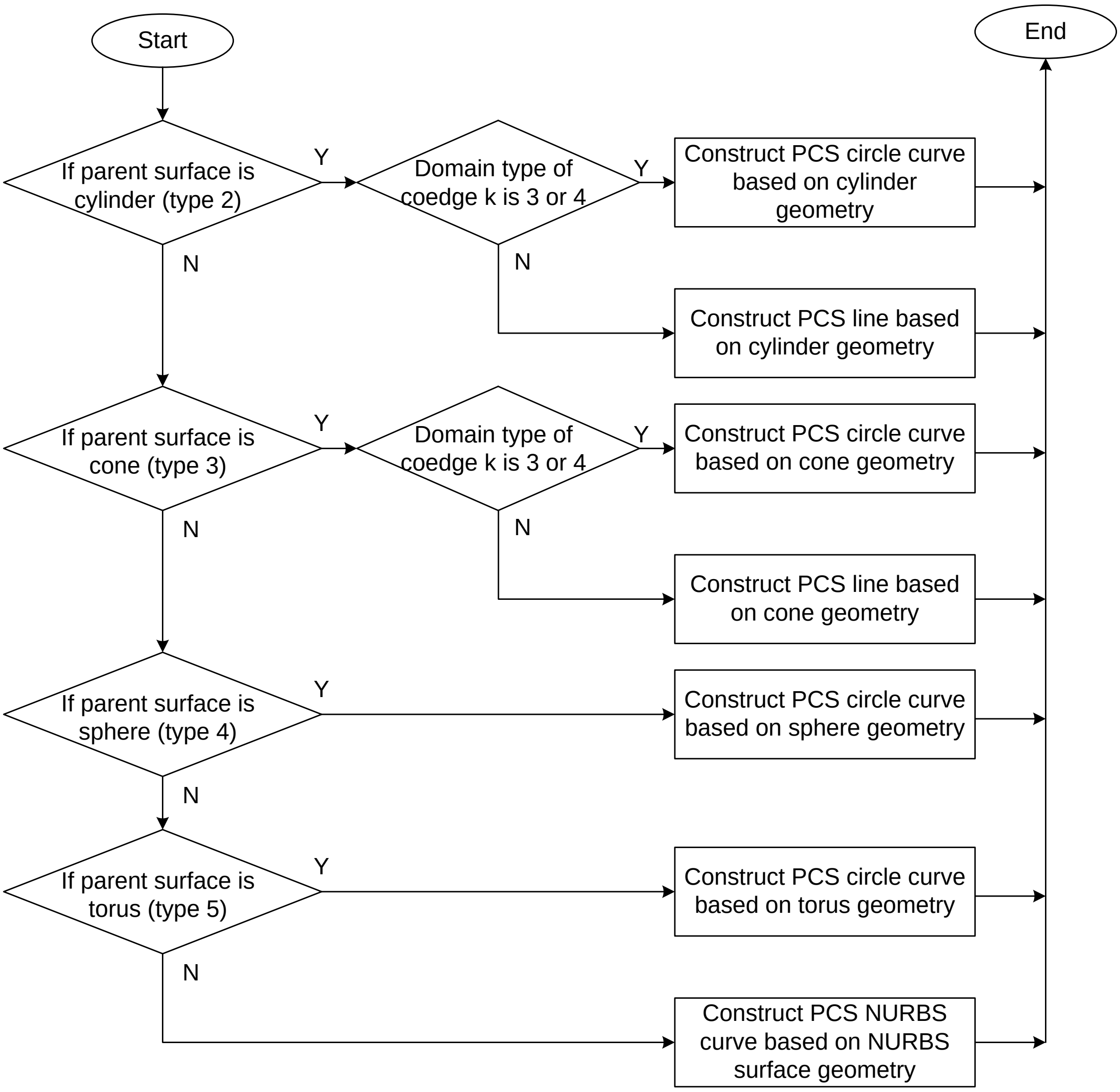


Figure 204 — MCS Curve Recovery from Surface Geometry

Annex H
JT Smart Topology Table (STT) Segment

JT Smart Topology Table (hereafter referred to as STT) Segment contains an Element that defines the lightweight B-Rep description for a particular Part.

JT STT Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT STT Segment type supports compression on all element data, so all elements in JT STT Segment use the Logical Element Header Compressed form of element header data.

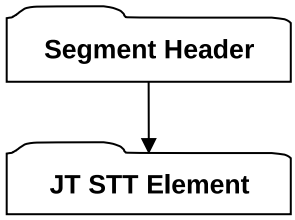


Figure 205 — JT STT Segment data collection

Complete description for Segment Header can be found in the File Header section of Base Format Description under Data Segment.

H.1 JT STT Element

Object Type ID: 0xca7e6f89, 0x97c8, 0x47f0, 0x9f, 0xca, 0x16, 0x99, 0xc, 0xfb, 0xe2, 0x17

JT STT Element represents a lightweight B-Rep data. It contains complete B-Rep topology information, analytic geometry information, and attribute information.

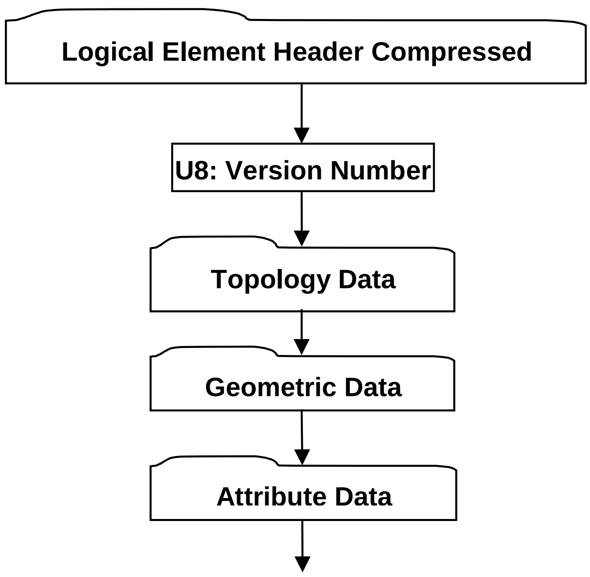


Figure 206 — JT STT Element data collection

Complete description for Logical Element Header Compressed can be found in the File Header section of Base Format Description under Data Segment, Data.

H.1.1 Topology Data

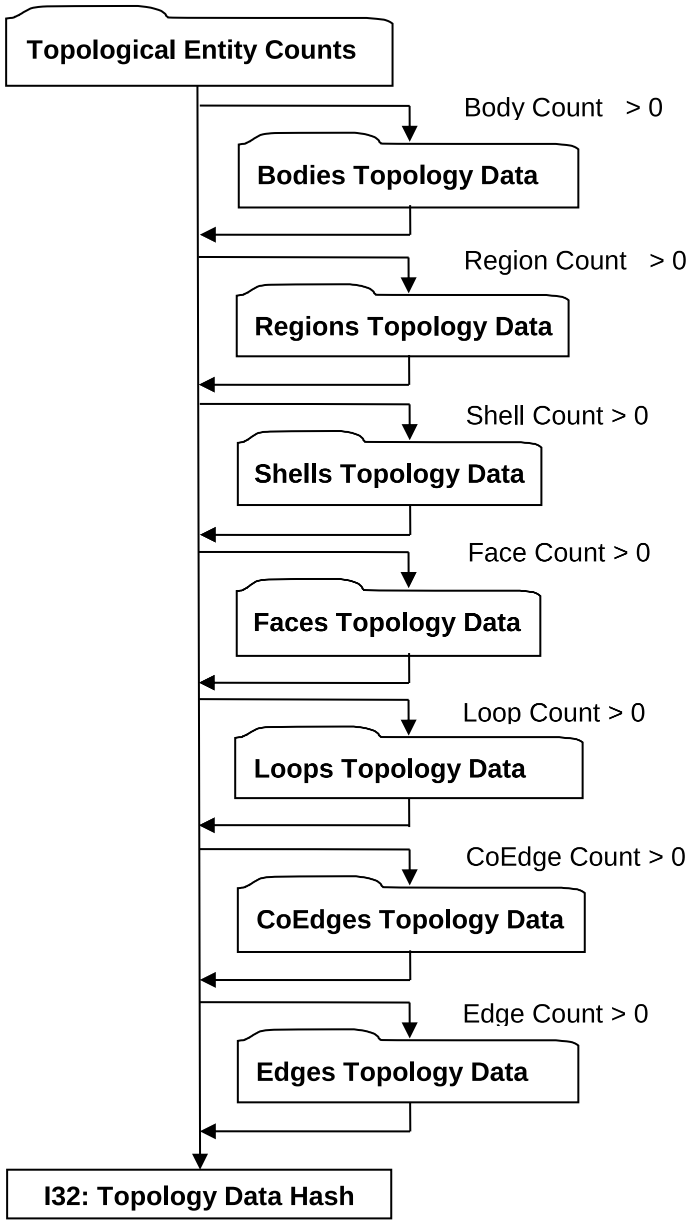


Figure 207 — Topology Data collection

Topological Entity Counts

Topological Entity Counts data collection defines the counts for each of the various topological entities within a STT.

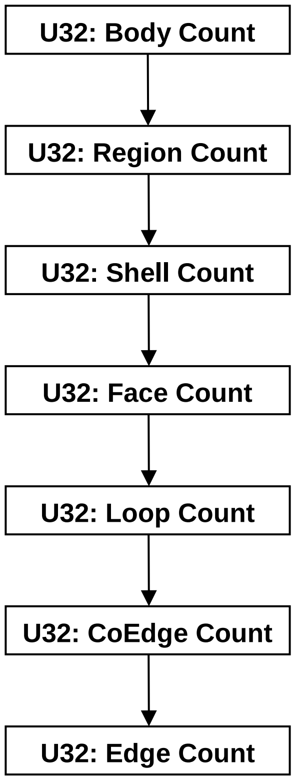


Figure 208 — Topological Entity Counts data collection

U32: Body Count

Body Count indicates the number of topological body entities.

U32: Region Count

Region Count indicates the number of topological region entities.

U32: Shell Count

Shell Count indicates the number of topological shell entities.

U32: Face Count

Face Count indicates the number of topological face entities.

U32: Loop Count

Loop Count indicates the number of topological loop entities.

U32: CoEdge Count

CoEdge Count indicates the number of topological coedge entities.

U32: Edge Count

Edge Count indicates the number of topological edge entities.

Body Topology Data

Body Topology Data defines the disjoint set of non-overlapping Regions making up each Body. Each Body is defined by one or more non-overlapping Regions. A Body is the sum of all regions of this Body.

Each Body's defining Regions are identified in the Start Region Index. The indices of all the Regions in a single Body are contiguous. The first Region index of the first Body is 0. The first Region index of Body $\mathrm{k}, \mathrm{k} \geq 0$ is the value of the kth element in Start Region Index. The last Region index of Body $\mathrm{k}, \mathrm{k} \geq 0$ equals to the first Region index of Body k+1 minus 1.

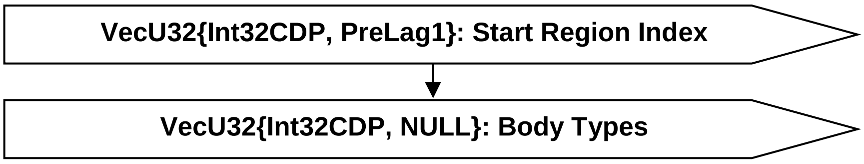


Figure 209 — Body Topology Data collection

VecU32{Int32CDP, PreLag1}: Start Region Index

Start Region Index is a vector of indices representing the integer index value of start region for this body, which is equal to the last region index from last body plus 1. Start Region Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Body Types

Each Body has a type identifying the type of the Body. Body Types indexed by Body Index contains the body type enum values for corresponding Bodies.

In an uncompressed/decoded form the type values can be one of the following values:

Table 184 — Body Type Values

0	A body consisting solely of an unbounded void region.
1	A topologically zero dimensional manifold body containing a single isolated vertex.
2	A topologically zero dimensional manifold body containing two or more isolated vertices.
3	A topologically one dimensional manifold body containing one or more connected sets of edges, where any vertex is at the junction of no more than two edges.
4	A topologically two dimensional manifold body containing one or more connected sets of faces, where any edge is at the junction of no more than two faces.
5	A topologically three dimensional manifold body containing one or more disjoint and separate solid regions. All faces form a boundary between a solid and a void region.
6	A body which is non-manifold and/or of mixed topological dimensionality.
7	The body type is not specified.

Body types uses the Int32 version of the CODEC to compress and encode data.

Region Topology Data

Region Topology Data defines the disjoint set of non-overlapping Shells making up each Region. Each Region is defined by one or more non-overlapping Shells. The volume of a Region is that volume lying inside each “anti-hole Shell” and outside each simply-contained “hole Shell” belonging to the particular Region. A Region is analogous to a dimensionally elevated face where Region corresponds to Face and Shell corresponds to Trim Loop.

Each Region’s defining Shells are identified in the Start Shell Index. The indices of all the Shells in a single Region are contiguous. The first Shell index of the first Region is 0. The first Shell index of Region $\mathrm{k}, \mathrm{k} \geq 0$ is the value of the kth element in Start Shell Index. The last Shell index of Region $\mathrm{k}, \mathrm{k} \geq 0$ equals to the first Shell index of Region k+1 minus 1.

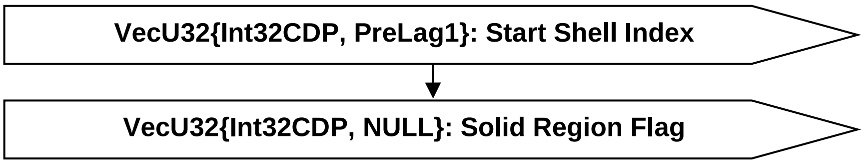


Figure 210 — Region Topology Data collection

VecU32{Int32CDP, PreLag1}: Start Shell Index

Start Shell Index is a vector of indices representing the integer index value of start shell for this region, which is equal to the last shell index from last region plus 1. Start Shell Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Solid Region Flag

Solid Region Flag is a vector of flags representing the solid property of the Regions. The flag value is set according to the table below . Solid Region Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 185 — Solid Region Flag Values

= 0	Region is not solid
= 1	Region is solid

Shell Topology Data

Shell Topology Data defines the set of topological adjacent faces making up each shell. A shell’s set of topological adjacent faces define a single (usually closed) two manifold solid that in turn defines the boundary between the finite volume of space enclosed within the shell and the infinite volume of space outside the shell. In addition, each shell has a flag that denotes whether the shell refers to the finite interior volume (i.e. a “inner shell”), the infinite exterior volume (i.e. an “outer shell”), or an open shell. Because an inner shell and its counterpart outer shell share the same set of faces, face information is only represented for outer shells and open shells. An outer or open shells for which face information is represented is called “a represented shell”.

Each represented shell’s defining faces are identified in the Start Face Index. The indices of all the faces in a single shell are contiguous. The first face index of the first shell is 0. The first face index of represented shell $\mathrm{k}, \mathrm{k} \geq 0$ is the value of the kth element in Start Face Index. The last face index of shell $\mathrm{k}, \mathrm{k} \geq 0$ equals to the first face index of shell k+1 minus 1.

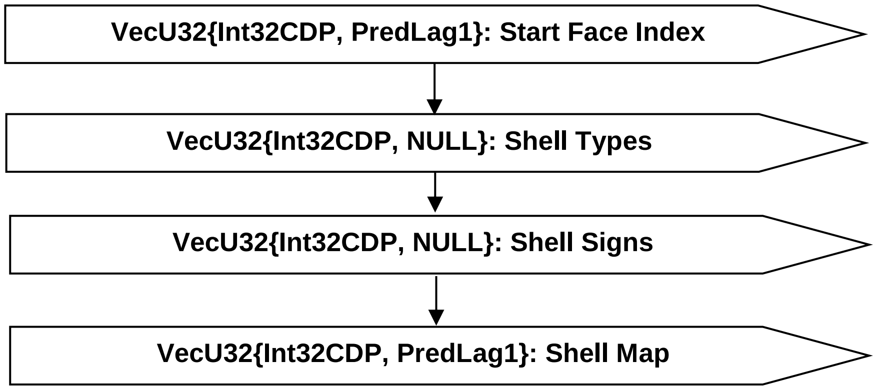


Figure 211 — Shell Topology Data collection

VecU32{Int32CDP, PredLag1}: Start Face Index

Start Face Index is a vector of indices representing the integer index value of start face for this shell, which is equal to the last face index from last shell plus 1. The length of this array is equal to the number of represented shells that are either open or positive. Start Face Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Shell Types

Each shell has a type identifying the type of the Shell. Shell Types indexed by the Shell index contains enum type values of corresponding Shells.

In an uncompressed/decoded form the type values is from the following enumeration:

Table 186 — Shell Types

0	Shell has a single acorn vertex
1	Shell has one or more wireframe edges
2	Shell has no wireframe edges, but one or more faces
3	Shell has both wireframe edges and faces
4	The shell type is not specified.

Shell types uses the Int32 version of the CODEC to compress and encode data.

VecU32{Int32CDP, NULL}: Shell Signs

Each shell has a sign. The Shell Signs array indexed by shell index contains the sign values of all the shells.

In an uncompressed/decoded form the type values is from the following enumeration:

Table 187 — JT STT Shell Signs

0	JT_SHELL_sign_positive_c	There is a subset of the faces of the shell which divides space into two volumes, the volume inside the shell being finite; for example, the outermost shell of a solid region.
1	JT_SHELL_sign_negative_c	There is a subset of the faces of the shell which divides space into two volumes, the volume inside the shell being infinite; for example, an inner shell of a solid region.
2	JT_SHELL_sign_open_c	There is no subset of faces of the shell which divides

		space into two volumes, in other words, all points are either in or on the shell; for example, any shell consisting only of wireframe edges.
3	JT_SHELL_sign_unset	The shell sign is not specified

Shell signs uses the Int32 version of the CODEC to compress and encode data.

VecU32{Int32CDP, PredLag1}: Shell Map

Shell Map is a vector that indicates the index of represented shell for every shell. Shell Map is compressed and encoded using the Int32CDP CODEC described in 12.1.1 Int32 Compressed Data Packet.

Face Topology Data

A Face shall be trimmed with at least one “anti-hole” Trim Loop and may be trimmed with one or more “hole” Trim Loops.

Each face’s defining loops are identified in the Start Loop Index. The indices of all the loops in a single face are contiguous. The first loop index of the first face is 0. The first loop index of face $k, k \geq 0$ is the value of the kth element in Start Loop Index. The last loop index of face $k, k \geq 0$ equals to the first loop index of face k+1 minus 1.

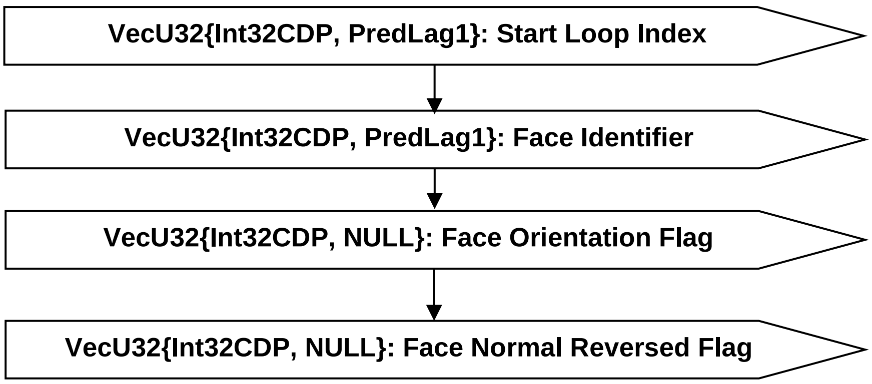


Figure 212 — Face Topology Data collection

VecU32{Int32CDP, PredLag1}: Start Loop Index

Start Loop Index is a vector of indices representing the integer index value of start loop for each face, which is equal to the last loop index of previous face plus 1. Start Loop Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, PredLag1}: Face Identifier

Assigned identifiers of each face. Face Identifier is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Face Orientation Flag

Face Orientation Flag is a vector of flags, indexed by Face Identifier, representing the orientation of the face with respect to the volume bounded by its owner shell. The flag is set to 1 if face’s normal points into its owner shell, and 0 otherwise. Face Orientation Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 188 — Face Orientation Flag Values

= 0	The face's normal points into its owner shell
= 1	The face's normal points away from its owner shell

VecU32{Int32CDP, NULL}: Face Normal Reversed Flag

Face Normal Reversed Flag is a vector of flags, indexed by face identifier, representing the normal direction of the face with respect to the underlying surface normal. The flag is set to be 1 if the face normal is anti-parallel to the surface normal, and 0 otherwise. Face Orientation Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 189 — Face Normal Reversed Flag values

= 1	the face normal is opposite to the surface normal
= 0	the face normal is parallel to the surface normal

Loop Topology Data

A loop defines in parameter space a 1D boundary around which geometric surfaces are trimmed to form a face. Loops Topology Data specifies the CoEdges making up each loop along with an anti-hole flag and identifier tag for each loop. Each Loop's defining CoEdges are identified in the Start CoEdge Index. The indices of all the CoEdges in a single Loop are contiguous. The first CoEdge index of the first Loop is 0. The first CoEdge index of Loop $k, k \geq 0$ is the value of the kth element in Start CoEdge Index. The last CoEdge index of Loop $k, k \geq 0$ equals to the first CoEdge index of Loop k+1 minus 1.

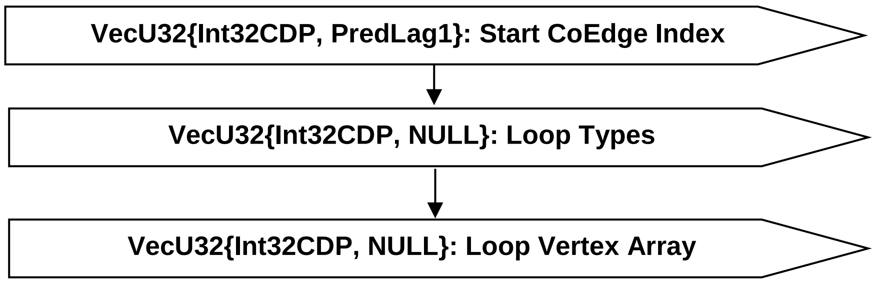


Figure 213 — Loop Topology Data collection

VecU32{Int32CDP, PredLag1}: Start CoEdge Index

Start CoEdge Index is a vector of indices representing the integer index value of start CoEdge for this Loop, which is equal to the last coedge index from previous loop plus 1. Start CoEdge Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Loop Types

Each Loop has a type identifying the type of the Loop.

In an uncompressed/decoded form the type values is from the following enumeration:

Table 190 — Loop Types

0	Loop is just a vertex without any edges
1	Loop has no interior, e.g. from a wire body
2	Simple peripheral loop
3	Loop is a simple hole

4	Winding loop on a periodic surface e.g. a circle on a cylinder or doughnut
5	Loop is a hole around the surface singularity e.g. chopping the top off a cone
6	An apparently peripheral loop on a doubly closed surface
7	An apparent hole in a doubly closed surface
8	A loop dividing a periodic degenerate surface in two (contains just one pole)
9	Invalid loop or algorithm failure
10	The loop type is not specified

Loop types uses the Int32 version of the CODEC to compress and encode data.

Loop Vertex Array

The element of Loop Vertex Array is the associated vertex identifier if it exists, -1 if the loop does not have associated vertex. Loop Vertex Array uses the Int32 version of the CODEC to compress and encode data.

CoEdges Topology Data

A CoEdge defines a parameter space edge trim Loop segment (i.e. the projection of an Edge into the parameter space of the Face). A CoEdge respresents the oriented use of an Edge by a Loop.

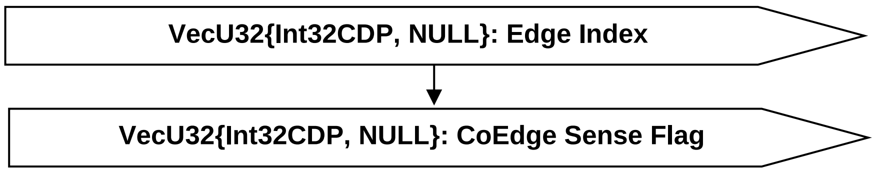


Figure 214 — CoEdge Topology Data collection

VecU32{Int32CDP, NULL}: Edge Index

Edge Index indicates which Edge each CoEdge belongs to. Edge Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: CoEdge Sense

CoEdge Sense Flag is a vector of flags representing the direction of the CoEdge with respect to the corresponding Edge. The flag is set to 1 if the CoEdge direction conforms to its corresponding Edge or 0 otherwise. CoEdge Sense Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 191 — CoEdge Sense Flag Values

= 0	The CoEdge direction is the opposite to its corresponding Edge
= 1	The CoEdge direction is the same as its corresponding Edge

Edge Topology Data

An Edge defines a model space trim Loop segment. Its boundary is a collection of zero, one or two vertices.

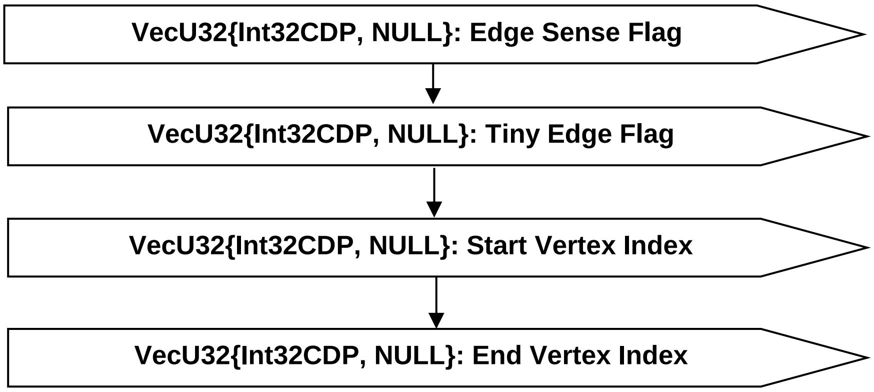


Figure 215 — Edge Topology Data collection

VecU32{Int32CDP, NULL}: Edge Sense Flag

Edge Sense Flag is a vector of flags representing the direction of the edge with respect to the corresponding curve. The flag is set to 1 if the edge direction conforms to its corresponding curve and 0 otherwise. Edge Sense Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 192 — Edge Sense Flag Values

= 0	The Edge direction is the opposite to its corresponding curve
= 1	The Edge direction is the same as its corresponding curve

VecU32{Int32CDP, NULL}: Tiny Edge Flag

Tiny Edge Flag is a vector of flags representing whether the edge is tiny (e. g. shorter than $1*10^{-5}$ meters). The flag is set to 1 if the edge is tiny and 0. Tiny Edge Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Start Vertex Index

Start Vertex Index contains start vertex index of each edge. Start Vertex Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: End Vertex Index

End Vertex Index contains end vertex index of each edge. End Vertex Index is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

I32: Topology Data Hash

The Topology Data Hash is the combined hash of all the topology data. Refer to the Hashing Annex for a more detailed description of hashing.

H.1.2 Geometric Data

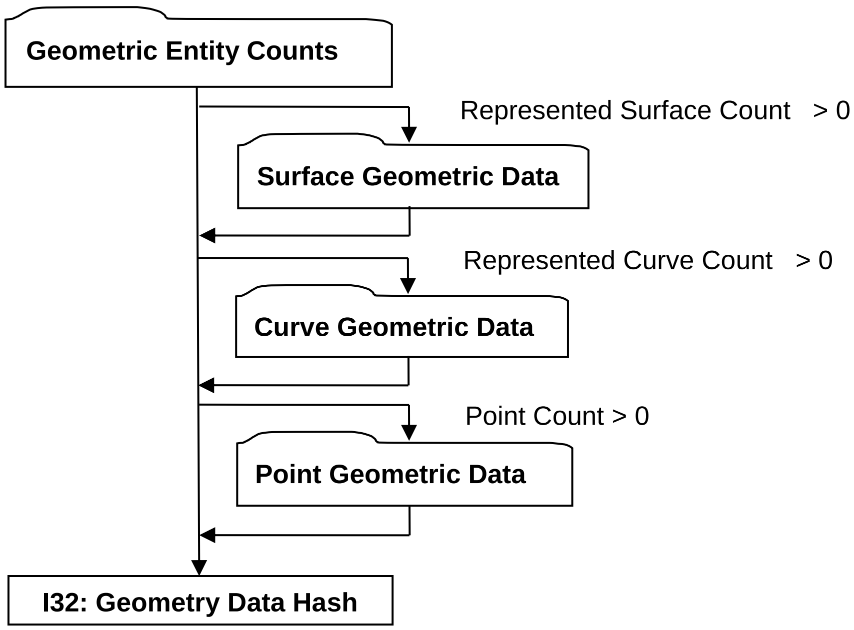


Figure 216 — Geometric Data collection

Geometric Entity Counts

Geometric Entity Counts data collection defines the counts for each of the various geometric entities within a STT.

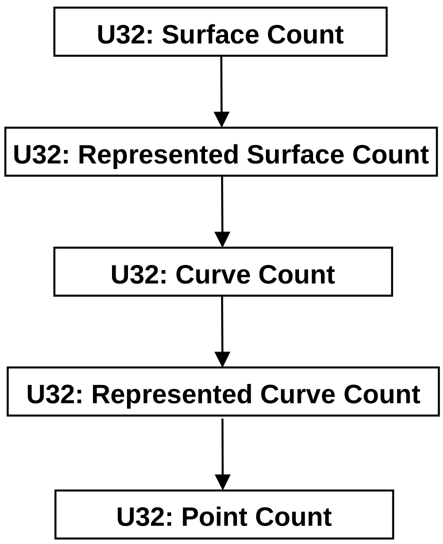


Figure 217 — Geometric Entity Counts

U32: Surface Count

Surface Count indicates the number of distinct geometric surface entities in the STT.

U32: Represented Surface Count

Represented Surface Count indicates the number of geometric surfaces that are represented in the STT.

U32: Curve Count

Curve Count indicates the number of distinct geometric curve entities in the STT.

U32: Represented Curve Count

Represented Curve Count indicates the number of geometric curves that are represented in the STT.

U32: Point Count

Point Count indicates the number of distinct geometric point entities in the STT.

Surface Geometric Data

Surface Geometric Data defines a collection of surfaces and their mapping to the original B-Rep surfaces. Only surfaces of type Plane, Cylinder, Cone, Sphere, and Torus are represented.

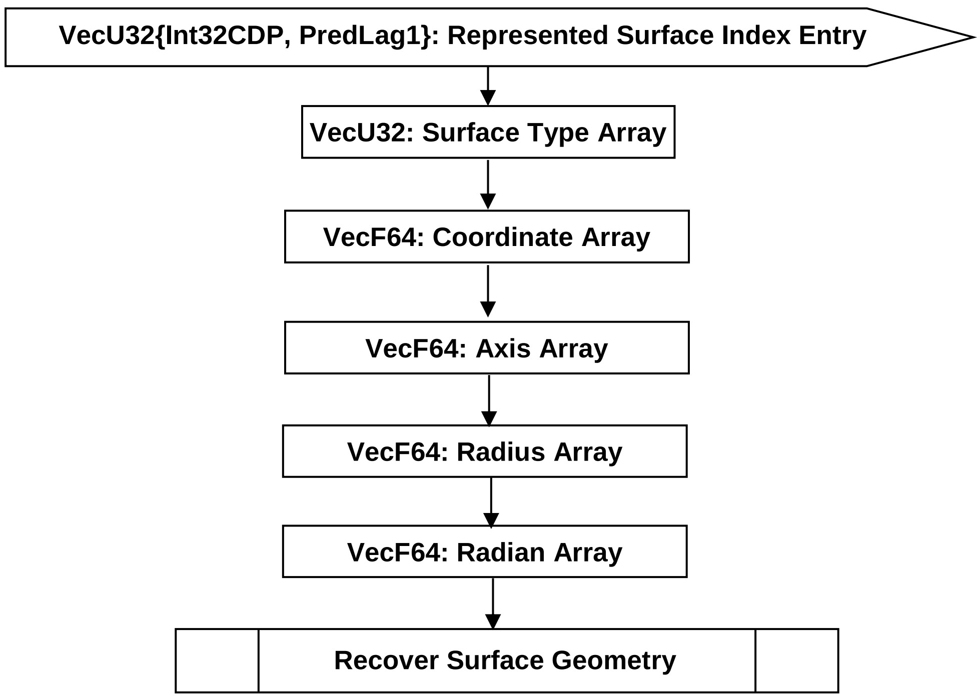


Figure 218 — Surface Geometric Data collection

VecU32{Int32CDP, PredLag1}: Represented Surface Index Entry

Represented Surface Index Entry is a vector of integers that stores the index of the represented surfaces. Represented Surface Index Entry is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32: Surface Type Array

Surface Type Array is a vector of integers that stores the enumeration values of the analytical surface type. Surface Type Array is encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 193 — Represented Surface Types

0	PLANE
1	CYLINDER
2	CONE
3	SPHERE
4	JT_SURF_TORUS

The detailed definitions of the Surface Types can be found: F.2.4.3.1 PLANE; F.2.4.3.2 CYLINDER; F.2.4.3.3 CONE; F.2.4.3.4 SPHERE; F.2.4.3.5 TORUS.

VecF64: Coordinate Array

Coordinate Array contains an array of double precision floating point numbers that represent the collection of point coordinate information in the definition of the analytic surface entities. The composite type VecF64 is defined in the Symbols table of Notational Conventions.

VecF64: Axis Array

Axis Array contains an array of double precision floating point numbers that represent the collection of unit vector information in the definition of the analytic surface entities. The composite type VecF64 is defined in the Symbols table found in Notational Conventions.

VecF64: Radius Array

Radius Array contains an array of double precision floating point numbers that represent the collection of radius information in the definition of the analytic surface entities. The composite type VecF64 is defined in in the Symbols table found in Notational Conventions.

VecF64: Radian Array

Radian Array contains an array of double precision floating point numbers that represent the collection of radian information in the definition of the analytic surface entities. The composite type VecF64 is defined in in the Symbols table found in Notational Conventions.

Recover Surface Geometry

The logic diagram to recover surface geometry from the arrays is shown below. All the represented surfaces are processed one by one sequentially, with its definition recovered from relevant arrays. In the diagram, vCoord represents “coordinate array”, vAxis represents “axis array”, vRadius represents “radius array”, and vRadian represents “radian array”.

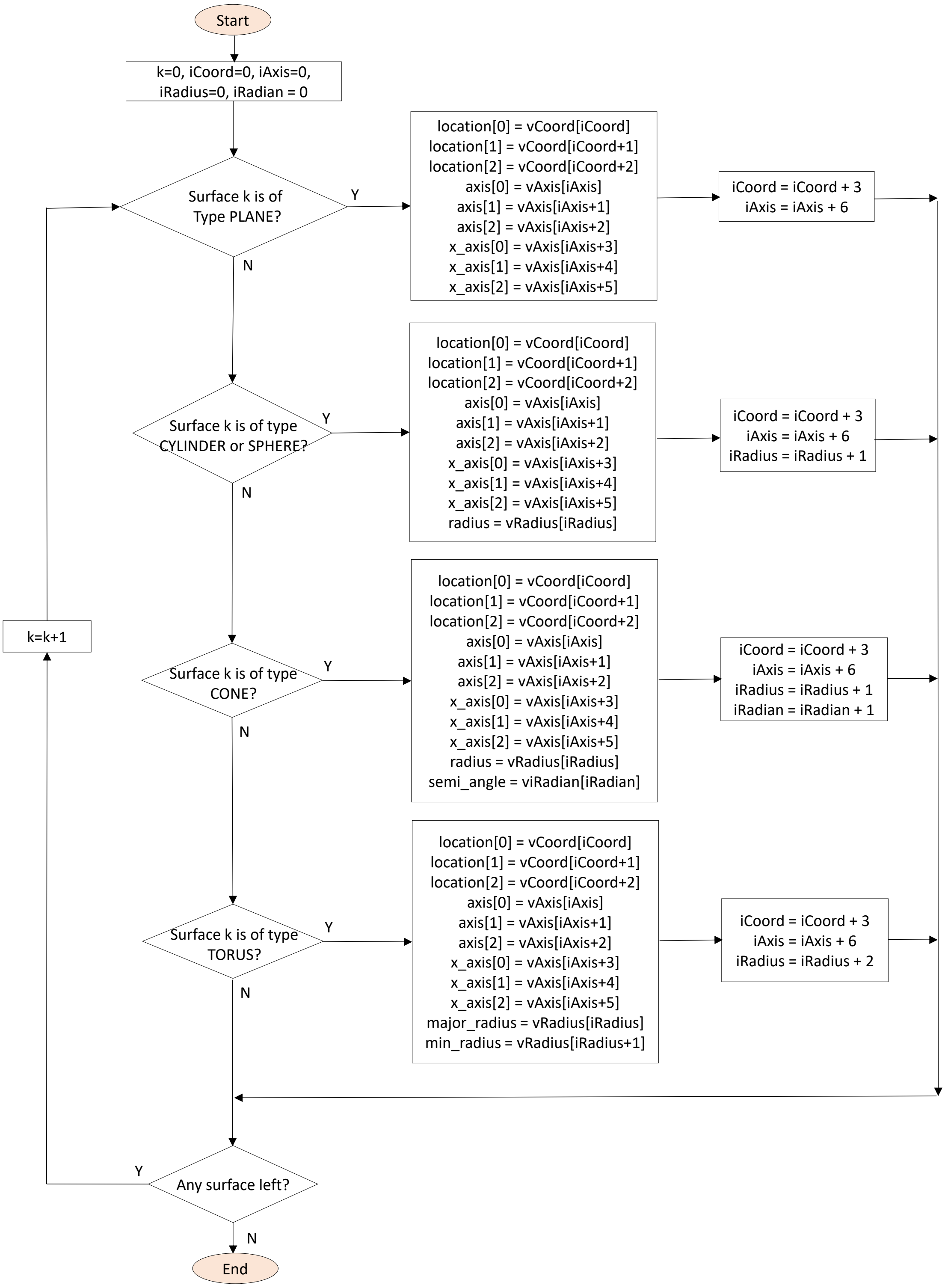


Figure 219 — Surface Geometry Recovery

Curve Geometric Data

Curve Geometric Data defines a collection of analytic curves and their mapping to the original B-Rep curves.

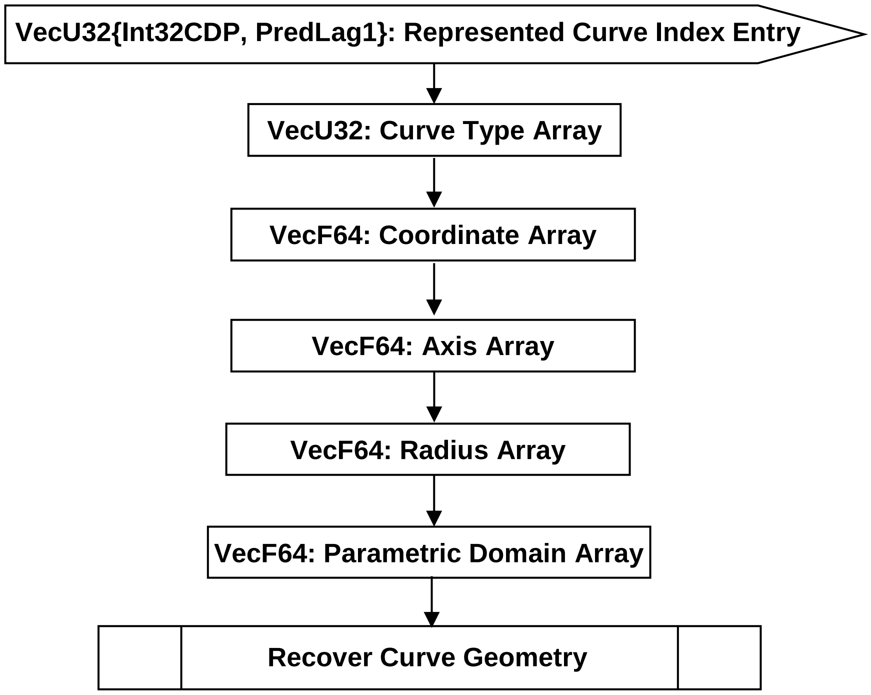


Figure 220 — Curve Geometric Data collection

VecU32{Int32CDP, PredLag1}: Represented Curve Index Entry

Represented Curve Index Entry is a vector of integers that stores the indices of curves represented in STT. Represented Curve Index Entry is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32: Curve Type Array

Curve Type Array is a vector of integers that stores the enumeration values of curve types. Curve Type Array is encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Table 194 — JT STT Analytical Curve Types

0	LINE
1	CIRCLE
2	ELLIPSE

The detailed definitions of the Curve Types can be found: F.2.4.2.2 Line; F.2.4.2.3 CIRCLE and F.2.4.2.4 ELLIPSE.

VecF64: Coordinate Array

Coordinate Array contains an array of double precision floating point numbers that represent the collection of point coordinate information in the definition of the curve entities. The composite type VecF64 is defined in the Symbols table of Notational Conventions.

VecF64: Axis Array

Axis Array contains an array of double precision floating point numbers that represent the collection of unit vector information in the definition of the curve entities. The composite type VecF64 is defined in the Symbols table found in Notational Conventions.

VecF64: Radius Array

Radius Array contains an array of double precision floating point numbers that represent the collection of radius information in the definition of the curve surface. The composite type VecF64 is defined in in the Symbols table found in Notational Conventions.

VecF64: Parametric Domain Array

Parametric Domain Array contains an array of double precision floating point numbers that represent the collection of parametric domain information in the definition of the curve entities. The composite type VecF64 is defined in in the Symbols table found in Notational Conventions.

Recover Curve Geometry

The logic diagram to recover curve geometry from the arrays is shown below. All the represented curves are processed one by one sequentially, with its definition recovered from relevant arrays. In the diagram, vCoord represents “coordinate array”, vAxis represents “axis array”, vRadius represents “radius array”, and vDomain represents “parameter domain array”.

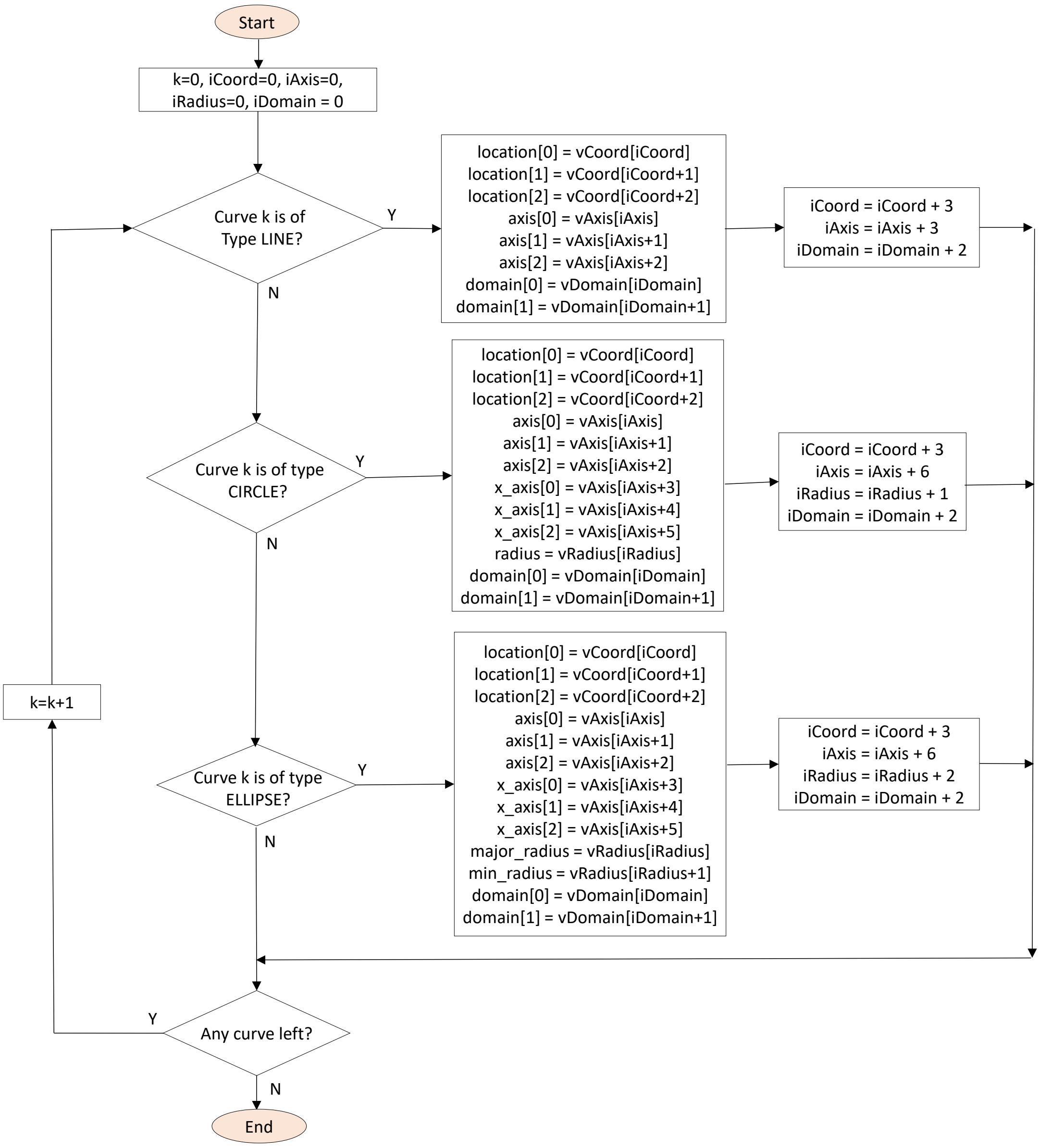


Figure 221 — Curve Geometry Recovery

Point Geometric Data

Point Geometric Data defines a collection of vertex points.

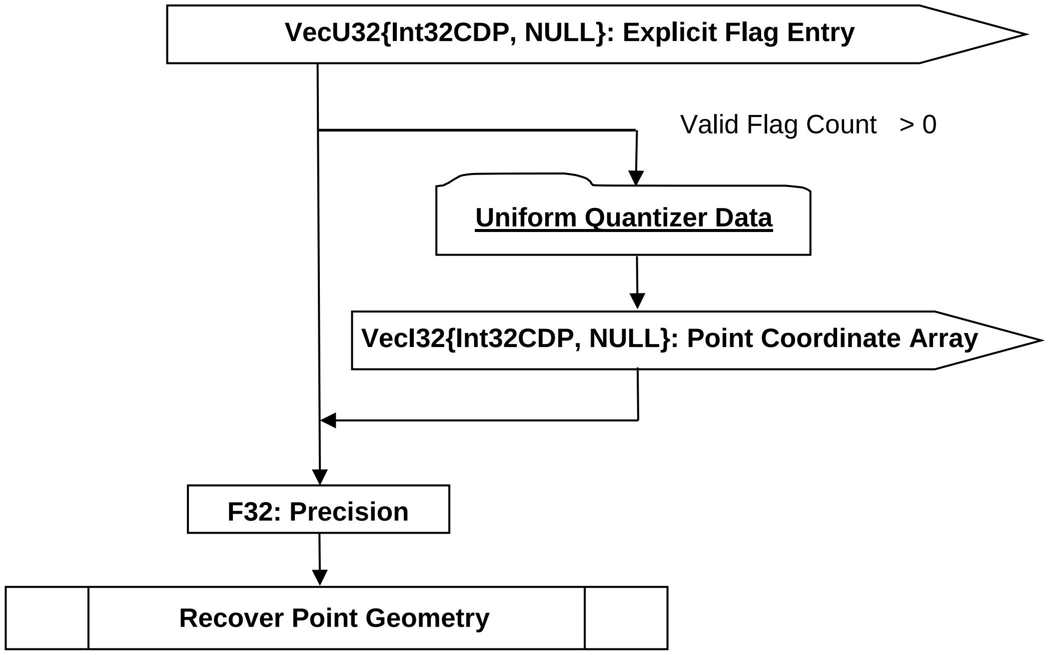


Figure 222 — Point Geometric Data collection

VecU32{Int32CDP, NULL}: Explicit Flag Entry

Explicit Flag Entry is a vector of flags that indicates if a particular point is explicitly written. A point may not be explicitly written if that point can be inferred from curve geometry, e.g., coincident with one of the end points of a represented curve. The flag value is 1 if the point geometry is explicitly written, and 0 otherwise. Explicit Flag Entry is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecF32{Int32CDP, NULL}: Point Coordinate Array

Point Coordinate Array contains explicit point geometry to write. Point Coordinate Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

F32: Precision

Precision of Point Coordinate Array.

Recover Point Geometry

The logic diagram to recover point geometry by evaluating represented curves is shown below.

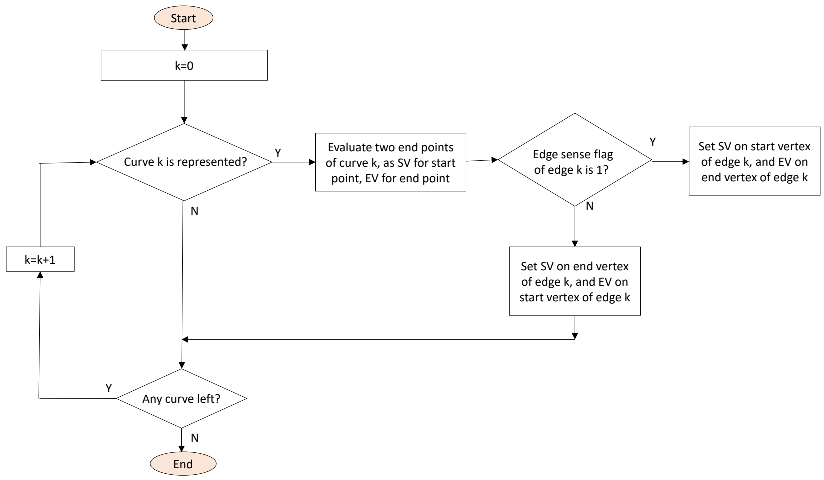


Figure 223 — Point Geometry Recovery

I32: Geometry Data Hash

The Geometry Data Hash is the combined hash of all the elements in Geometry Data. Refer to the Hashing Annex for a more detailed description on hashing.

H.1.3 Attribute Data

The Attribute Data is B-Rep attributes from XTBrep. Attributes contain Body Attribute, Face Attribute and Edge Attribute.

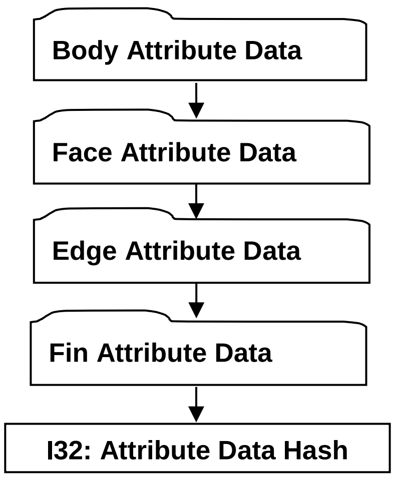


Figure 224 — Attribute Data Collection

Body Attribute Data

The Body Attribute Data includes Body Identifiers, Body Checksums,

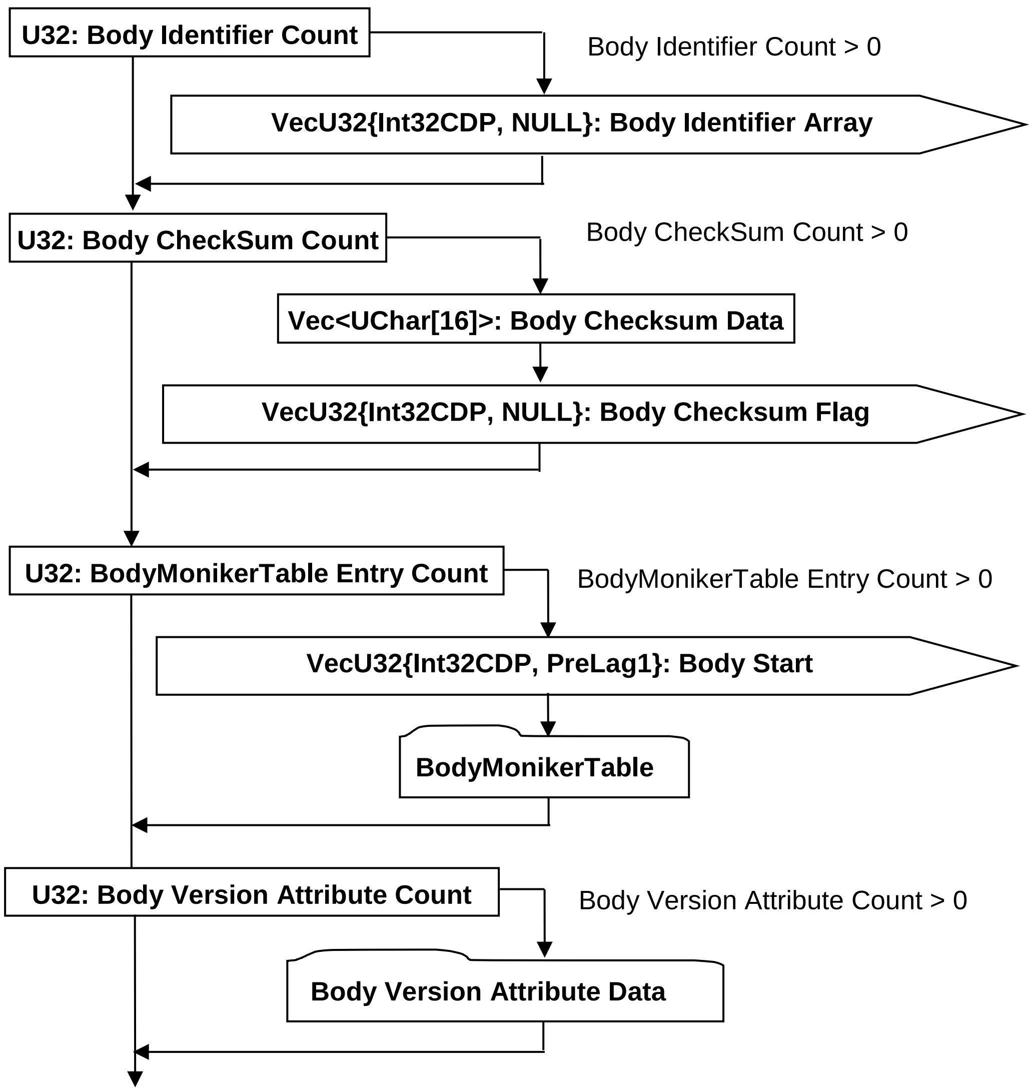


Figure 18 — Bodies Attribute Data collection

U32: Body Identifier Count

Body Identifier Count indicates the number of body identifiers. Its value is either the number of bodies or 0.

VecU32{Int32CDP, NULL}: Body Identifier Array

Body Identifier Array is a vector of integer attributes, with its length indicated by Body Identifier Count. When Body Identifier Array exists, i.e., Body Identifier Count is greater than 0, it is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: Body Checksum Count

Body Checksum Count indicates the number of body checksums. Its value is either the number of bodies or 0.

Vec<UChar[16]>: Body Checksum Data

Body Checksum Data is a vector of checksums, where each checksum, composed of 16 UChar values, represents the checksum of corresponding body in STT. The length of Body Checksum Data is indicated by Body Checksum Count.

VecU32{Int32CDP, NULL}: Body CheckSum Flag

Body CheckSum Flag is a vector of flags representing additional information about the Checksum value. It has the same length as Body Checksum Data, so that each checksum has an associated flag. The flag value can be either 0 or 1. Body Checksum Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, PreLag1}: Body Start Array

Body Start Array is a vector of indices representing the integer index value of start MonikerGuidTable entry for each body. Body Start Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: BodyMonikerTable Entry Count

BodyMonikerTable Entry Count indicates the number of entries in BodyMonikerTable Its value is either the number of bodies or 0.

BodyMonikerTable Data

BodyMonikerTable Data contains all the moniker entries for all the bodies in STT. The association between bodies and these entries are represented in Body Start Array. Each entry includes three pieces of information: an integer that identifies the GUID, the GUID representation, and a string that describes the source of the GUID.

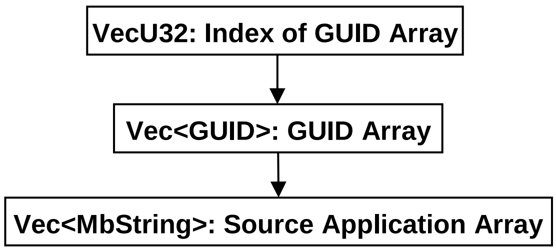


Figure 19 — MonikerGuidTable Data collection

VecU32: Index of GUID Array

Index of GUID Array is a vector of integers that collects all the identifier fields from all MonikerGuidTable entries.

Vec<GUID>: GUID Array

GUID Array is a vector of GUIDs that collects all the GUIDs from all MonikerGuidTable entries.

Vec<MbString>: Source Application Array

Source Application Array is a vector of strings that collects all the source strings from all MonikerGuidTable entries.

U32: Body Version Attribute Count

Body Version Attribute Count indicates the number of body Version attributes. Its value is either the number of bodies or 0.

Body Version Attribute Data

Body Version Attribute Data consists of Body Version ID and Source Application descriptions.

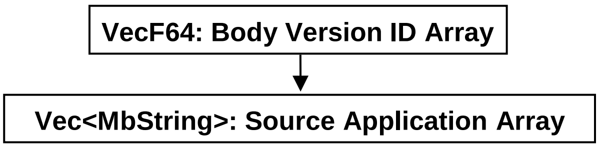


Figure 20 — Body ID Attribute Data

VecF64: Body Version ID

Body Version ID Array is a vector of Float64 numbers, each of which describes version information for a body in STT.

Vec<MbString>: Source Application Array

Source Application Array is a vector of strings, each of which describes the source application that is associated with the version information.

Face Attribute Data

The Face Attribute Data describes attribute information for all the Faces in STT.

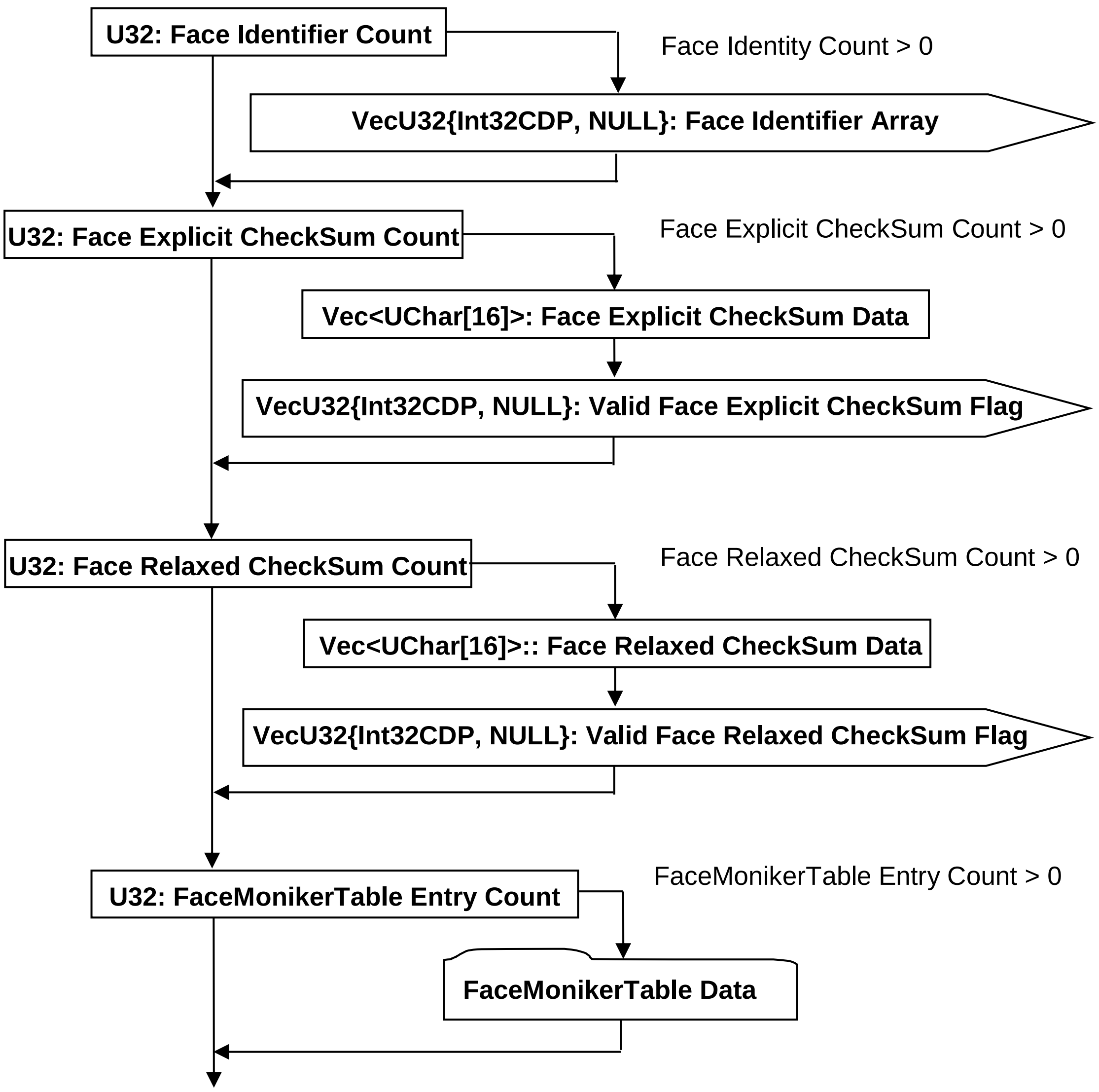


Figure 225 — Face Attribute Data collection

U32: Face Identifier Count

Face Identifier Count indicates the number of face identifiers. Its value is either the number of faces or 0.

VecU32{Int32CDP, NULL}: Face Identifier Array

Face Identifier Array is a vector of integers that provides face identification. Face Identifier Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: Face Explicit CheckSum Count

Face Explicit CheckSum Count indicates the number of Face Explicit CheckSums. Its value is either the number of faces or 0.

Vec<UChar[16]>: Face Explicit CheckSum Data

Face Explicit CheckSum Data describes Explicit CheckSum for all the faces, where each CheckSum is composed of 16 UChar values.

VecU32{Int32CDP, NULL}: Valid Face Explicit CheckSum Flag

Valid Face Explicit CheckSum Flag is a vector of flags representing if the corresponding CheckSum in Face Explicit CheckSum Data is valid. Valid Face Explicit CheckSum Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: Face Relaxed CheckSum Count

Face Relaxed CheckSum Count indicates the number of face Relaxed CheckSums. Its value is either the number of faces or 0.

UChar: Face Relaxed CheckSum Data

Face Relaxed CheckSum Data describes Relaxed CheckSum for all the faces, where each CheckSum is composed of 16 UChar values.

VecU32{Int32CDP, NULL}: Valid Face Relaxed CheckSum Flag

Valid Face Relaxed CheckSum Flag is a vector of flags representing if the corresponding CheckSum is valid. Valid Face Relaxed CheckSum Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: FaceMonikerTable Entry Count

FaceMonikerTable Entry Count indicates the number of entries in FaceMonikerTable Its value is either the number of faces in STT or 0.

Face Moniker Table Data

FaceMonikerTable Data includes a vector of entries, each of which describes moniker attribute for a face in STT. The description of each face includes three fields: a string label, an integer describing the index of GUID, and an integer that provides identification. Each field is grouped into a separate array for serialization.

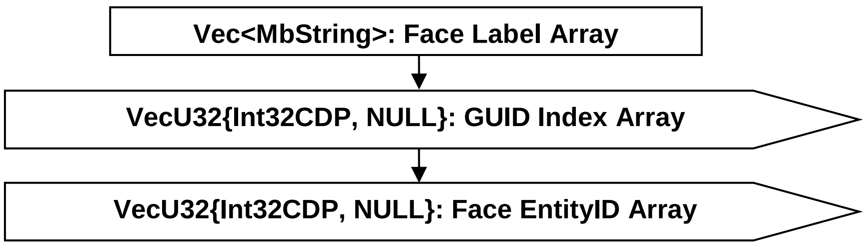


Figure 226 — MONIKER/MONIKER_DATA_ATTRIB Data collection

Vec<MbString>: Face Label Array

Face Label Array is a vector of strings that provides label description for each Face.

VecU32{Int32CDP, NULL}: GUID Index Array

GUID Index Array is a vector of integers that provides GUID index into the GUID table associated with the body in which the face is part of. GUID Index Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Face Identity Array

Face Identity Array is a vector of integers that provides face identification. Face Identity Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Edge Attribute Data

The Edge Attribute Data contains attribute information for edges in STT.

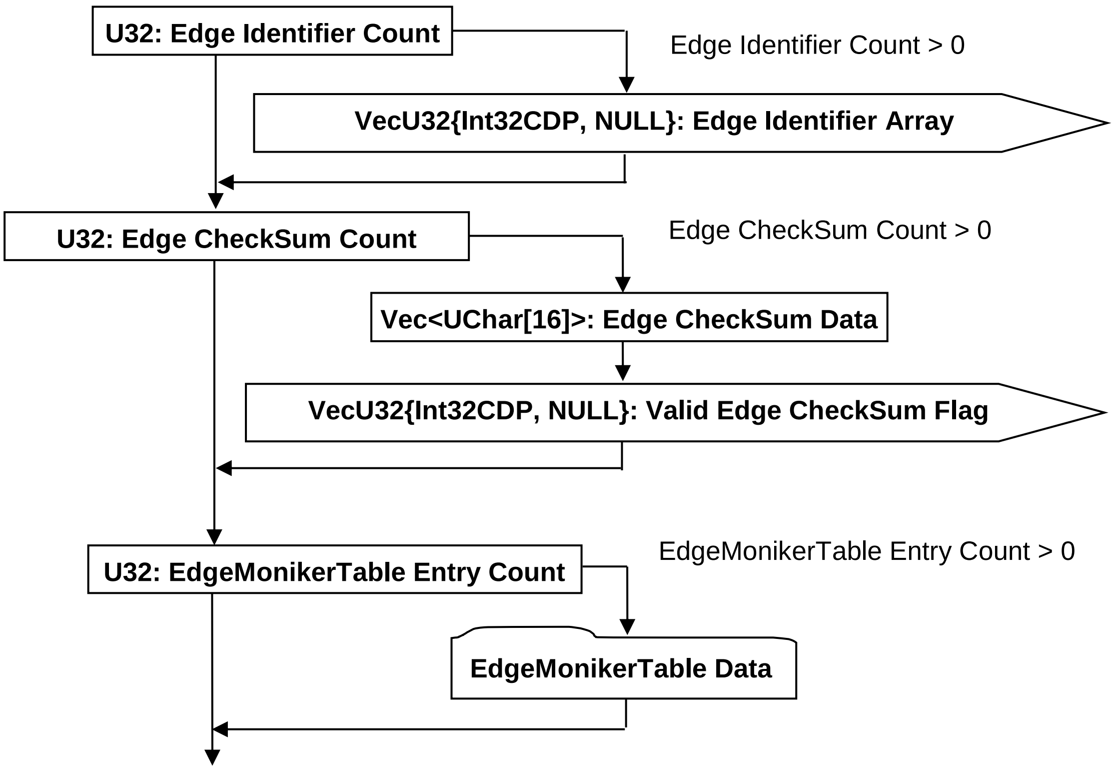


Figure 227 — Edge Attribute Data collection

U32: Edge Identifier Count

Edge Identifier Count indicates the number of Edge identifiers. Its value is either the number of Edges in STT or 0.

VecU32{Int32CDP, NULL}: Edge Identifier Array

Edge Identifier Array is a vector of integer Edge identifiers. Edge Identifier Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: Edge CheckSum Count

Edge CheckSum Count indicates the number of Edge CheckSums. Its value is either the number of Edges or 0.

Vec<UChar[16]>: Edge CheckSum Data

Edge CheckSum Data is a vector of checksum values for all the edges, where each checksum is composed of a 16 UChar values.

VecU32{Int32CDP, NULL}: Valid Edge CheckSum Flag

Valid Edge CheckSum Flag is a vector of integer flags representing whether the corresponding edge CheckSum is valid. Valid Edge CheckSum Flag is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

U32: EdgeMonikerTable Entry Count

EdgeMonikerTable Entry Count indicates the number of edge moniker entries. Its value is either the number of edges or 0.

Edge Moniker Table Data

Edge MonikerTable Data describes moniker attribute information for all the edges. The moniker attribute information includes three fields: Edge Label string, GUID index, and integer identifier. Each field is grouped into separate arrays.

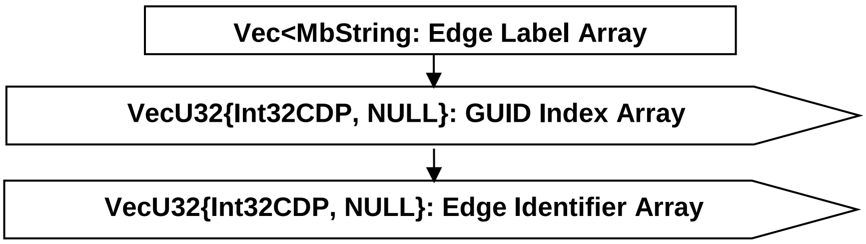


Figure 228 — EdgeMonikerTable Data collection

Vec<MbString>: Edge Label Array

Edge Label Array is a vector of strings describing the Edge Label for all the edges.

VecU32{Int32CDP, NULL}: GUID Index Array

GUID Index Array is a vector of integers that provides GUID index into the GUID table associated with the body in which the Edge is part of. GUID Index Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

VecU32{Int32CDP, NULL}: Edge Identifier Array

Edge Identifier Array is a vector of integers that provides Edge identification. Edge Identity Array is compressed and encoded using the Int32CDP CODEC described in Int32 Compressed Data Packet.

Fin Attribute Data

The Fin Attribute Data describes information associated with the fins in STT.

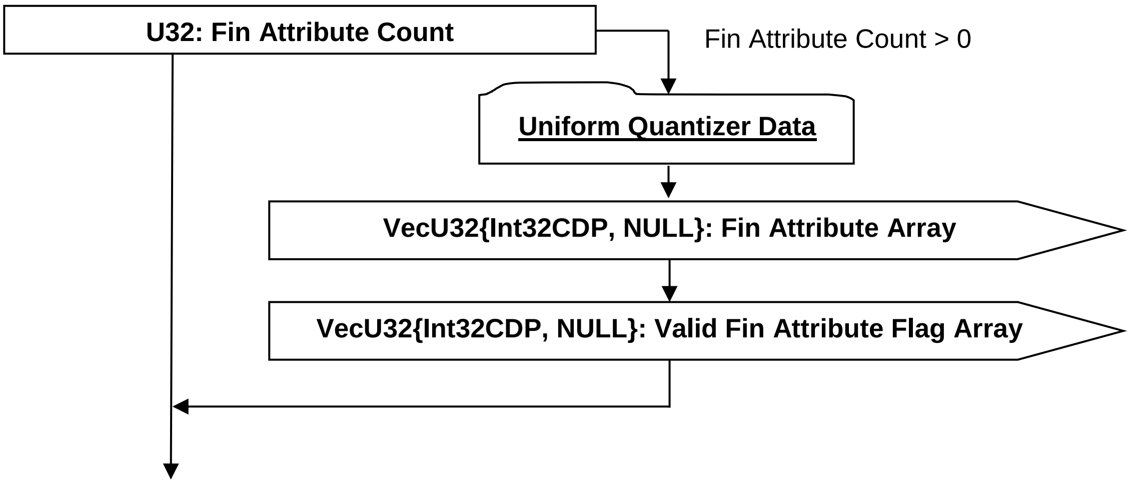


Figure 25 — Fin Attribute Data collection

U32: Fin Attribute Count

Fin Attribute Count indicates the number of Fin Attributes. Its value is either the number of Fins or 0.

VecU32{Int32CDP, NULL}: Fin Attribute Array

Fin Attribute Data is a vector of integers for all the fins, where each integer represents a quantized value. There are two integers for each fin, with the first integer describing the normalized u parameter of the middle point on the Fin and second integer describing the normalized v parameter of the middle point on the Fin. These two integers of the same fin are continuous in the vector, and therefore the length of Fin Attribute Array is twice the number of fins in STT. Fin Attribute Array uses the Int32CDP CODEC Int32 Compressed Data Packet to compress and encode data.

VecU32{Int32CDP, NULL}: Valid Fin Attribute Flag Array

Valid Fin Attribute Flag is a vector of integer flags representing whether each corresponding Fin attribute, composed of two integer numbers in Fin Attribute Array, is valid. The corresponding Fin attribute is invalid if the flag has value 0 and valid otherwise.

Version Data

Version data is informative. It describes additional version information related to STT.

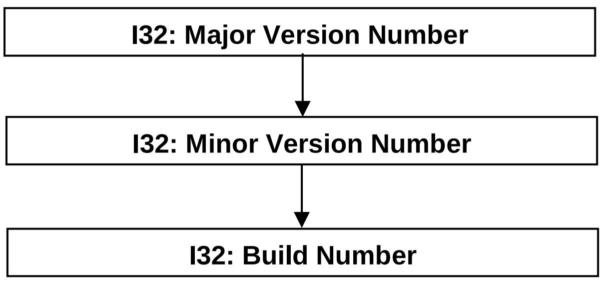


Figure 26 — Version Data

I32: Major Version Number

Major Version Number specifies the major version number for the STT data in the JT File. Major version number is an informative field which can be set to 0 without negative impact to implementation.

I32: Minor Version Number

Minor Version Number specifies the minor version number for the STT data in the JT File. Minor version number is an informative field which can be set to 0 without negative impact to implementation.

I32: Build Number

Build Number specifies the build number for the STT data in the JT File. Build number is an informative field which can be set to 0 without negative impact to implementation.

I32: Attribute Data Hash

The Attribute Data Hash is the combined hash of all the data in Attribute Data. Refer to the Hashing Annex for a more detailed description on hashing.

Annex I
JT LWPA Segment

JT LWPA Segment contains an Element that defines light weight precise analytic data for a particular part. More specifically LWPA contains the collection of analytic surfaces in the B-Rep definition of the part.

JT LWPA Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT LWPA Segment type supports LZMA compression on all element data, so all elements in JT LWPA Segment use the Logical Element Header Compressed form of element header data.

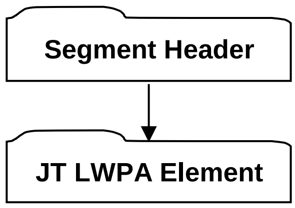


Figure 229 — JT LWPA data collection

Complete description for Segment Header can be found in the File Header section of Base Format Description under Data Segment.

I.1 JT LWPA Element

Object Type ID: 0xd67f8ea8, 0xf524, 0x4879, 0x92, 0x8c, 0x4c, 0x3a, 0x56, 0x1f, 0xb9, 0x3a

JT LWPA Segment represents a particular Part's precise analytic surfaces. It can be viewed as a subset of B-Rep representation where the subset refers to the complete collection of all the surfaces that are of one of the analytic types shown in the Supported Surface Type table, i.e., plane, cylinder, cone, sphere, or torus. Unlike JT B-Rep Element or XT B-Rep Element, JT LWPA Element does not contain any B-Rep topology information, nor does it contain geometric curve or point information. LWPA is designed to represent most essential part geometry information with much lighter weight on disk and much faster to load than B-Rep. Typically LWPA is less than 2 percent of B-Rep size on disk, and takes less than 5 percent time to load into memory. The analytic representation of LWPA follows XT convention as detailed in the XT B-Rep geometry Annex.

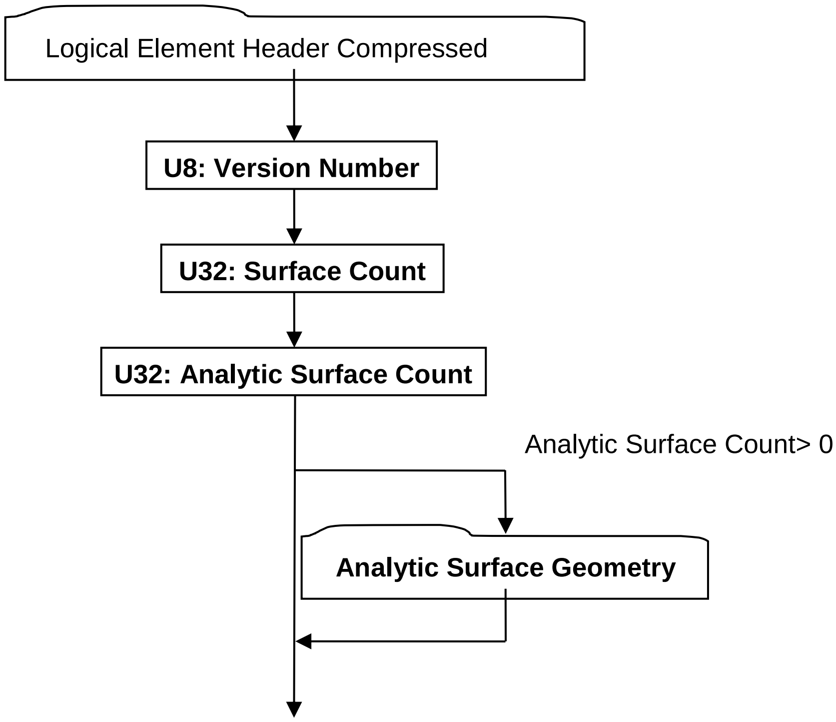


Figure 230 — JT LWPA Element data collection

U8: Version Number

Version Number is the version identifier for this JT LWPA Element. Information on local version numbers can be found in the Base Format Description under Common Data Conventions and Constructs Local version numbers.

U32: Surface Count

Surface Count indicates the number of surface entries in LWPA. The number of surface entries is equal to the number of surfaces in the B-Rep representation. The surface entry does not contain any information if the corresponding B-Rep surface is not of analytic type.

U32: Analytic Surface Count

Analytic Surface Count indicates the number of analytic surface entries in LWPA.

Analytic Surface Geometry

Analytic Surface Geometry defines a collection of analytic surfaces and their mapping to the original B-Rep surfaces.

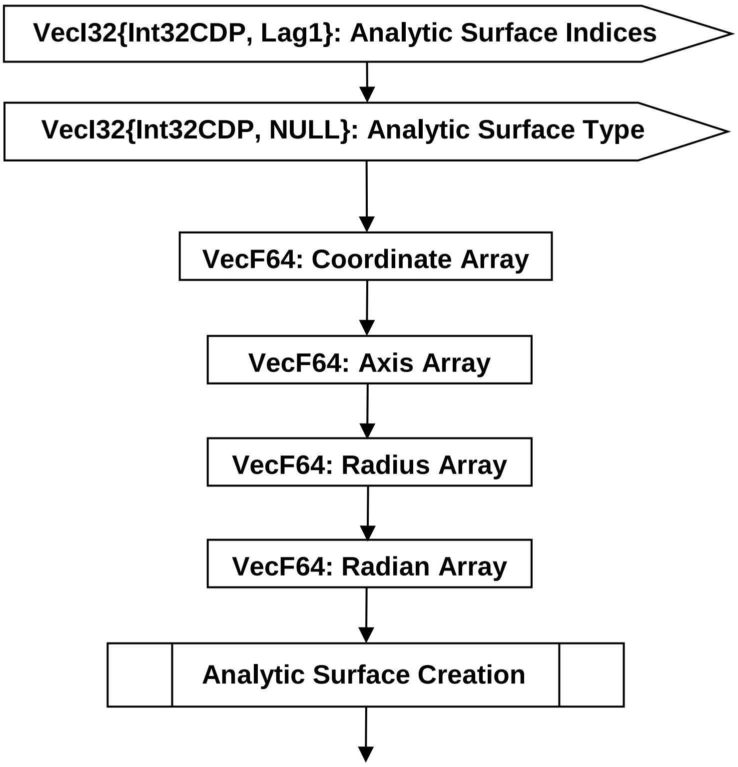


Figure 231 — Analytic Surface Geometry data collection

VecI32{Int32CDP, Lag1}: Analytic Surface Indices

Analytic Surface Indices is an integer array with its length equal to the number of analytic surfaces in the LWPA. The value of each element in this array represents the index of this analytic surface in the original B-Rep representation.

VecI32{Int32CDP, NULL}: Analytic Surface Type

Analytic Surface Type is an integer array with its length equal to the number of analytic surfaces in the LWPA. The value of each element in this array represents the type of each analytic surface, as defined in table Supported Surface Type.

VecF64: Coordinate Array

Coordinate Array contains an array of double precision floating point numbers that represent the collection of point coordinate information in the definition of the analytic surface entities. The composite type VecF64 is defined in the Symbols table of Notational Conventions. Each floating point number in the array is written in binary form.

VecF64: Axis Array

Axis Array contains an array of double precision floating point numbers that represent the collection of unit vector information in the definition of the analytic surface entities. The composite type VecF64 is defined in the Symbols table found in Notational Conventions. Each floating point number in the array is written in binary form.

VecF64: Radius Array

Radius Array contains an array of double precision floating point numbers that represent the collection of radius information in the definition of the analytic surface entities. The composite type VecF64 is defined in in the Symbols table found in Notational Conventions. Each floating point number in the array is written in binary form.

VecF64: Radian Array

Radian Array contains an array of double precision floating point numbers that represent the collection of radian information in the definition of the analytic surface entities. The composite type VecF64 is defined in in the Symbols table found in Notational Conventions. Each floating point number in the array is written in binary form.

Analytic Surface Creation

Analytic surfaces in LWPA are constructed based on the information of the above arrays, as illustrated by logical diagram in the Figure 233 - Analytic Surface Creation.

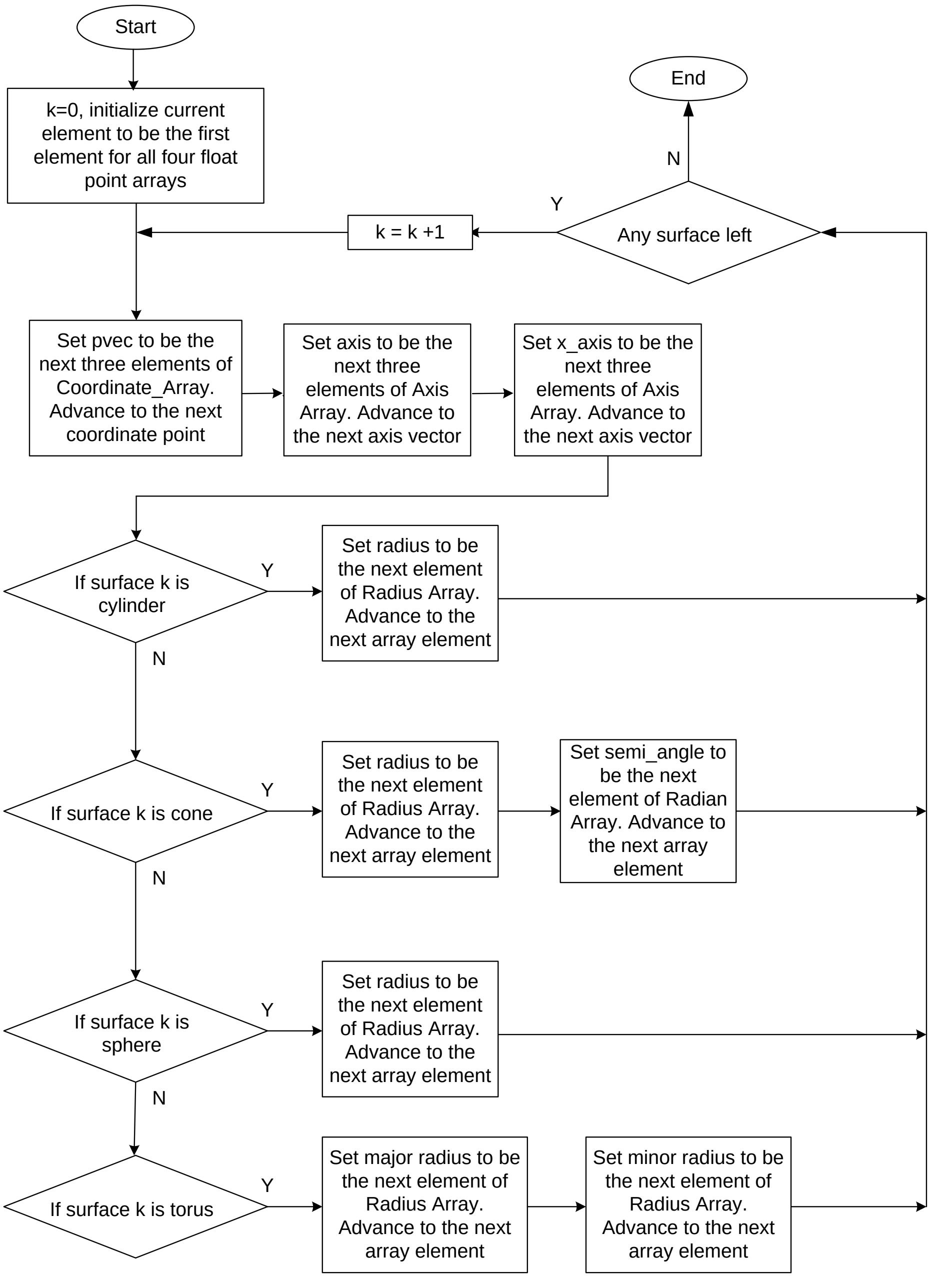


Figure 232 — Analytic Surface Creation

Annex J
Wireframe Segment

Wireframe Segment contains an Element that defines the precise 3D wireframe data for a particular Part. A Wireframe Segment is typically referenced by a Part Node Element (see 6.1.1.5 Part Node Element) using a Second specifies the date Second value. Valid values are [0, 59] inclusive.

Late Loaded Property Atom Element (see Late Loaded Property Atom Element). The Wireframe Segment type supports LZMA compression on all element data, so all elements in Wireframe Segment use the Logical Element Header Compressed form of element header data.

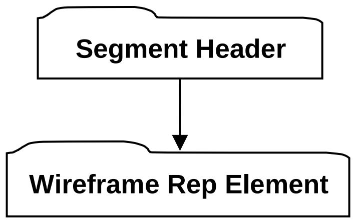


Figure 233 — Wireframe Segment data collection

Complete description for Segment Header can be found in the File Header section of Base Format Description under Data Segment.

J.1 Wireframe Rep Element

Object Type ID: 0x873a70d0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

A Wireframe Rep Element represents a particular Part's precise 3D wireframe data (e.g. reference curves, section curves). Much of the "heavyweight" data contained within a Wireframe Rep Element is compressed and/or encoded. The compression and/or encoding state is indicated through other data stored in each Wireframe Rep Element.

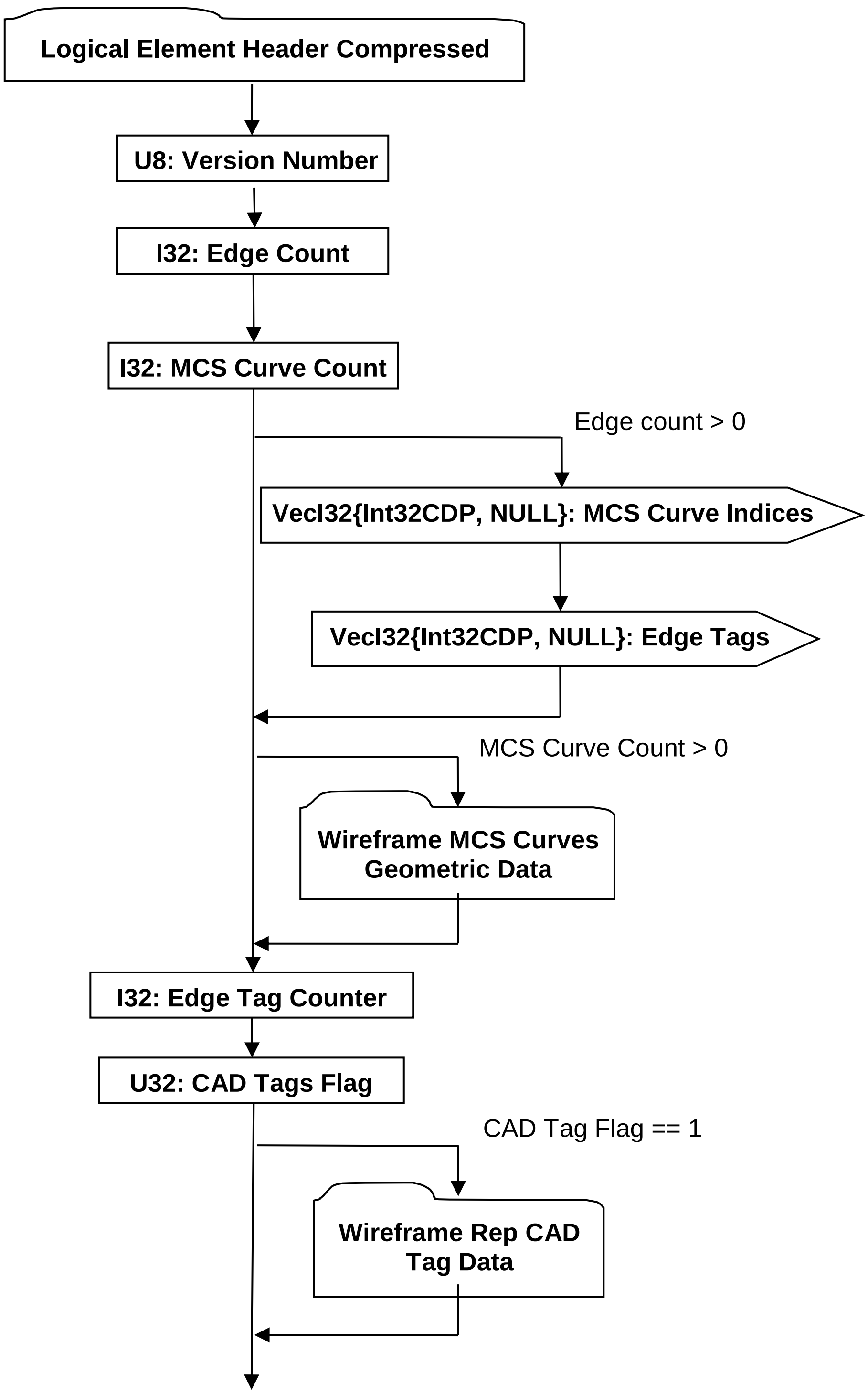


Figure 234 — Wireframe Rep Element data collection

Complete description for Logical Element Header Compressed can be found in the File Header section of Base Format Description under Data Segment, Data.

U8: Version Number

Version Number is the version identifier for this JT Wireframe Rep Element. Information on local version numbers can be found in the Base Format Description under Common Data Conventions and Constructs Local version numbers.

I32: Edge Count

Edge Count indicates the number of topological Edge entities in the Wireframe Rep.

I32: MCS Curve Count

MCS Curve Count indicates the number of distinct geometric (Model Coordinate Space) curves (i.e. XYZ curve) entities in the Wireframe Rep.

VecI32{Int32CDP, NULL}: MCS Curve Indices

MCS Curve Indices is a vector of indices representing the index of the MCS Curve (Model Space curve) for each Edge. MCS Curve Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: Edge Tags

Each Edge has an identifier Tag. Edge Tags is a vector of identifier Tags for a set of Edges. Edge Tags uses the Int32 version of the CODEC to compress and encode data.

I32: Edge Tag Counter

Edge Tag Counter specifies the next available “unique’ tag value for Edge entity.

U32: CAD Tags Flag

CAD Tags Flag is a flag indicating whether CAD Tag data exist for the Wireframe Rep.

Wireframe MCS Curves Geometric Data

Wireframe MCS Curves Geometric Datacollection contains the Wireframe Rep’s Model Coordinate System geometric Curve data (i.e. XYZ Curve data). Currently only NURBS Curve types are supported within a Wireframe Rep. The count/number of MCS Curves within a Wireframe Rep is indicated by data field MCS Curve Count documented in Wireframe Rep Element.

Figure 235 — Wireframe MCS Curves Geometric Data collection

Complete description for Compressed Curve Data can be found in Compressed Curve Data.

Wireframe Rep CAD Tag Data

The Wireframe Rep CAD Tag Data collection contains the list of persistent IDs, as defined in the CAD System, to uniquely identify individual Edges in the Wireframe Rep. The existence of this Wireframe Rep CAD Tag Data collection is dependent upon the value of previously read data field CAD Tags Flag as documented in Wireframe Rep Element.

If Wireframe Rep CAD Tag Data collection is present, there will be a CAD Tag for every Edge in the Wireframe Rep. Therefore the total number of CAD Tags in the list should be equal to “Edge Count” as documented in Wireframe Rep Element.

Figure 236 — Wireframe Rep CAD Tag Data collection

Complete description for Compressed CAD Tag Data can be found in Compressed CAD Tag Data.

Annex K
(deprecated)
JT B-Rep Segment

JT B-Rep Segment contains an Element that defines the precise geometric Boundary Representation data for a particular Part in JT B-Rep format. The JT B-Rep segment specification is deprecated with JT IAP V2. The description is provided here to support legacy JT content.

JT B-Rep Segments are typically referenced by Part Node Elements (see Part Node Element) using Late Loaded Property Atom Elements (see Late Loaded Property Atom Element). The JT B-Rep Segment type supports compression on all element data, so all elements in JT B-Rep Segment use the Logical Element Header Compressed form of element header data.

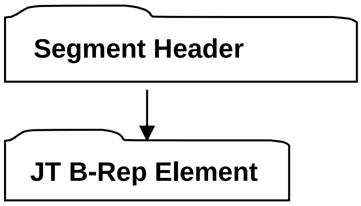


Figure 237 — JT B-Rep Segment data collection

A Complete description for Segment Header can be found in the File Header section of Base Format Description under Data Segment.

K.1 JT B-Rep Element

Object Type ID: 0x873a70c0, 0x2ac8, 0x11d1, 0x9b, 0x6b, 0x00, 0x80, 0xc7, 0xbb, 0x59, 0x97

JT B-Rep Element represents a particular Part's precise data in JT boundary representation format. Much of the "heavyweight" data contained within a JT B-Rep Element is compressed and/or encoded. The compression and/or encoding state is indicated through other data stored in each JT B-Rep Element.

Two important aspects of a Part are its geometry and its topology. The geometry describes the shape of a Part: this Surface is a plane, that Surface is a cylinder, this Curve is an arc, etc. The topology describes the connectivity of the Part: this Point is inside the Part, these Surfaces are next to each other, etc. The 0, 1, and 2 dimensional building blocks of geometry are Points, Curves, and Surfaces. The corresponding topological building blocks are Vertices, Edges, and Faces. Topology also uses Shells and Regions to conceptually divide up the three dimensional space.

Parts may have the same topology, but wildly different geometry. Imagine the Surfaces of a Part being composed of rubber. The topology of the Part does not change as we deform the Part by bending or stretching the surfaces, as long as we do not cut or glue them (we call this a "nice" deformation). A Part's topology can be classified as being "manifold" or "non-manifold"; where "manifold" implies that the Part has the property that each Edge, excluding seams and poles, has exactly two faces using it.

Similarly, Parts may have nearly identical geometry but different topology. The topology of a Part depends on how the geometry is put together. A Part may be manifold or non-manifold simply depending on how the geometry is put together. In addition to describing connectivity in space, topology is used to describe areas of interest (active areas) on Surfaces. These active Surface areas are used in defining a complex Part. The areas are specified by oriented Loops and often referred to as trimmed Surfaces which are exactly the 2-dimensional topological building block called a Face.

Readers desiring/needing a more in-depth exploration of boundary representation theory in order to understand the significance/meaning of some of the JT B-Rep data fields are referred to references [22] and [23] listed in the bibliography section of this specification.

Since the topology is a convenient way to describe or organize the Part, it is also convenient to store the geometry of the Part in the topological structures. The following sub-sections document the JT B-Rep format for storing the topology and geometry of a Part in a JT file.

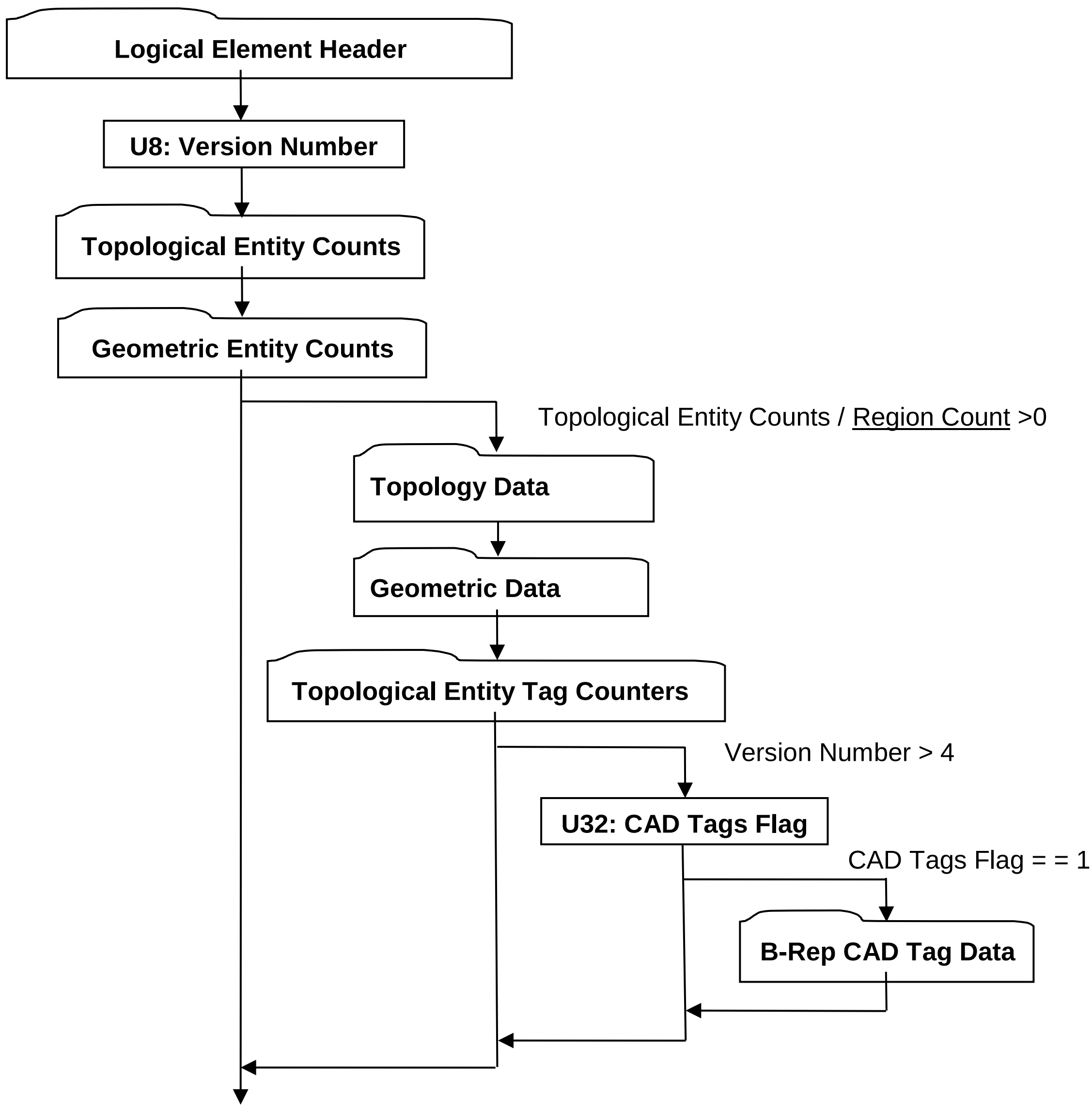


Figure 238 — JT B-Rep Element data collection

Complete description for Logical Element Header Compressed can be found in the File Header section of Base Format Description under Data Segment, Data.

U8: Version Number

Version Number is the version identifier for this JT B-Rep Element. Information on local version numbers can be found in the Base Format Description under Common Data Conventions and Constructs Local version numbers.

U32: CAD Tags Flag

CAD Tags Flag is a flag indicating whether CAD Tag data exist for the JT B-Rep.

K.1.1 Topological Entity Counts

Topological Entity Counts data collection defines the counts for each of the various topological entities within a B-Rep.

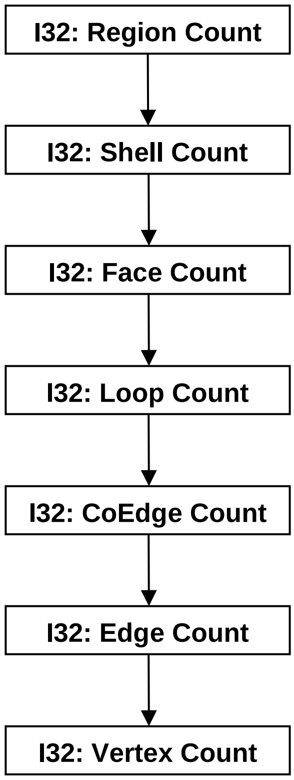


Figure 239 — Topological Entity Counts data collection

I32: Region Count

Region Count indicates the number of topological region entities in the B-Rep.

I32: Shell Count

Shell Count indicates the number of topological shell entities in the B-Rep.

I32: Face Count

Face Count indicates the number of topological face entities in the B-Rep.

I32: Loop Count

Loop Count indicates the number of topological loop entities in the B-Rep.

I32: CoEdge Count

CoEdge Count indicates the number of topological coedge entities in the B-Rep.

I32: Edge Count

Edge Count indicates the number of topological edge entities in the B-Rep.

I32: Vertex Count

Vertex Count indicates the number of topological vertex entities in the B-Rep.

K.1.2 Geometric Entity Counts

Geometric Entity Counts data collection defines the counts for each of the various geometric entities within a B-Rep.

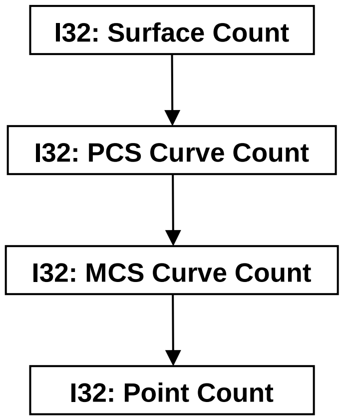


Figure 240 — Geometric Entity Counts data collection

I32: Surface Count

Surface Count indicates the number of distinct geometric surface entities in the B-Rep.

I32: PCS Curve Count

PCS Curve Count indicates the number of distinct geometric Parameter Coordinate Space curves (i.e. UV curve) entities in the B-Rep.

I32: MCS Curve Count

MCS Curve Count indicates the number of distinct geometric (Model Coordinate Space) curves (i.e. XYZ curve) entities in the B-Rep.

I32: Point Count

Point Count indicates the number of distinct geometric point entities in the B-Rep.

K.1.2.1 Topology Data

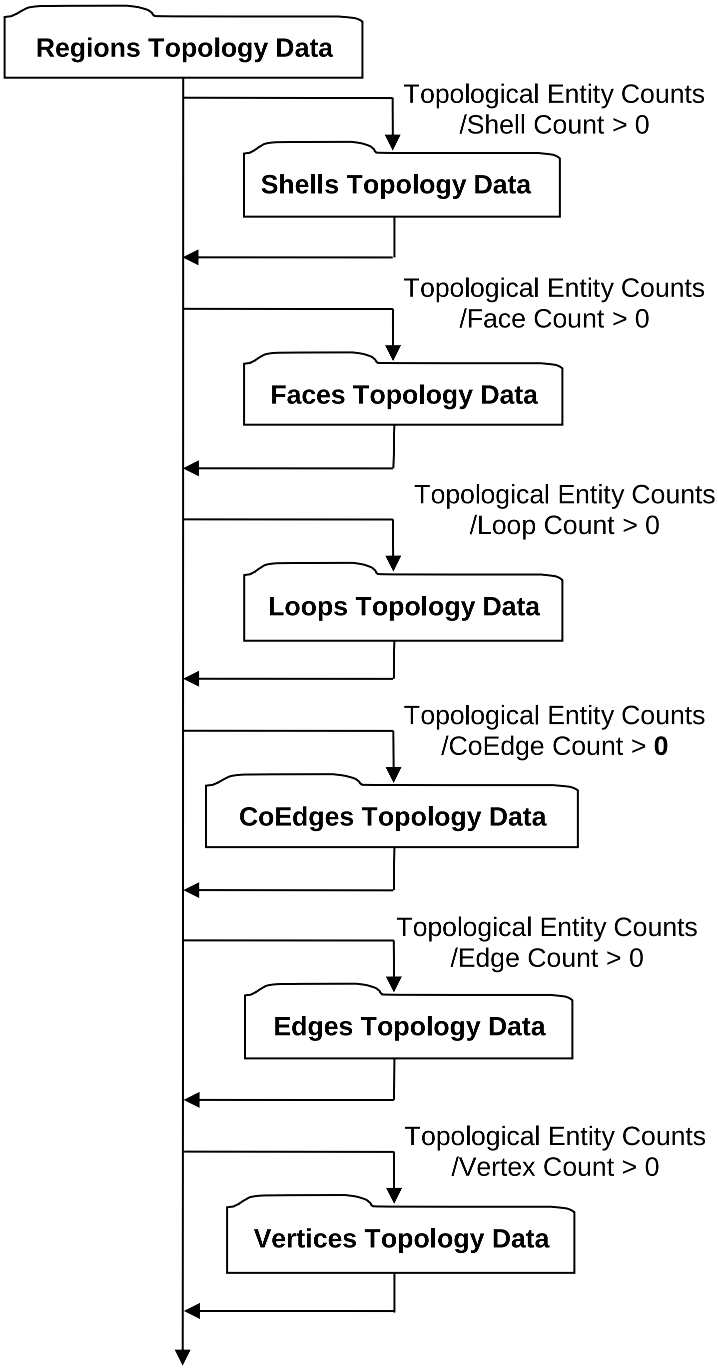


Figure 241 — Topology Data collection

K.1.2.1.1 Regions Topology Data

Regions Topology Data defines the set of non-overlapping Shells comprising each Region. The volume of a Region is that volume lying inside each “anti-hole Shell” and outside each simply-contained “hole Shell” belonging to the particular Region. A Region is analogous to a dimensionally elevated face where Region corresponds to Face and Shell corresponds to Trim Loop.

Each Region's defining Shells are identified in a list of Shells by an index for both the first Shell and the last Shell in each Region (i.e. all Shells inclusive between the specified first and last Shell list index define the particular Region).

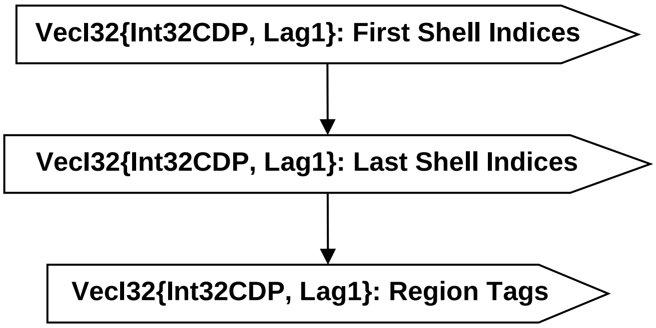


Figure 242 — Regions Topology Data collection

VecI32{Int32CDP, Lag1}: First Shell Indices

First Shell Indices is a vector of indices representing the index of the first Shell in each Region. First Shell Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Last Shell Indices

Last Shell Indices is a vector of indices representing the index of the last Shell in each Region. Last Shell Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Region Tags

Each Region has an identifier tag. Region Tags is a vector of identifier tags for a set of Regions. Region Tags uses the Int32 version of the CODEC to compress and encode data.

Shells Topology Data

Shells Topology Data defines the set of topological adjacent Faces making up each Shell. A Shell's set of topological adjacent Faces define a single (usually closed) two manifold solid that in turn defines the boundary between the finite volume of space enclosed within the Shell and the infinite volume of space outside the Shell. Additional, each Shell has a flag that denotes whether the Shell refers to the finite interior volume (i.e. a "hole Shell") or the infinite exterior volume (i.e. an "anti-hole Shell").

Each Shell's defining Faces are identified in a list of Faces by an index for both the first Face and the last Face in each Shell (i.e. all Faces inclusive between the specified first and last Face list index define the particular Shell).

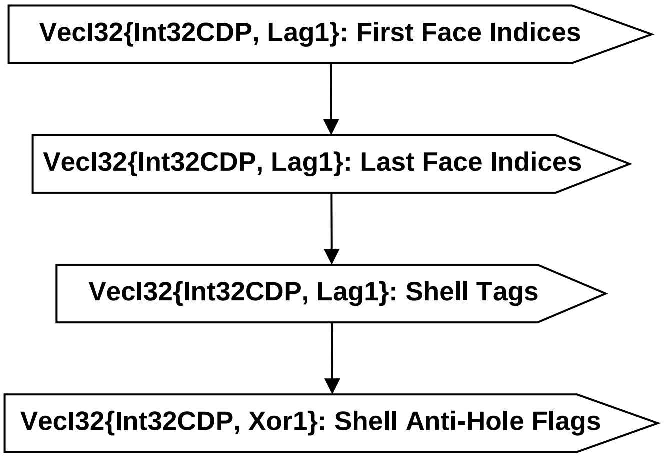


Figure 243 — Shells Topology Data collection

VecI32{Int32CDP, Lag1}: First Face Indices

First Face Indices is a vector of indices representing the index of the first Face in each Shell. First Face Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Last Face Indices

Last Face Indices is a vector of indices representing the index of the last Face in each Shell. Last Face Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Shell Tags

Each Shell has an identifier tag. Shell Tags is a vector of identifier tags for a set of Shells. Shell Tags uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Xor1}: Shell Anti-Hole Flags

Each Shell has a flag identifying whether the Shell is an anti-hole Shell. Shell Anti-Hole Flags is a vector of anti-hole flags for a set of Shells.

In an uncompressed/decoded form the flag values have the following meaning:

Table 195 — JT B-Rep Shell Topology Anti-Hole Flag values

= 0	Shell is not an anti-hole Shell
= 1	Shell is an anti-hole Shell

Shell Anti-Hole Flags uses the Int32 version of the CODEC to compress and encode data.

Faces Topology Data

A Face is a two-dimensional topological building block defined as the active (that portion to be used in the model) regions/areas of a Geometric Surface; where active regions/areas of a Geometric Surface are indicated using oriented Trim Loops. Faces Topology Data specifies the underlying Geometric Surface and Trim Loops making up each Face along with a “reverse normal” flag and identifier tag for each Face.

A Face shall be trimmed with at least one “anti-hole” Trim Loop and zero or more “hole” Trim Loops. Thus the area of the Geometric Surface defined as the Face, is the area inside the “anti-hole” Trim Loops and outside each “hole” Trim Loop. No Trim Loops (“hole’ or “anti-hole”) may intersect/cross or be tangent at any point. “Anti-Hole” Trim Loops shall be defined with a counter-clockwise orientation in the underlying surface's parameter space whereas “hole” Trim Loops shall be defined with a clockwise orientation. With this Trim Loop orientation definition, as one traverses a Trim Loop of a Face, the material or “active region” is always to one’s left. The figure below gives an example in parameter space of proper trim loop definition and orientation (as indicated by the arrows on the loop’s CoEdges) for a face with two holes. “L1” represents the face “anti-hole” Trim Loop while “L2” and L3” represent the two “hole” Trim Loops. Note that each hole is always represented by a separate distinct “hole” Trim Loop.

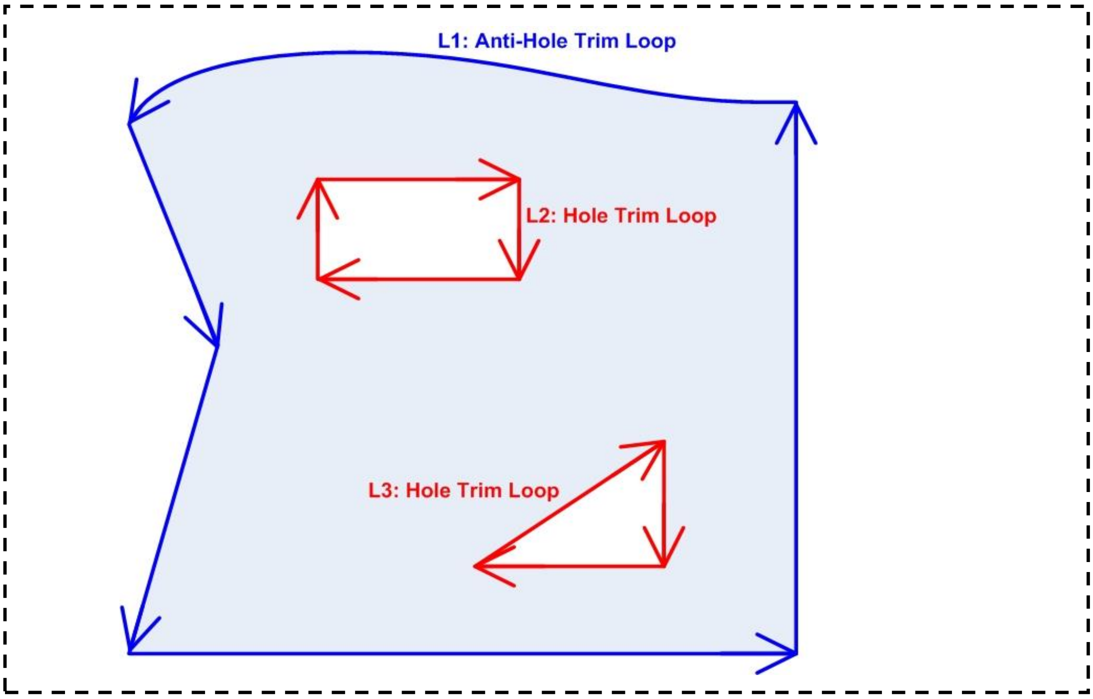


Figure 244 — Trim Loop example in parameter Space - One Face with 2 Holes

Each Face’s underlying Geometric Surface is identified by an index into a list of Geometric Surfaces. Each Face’s defining Trim Loops are identified in a list of trim Loops by an index for both the first Trim Loop and the last Trim Loop in each Face (i.e. all Trim Loops inclusive between the specified first and last Trim Loop list index define the particular Face).

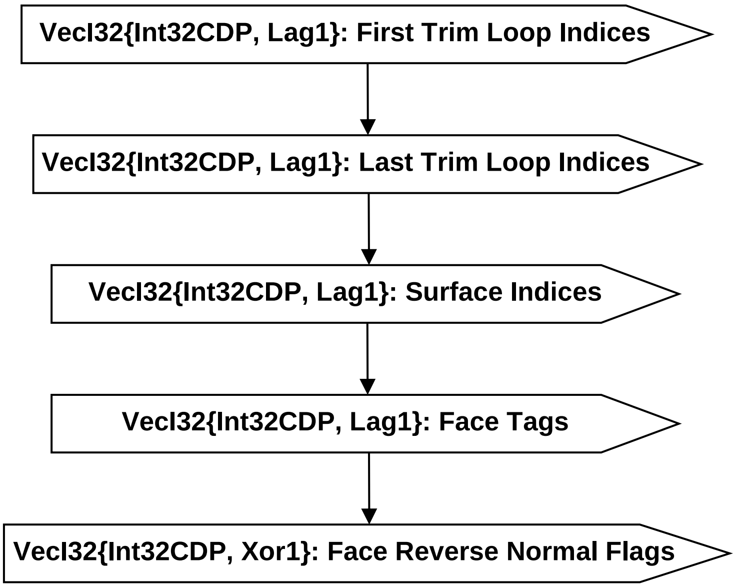


Figure 245 — Faces Topology Data collection

VecI32{Int32CDP, Lag1}: First Trim Loop Indices

First Trim Loop Indices is a vector of indices representing the index of the first Trim Loop in each Face. First Trim Loop Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Last Trim Loop Indices

Last Trim Loop Indices is a vector of indices representing the index of the last Trim Loop in each Face. Last Trim Loop Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Surface Indices

Surface Indices is a vector of indices representing the index of the underlying Geometric Surface for each Face. Surface Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Face Tags

Each Face has an identifier tag. Face Tags is a vector of identifier tags for a set of Faces. Face Tags uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Xor1}: Face Reverse Normal Flags

Each Face has a flag identifying whether the Face's normal(s) should be interpreted to point in the direction opposite of the usual U cross V normal (note that these flags do not imply any sort of parameter reversal, the flag only implies that the material is on the other side of the surface).

Face Reverse Normal Flags is a vector of reverse-normal flags for a set of Faces.

In an uncompressed/decoded form the flag values have the following meaning:

Table 196 — JT B-Rep Face Reverse Normal Flag values

= 0	Face normal is not reversed
= 1	Face normal is reversed.

Face Reverse Normal Flags uses the Int32 version of the CODEC to compress and encode data.

Loops Topology Data

A Loop (often called Trimming Loop) defines in parameter space a 1D boundary around which geometric surfaces are trimmed to form a Face. Loops Topology Data specifies the CoEdges making up each Loop along with an anti-hole flag and identifier tag for each Loop.

A Loop is composed of one or more CoEdges and the Loop shall be closed and non-self-intersecting.

Each Loop's defining CoEdges are identified in a list of CoEdges by an index for both the first CoEdge and the last CoEdge in each Loop (i.e. all CoEdges inclusive between the specified first and last CoEdge list index define the particular Loop).

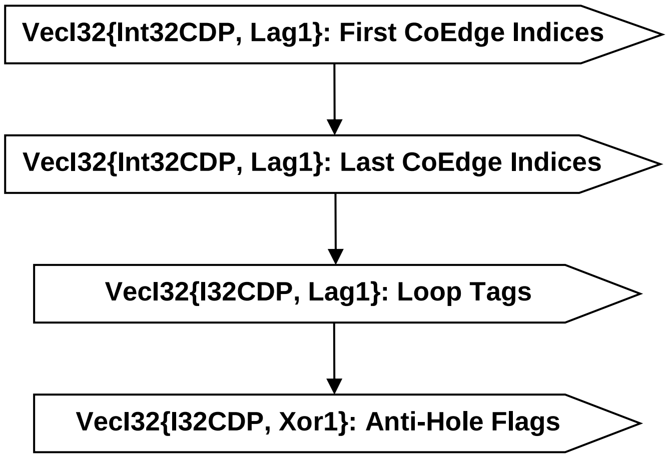


Figure 246 — Loops Topology Data collection

VecI32{Int32CDP, Lag1}: First CoEdge Indices

First CoEdge Indices is a vector of indices representing the index of the first CoEdge in each Loop. First CoEdge Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Last CoEdge Indices

Last CoEdge Indices is a vector of indices representing the index of the last CoEdge in each Loop. Last CoEdge Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{I32CDP, Lag1}: Loop Tags

Each Loop has an identifier tag. Loop Tags is a vector of identifier tags for a set of Loops. Loop Tags uses the Int32 version of the CODEC to compress and encode data.

VecI32{I32CDP, Xor1}: Anti-Hole Flags

Each Loop has a flag identifying whether the Loop is an anti-hole Loop. Anti-Hole Flags is a vector of anti-hole flags for a set of Loops.

In an uncompressed/decoded form the flag values have the following meaning:

Table 197 — JT B-Rep Loops Topology Data Anti-Hole Flag values

= 0	Loop is not an anti-hole Loop

= 1	Loop is an anti-hole Loop

Anti-Hole Flags uses the Int32 version of the CODEC to compress and encode data.

CoEdges Topology Data

A CoEdge defines a parameter space edge trim Loop segment (i.e. the projection of an Edge into the parameter space of the Face). CoEdges Topology Data specifies the underlying Edge and PCS Curve making up each CoEdge along with a MCS curve reversed flag and tag for each CoEdge.

The “Co” portion of the CoEdge name derives from the manifold topology definition that each Edge has exactly two Faces containing it; thus a CoEdge defines one Face’s “use” of an Edge and the adjoining Face also has a CoEdge (“edge use” in some other terminologies) for the same underlying Edge. This sharing of the same underlying Edge by two adjoining Faces requires an “MCS Curve Reversed Flag” on each CoEdge to indicate the edge traversal direction (i.e. for a proper manifold topology definition each CoEdge shall traverse the Edge in opposite directions).

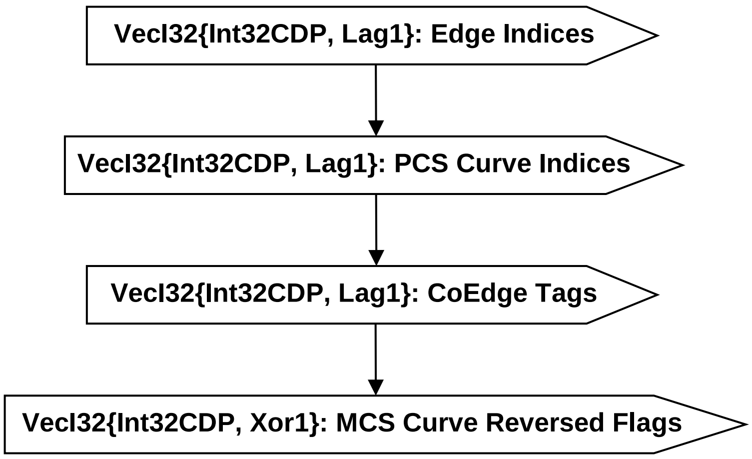


Figure 247 — CoEdges Topology Data collection

VecI32{Int32CDP, Lag1}: Edge Indices

Edge Indices is a vector of indices representing the index of the underlying Edge for each CoEdge. Edge Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: PCS Curve Indices

PCS Curve Indices is a vector of indices representing the index of the PCS Curve (UV Curve) for each CoEdge. PCS Curve Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: CoEdge Tags

Each CoEdge has an identifier tag. CoEdge Tags is a vector of identifier tags for a set of CoEdges. CoEdge Tags uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Xor1}: MCS Curve Reversed Flags

Each CoEdge has a flag indicating whether the directional sense of the associated Edge’s MCS curve should be interpreted as opposite the direction its parameterization implies. MCS Curve Reversed Flags is a vector of reverse flags for a set of CoEdges.

In an uncompressed/decoded form the flag values have the following meaning:

Table 198 — JT B-Rep MCS Curve Reversed Flag values

= 0	Directional sense of associated edges MCS curve should not be interpreted as opposite the direction its parameterization implies.
= 1	Directional sense of associated edges MCS curve should be interpreted as opposite the direction its parameterization implies.

MCS Curve Reversed Flags uses the Int32 version of the CODEC to compress and encode data.

Edges Topology Data

An Edge defines a model space trim Loop segment. Edges Topology Data specifies the underlying MCS Curve and start and end Vertex making up each Edge along with an identification tag for each Edge.

If manifold topology, then two faces join at a single model Edge and thus an edge is shared/referenced by two CoEdges (one per Face).

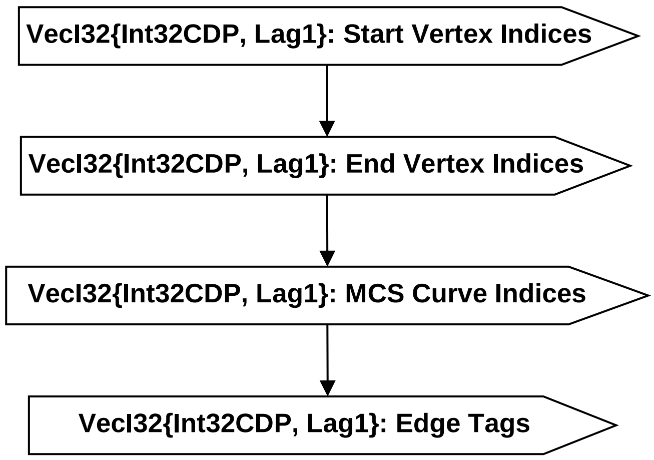


Figure 248 — Edges Topology Data collection

VecI32{Int32CDP, Lag1}: Start Vertex Indices

Start Vertex Indices is a vector of indices representing the index of the start Vertex in each Edge. Start Vertex Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: End Vertex Indices

End Vertex Indices is a vector of indices representing the index of the end Vertex in each Edge. End Vertex Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: MCS Curve Indices

MCS Curve Indices is a vector of indices representing the index of the MCS Curve (Model Space curve) for each Edge. MCS Curve Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Edge Tags

Each Edge has an identifier Tag. Edge Tags is a vector of identifier Tags for a set of Edges. Edge Tags uses the Int32 version of the CODEC to compress and encode data.

Vertices Topology Data

A Vertex is the simplest topological entity and is basically made up of a geometric Point. Vertices Topology Data specifies the underlying geometric Point making up each Vertex along with an identification tag for each Vertex.

The presence of Vertices Topology Data in a JT B-Rep topology definition is optional. Vertex data is optional because unlike most topological entities, no connectivity information is contained in a Vertex structure and Vertex data is also not necessary for performing operations such as tessellation or mass properties calculations.

A Vertex is usually shared/referenced by two or more Edges (e.g. if the corners of four rectangular Faces touches at a common point, this point is represented by a Vertex and is shared by four Edges).

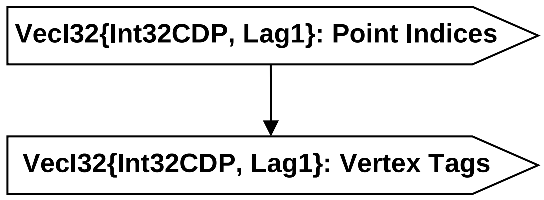


Figure 249 — Vertices Topology Data collection

VecI32{Int32CDP, Lag1}: Point Indices

Point Indices is a vector of indices representing the index of the geometric point for each Vertex. Point Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Vertex Tags

Each Vertex has an identifier Tag. Vertex Tags is a vector of identifier Tags for a set of Vertices. Vertex Tags uses the Int32 version of the CODEC to compress and encode data.

K.1.2.2 Geometric Data

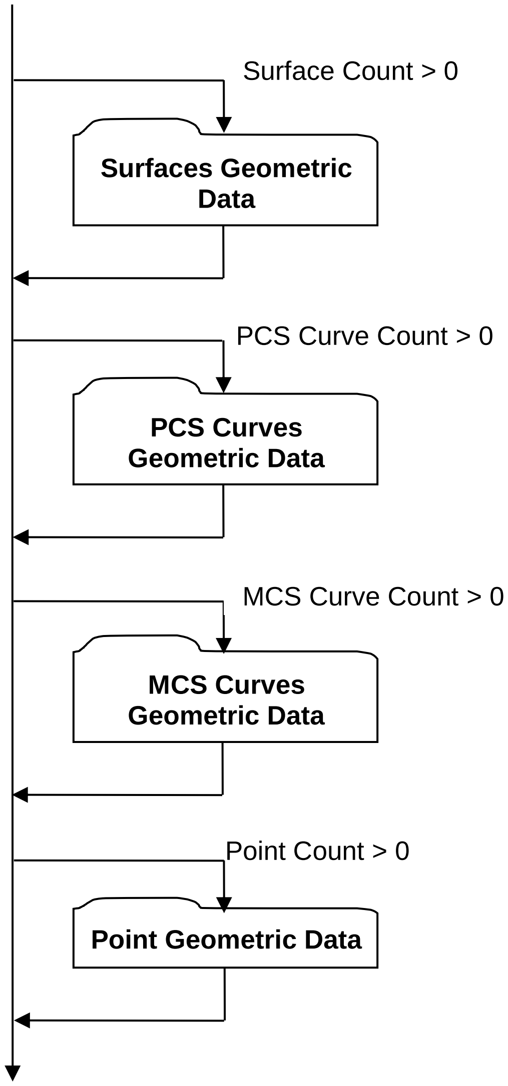


Figure 250 — Geometric Data collection

K.1.2.2.1 Surfaces Geometric Data

Surfaces Geometric Data collection contains the JT B-Rep's geometric Surface data. Currently only NURBS Surface types are supported within a JT B-Rep. The count/number of Surfaces within a JT B-Rep is indicated by data field Surface Count documented in Geometric Entity Counts.

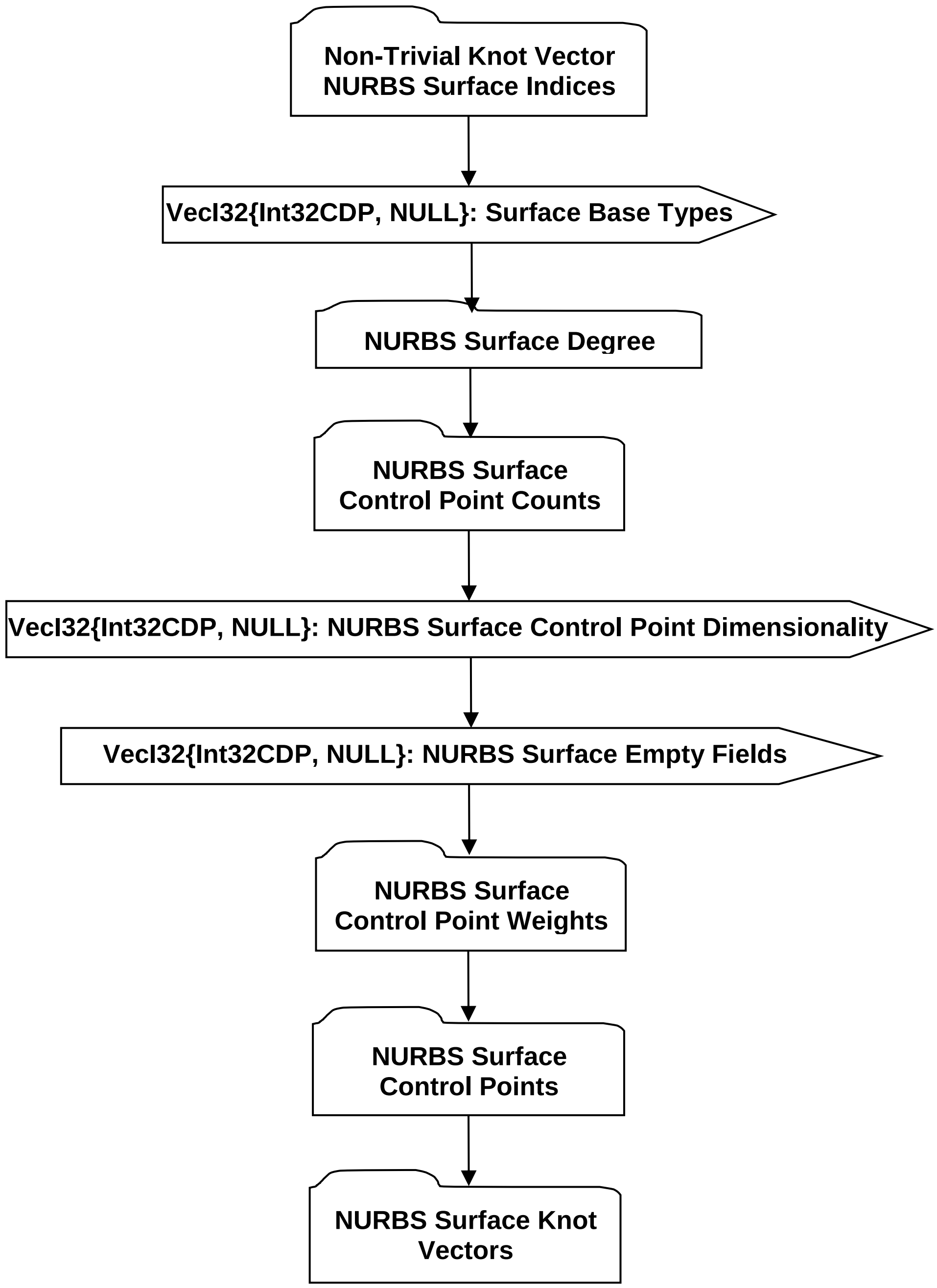


Figure 251 — Surfaces Geometric Data collection

VecI32{Int32CDP, NULL}: Surface Base Types

Each Surface is assigned a base type identifier. Surface Base Types is a vector of base type identifiers for each Surface in a list of Surfaces. Currently only NURBS Surface Base Type is supported, but a type identifier is still included in the specification to allow for future expansion of the JT Format to support other surface types within a JT B-Rep.

In an uncompressed/decoded form the Surface base type identifier values have the following meaning:

Table 199 — JT B-Rep Surface Base Type value

= 1	Surface is a NURBS surface

Surface Base Types uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: NURBS Surface Control Point Dimensionality

NURBS Surface Control Point Dimensionality is a vector of control point dimensionality values for each NURBS Surface in a list of Surfaces (i.e. there is a stored values for each NURBS Surface in the list).

In an uncompressed/decoded form dimensionality values have the following meaning:

Table 200 — JT B-Rep NURBS Surface Control Point Dimensionality values

= 3	Non-Rational (each control point has 3 coordinates)
= 4	Rational (each control point has 4 coordinates)

NURBS Surface Control Point Dimensionality uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: NURBS Surface Empty Fields

NURBS Surface Empty Fields is a vector of data. Each NURBS Surface in a list of Surfaces has one empty data field entry in this NURBS Surface Empty Fields vector. NURBS Surface Empty Fields uses the Int32 version of the CODEC to compress and encode data. Refer to Common Data Conventions and Construct, Empty Field.

Non-Trivial Knot Vector NURBS Surface Indices

Non-Trivial Knot Vector NURBS Surface Indices data collection specifies for both U and V directions the Surface index identifiers (i.e. indices to particular NURBS Surfaces within a list of Surfaces) for all NURBS Surfaces containing non-trivial knot vectors. A description/definition for “non-trivial knot vector” can be found in Compressed Entity List for Non-Trivial Knot Vector.

This Surface index data is stored in a compressed format.

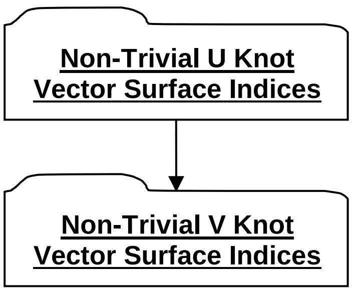


Figure 252 — Non-Trivial Knot Vector NURBS Surface Indices data collection

Both Non-Trivial U Knot Vector Surface Indices and Non-Trivial V Knot Vector Surface Indices have the same data format as that documented for data collection Compressed Entity List for Non-Trivial Knot Vector.

NURBS Surface Degree

NURBS Surface Degree data collection defines the Surface degree in both U and V directions for each NURBS Surface in a list of Surfaces (i.e. there are stored values for each NURBS Surface in the list). This degree data for the list of Surfaces is stored in a compressed format.

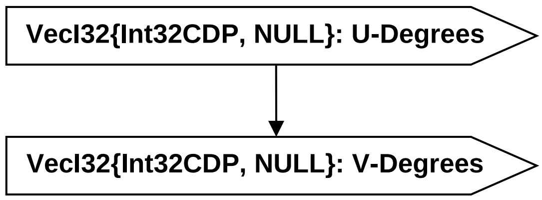


Figure 253 — NURBS Surface Degree data collection

VecI32{Int32CDP, NULL}: U-Degrees

U-Degrees is a vector of Surface degree values in U direction for each NURBS Surface in a list of Surfaces. U-Degrees uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: V-Degrees

V -Degrees is a vector of Surface degree values in V direction for each NURBS Surface in a list of Surfaces. V-Degrees uses the Int32 version of the CODEC to compress and encode data.

NURBS Surface Control Point Counts

NURBS Surface Control Point Counts defines the number of NURBS Surface control points for both U and V directions for each NURBS Surface in a list of Surfaces (i.e. there are stored values for each NURBS Surface in the list). The control point count data for the list of Surfaces in stored in a compressed format.

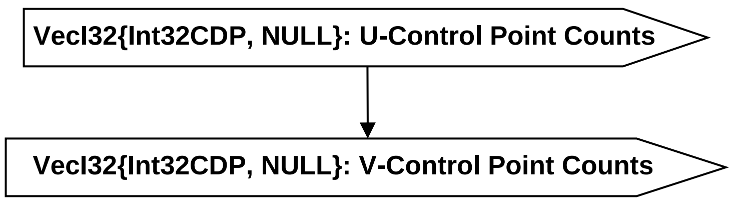


Figure 254 — NURBS Surface Control Point Counts data collection

VecI32{Int32CDP, NULL}: U-Control Point Counts

U-Control Point Counts is a vector of control point counts in U direction for each NURBS Surface in a list of Surfaces. U-Control Point Counts uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, NULL}: V-Control Point Counts

V-Control Point Counts is a vector of control point counts in V direction for each NURBS Surface in a list of Surfaces. V-Control Point Counts uses the Int32 version of the CODEC to compress and encode data.

NURBS Surface Control Point Weights

NURBS Surface Control Point Weights data collection defines the Weight values for a conditional set of Control Points for a list of NURBS Surfaces. The storing of the Weight value for a particular Control

Point is conditional, because if NURBS Surface Control Point Dimension is “non-rational” or the actual Control Point’s Weight value is “1”, then no Weight value is stored for the Control Point (i.e. Weight value can be inferred to be “1”).

The NURBS Surface Control Point Weights data is stored in a compressed format.

Figure 255 — NURBS Surface Control Point Weights data collection

Complete description for Compressed Control Point Weights Data can be found in Compressed Control Point Weights Data.

NURBS Surface Control Points

NURBS Surface Control Points is the compressed and/or encoded representation of the Control Point coordinates for each NURBS Surface in a list of Surfaces (i.e. there are stored values for each NURBS Surface in the list). Note that these are non-homogeneous coordinates (i.e. Control Point coordinates have been divided by the corresponding Control Point Weight values).

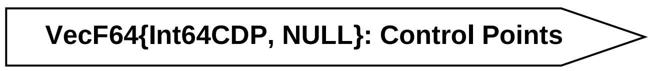


Figure 256 — NURBS Surface Control Points data collection

VecF64{Int64CDP, NULL}: Control Points

Control Points is a vector of Control Point coordinates for all the NURBS Surfaces in a list of Surfaces. All the NURBS Surfaces Control Point coordinates are cumulated into this single vector in the same order as the Surface appears in the Surface list (i.e. Surface-1 U Control Points, Surface-1 V Control Points, Surface-2 U Control Points, Surface-2 V Control Points, etc.). Control Points uses the Int64 version of the CODEC to compress and encode data in a “lossless” manner. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

NURBS Surface Knot Vectors

NURBS Surface Knot Vectors defines the knot vectors for both U and V directions for each NURBS Surface having non-trivial knot vectors in a list of Surfaces (i.e. there are stored values for each non-trivial knot vector NURBS Surface in the list). The NURBS Surfaces for which knot vectors are stored (i.e. those containing non-trivial knot vectors) are identified in data collection Non-Trivial Knot Vector NURBS Surface Indices documented in Non-Trivial Knot Vector NURBS Surface Indices.

The knot vector data for the list of Surfaces is stored in a compressed format.

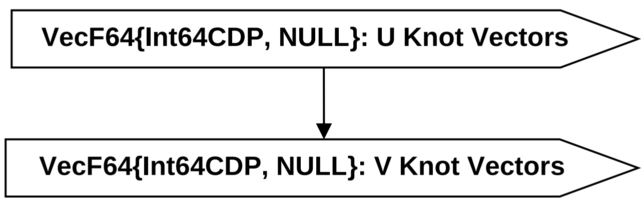


Figure 257 — NURBS Surface Knot Vectors data collection

VecF64{Int64CDP, NULL}: U Knot Vectors

U Knot Vectors is a list of knot vector values in U direction for each NURBS Surface having non-trivial knot vectors in a list of Surfaces. All these NURBS Surface U direction non-trivial knot vectors are cumulated into this single list in the same order as the Surface appears in the Surface list (i.e. Surface-N Non-Trivial U Knot Vector, Surface-M Non-Trivial U Knot Vector, etc.). U Knot Vectors uses the Int64 version of the CODEC to compress and encode data. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

VecF64{Int64CDP, NULL}: V Knot Vectors

V Knot Vectors is a list of knot vector values in V direction for each NURBS Surface having non-trivial knot vectors in a list of Surfaces. All these NURBS Surface V direction non-trivial knot vectors are cumulated into this single list in the same order as the Surface appears in the Surface list (i.e. Surface-N Non-Trivial V Knot Vector, Surface-M Non-Trivial V Knot Vector, etc.). V Knot Vectors uses the Int64 version of the CODEC to compress and encode data. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

K.1.2.2.2 PCS Curves Geometric Data

PCS Curves Geometric Data collection contains the JT B-Rep's Parameter Coordinate Space geometric Curve data (i.e. UV Curve data). This geometric PCS Curve data is divided up into two collection types; one data collection for what are considered "Trivial" PCS curves and one data collection for compressed/encoded PCS NURBS Curve data.

"Trivial" PCS Curves are those UV Curves whose definition is such that the actual UV Curve definition can be derived from the parametric domain definition by storing a limited amount of descriptive data for each UV curve (i.e. do not have to store the complete NURBS UV Curve definition).

The count/number of PCS Curves within a JT B-Rep is indicated by data field PCS Curve Count documented in Geometric Entity Counts.

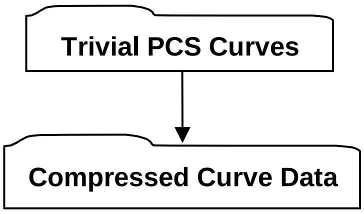


Figure 258 — PCS Curves Geometric Data collection

Complete description for Compressed Curve Data can be found in Compressed Curve Data.

Trivial PCS Curves

Trivial PCS Curves data collection represents those UV curves whose definition is such (i.e. "trivial" enough) that the actual UV curve definition can be derived from the parametric domain definition by storing a limited amount of descriptive data for each UV curve (i.e. do not have to store the complete UV curve definition). These Trivial PCS Curves are grouped into three classifications (Trivial Domain Loop, Trivial Box Loop, or Trivial Domain UV Curve) and stored as described in the following sub-sections.

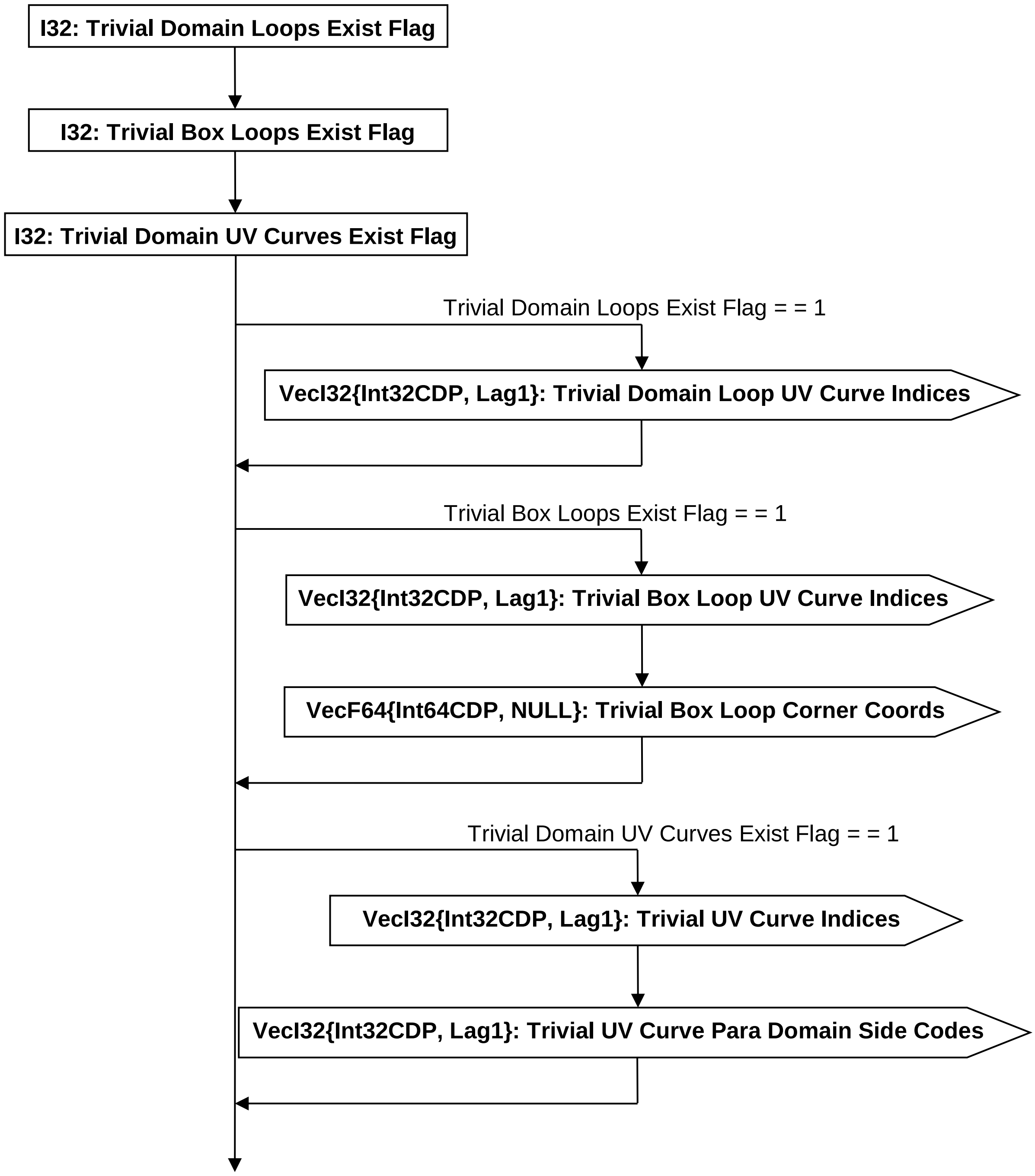


Figure 259 — Trivial PCS Curves data collection

I32: Trivial Domain Loops Exist Flag

Trivial Domain Loops Exist Flag is a flag indicating whether "trivial" domain loops exist/follow. A Trivial Domain Loop is a Loop that encloses the entire parametric domain (i.e. all UV Curves of the Loop span the entire length of the Surface parametric domain). Given this criteria a Trivial Domain Loop shall always be made up of four Trivial Domain UV curves.

Table 201 — Trivial Domain Loops Exist Flag values

= 0	Trivial Domain Loops do not exist.
= 1	Trivial Domain Loops exist.

I32: Trivial Box Loops Exist Flag

Trivial Box Loops Exist Flag is a flag indicating whether “trivial” box loops exist/follow. A trivial Box Loop is a Loop that forms a rectangle (i.e. corresponding curve end coordinates of opposite sides of the box are equal). Given this criteria a Trivial Box Loop shall always be made up of four UV curves.

Table 202 — Trivial Box Loops Exist Flag values

= 0	Trivial Box Loops do not exist.
= 1	Trivial Box Loops exist.

“Equality of corresponding curve end coordinates of opposite sides of the box” is represented graphically in the figure below.

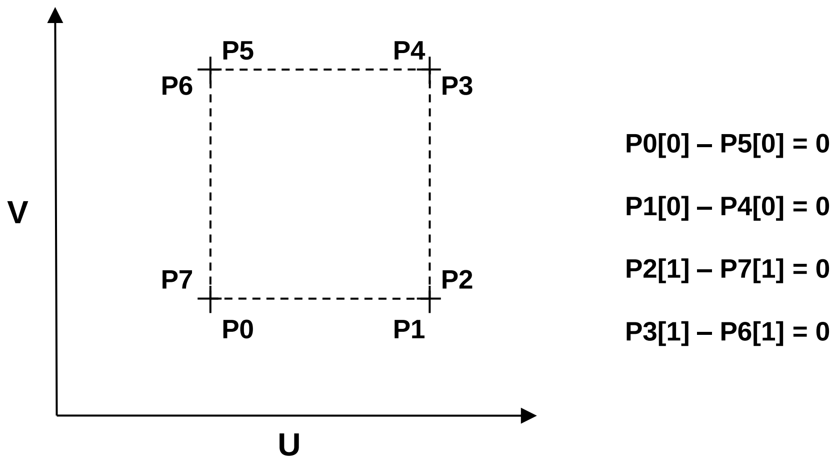


Figure 260 — Equality of corresponding curve end coordinates of opposite sides of the box

I32: Trivial Domain UV Curves Exist Flag

Trivial Domain UV Curves Exist Flag is a flag indicating whether “trivial” domain UV curves (Loop CoEdges) exist/follow that are not part of a Trivial Domain Loop or Trivial Box Loop (i.e. a Loop contains some UV curves that span the entire length of the Surface parametric domain but not all the Loop UV curves meet this criteria and thus not captured as part of the Trivial Domain Loop data).

Table 203 — Trivial Domain UV Curves Exist Flag values

= 0	Trivial Domain UV Curves do not exist.
= 1	Trivial Domain UV Curves exist.

VecI32{Int32CDP, Lag1}: Trivial Domain Loop UV Curve Indices

Trivial Domain Loop UV Curve Indices is a vector of all UV curve indices that are part of a Trivial Domain Loop. Note that each Trivial Domain Loop is always made up of four UV curves (thus four UV curve indices per Loop). Trivial Domain Loop UV Curve Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Trivial Box Loop UV Curve Indices

Trivial Box Loop UV Curve Indices is a vector of all UV Curve indices that are part of a Trivial Box Loop. Note that each Trivial Box Loop is always made up of four UV Curves (thus four UV Curve indices per Loop). Trivial Box Loop UV Curve Indices uses the Int32 version of the CODEC to compress and encode data.

VecF64{Int64CDP, NULL}: Trivial Box Loop Corner Coords

Trivial Box Loop Corner Coords is a vector of box corner coordinates for all Trivial Box Loops (i.e. each Box Loop will store two box corner coordinates). A Box Loop's set of "box corner coordinates" are the coordinates of the two min/max diagonally opposite corners of the box. Note that if the Box Loop is a "hole", then the max and min corners are the other ends of the respective box sides that contain the max and min corners. Trivial Box Loop Corner Coords uses the Int64 version of the CODEC to compress and encode data. Each deserialized 64 bit integer number should be converted to bit wise equivalent 64 bit floating number.

VecI32{Int32CDP, Lag1}: Trivial UV Curve Indices

Trivial UV Curve Indices is a vector of all Loop UV Curve indices that are not part of a Trivial Domain Loop or Trivial Box Loop. Trivial UV Curve Indices uses the Int32 version of the CODEC to compress and encode data.

VecI32{Int32CDP, Lag1}: Trivial UV Curve Para Domain Side Codes

Trivial UV Curve Para Domain Side Codes is a vector containing a "side code" for each Trivial UV Curve indicating which parametric domain side the UV Curve lies on.

In an uncompressed/decoded form the parametric domain side values have the following meaning:

Table 204 — Trivial UV Curve Para Domain Side Codes values

= 0	Bottom side of parametric domain
= 1	Right side of parametric domain
= 2	Top side of parametric domain
= 3	Left side of parametric domain

Trivial UV Curve Para Domain Side Codes uses the Int32 version of the CODEC to compress and encode data.

K.1.2.2.3 MCS Curves Geometric Data

MCS Curves Geometric Data collection contains the JT B-Rep's Model Coordinate System geometric Curve data (i.e. XYZ Curve data). Currently only NURBS Curve types are supported within a JT B-Rep. The count/number of MCS Curves within a JT B-Rep is indicated by data field MCS Curve Count documented in Geometric Entity Counts.

Figure 261 — MCS Curves Geometric Data collection

Complete description for Compressed Curve Data can be found in Compressed Curve Data.

K.1.2.2.4 Point Geometric Data

Point Geometric Data collection contains the JT B-Rep's geometric Point data. Each Point is simply represented by a CoordF32 for the Point's coordinate components. The count/number of Points within a JT B-Rep is indicated by data field Point Count documented in Geometric Entity Counts.

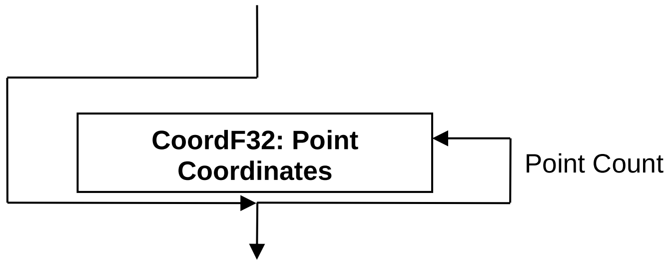


Figure 262 — Point Geometric Data collection

CoordF32: Point Coordinates

Point Coordinates specifies the XYZ coordinate components for a Point.

K.1.2.3 Topological Entity Tag Counters

Topological Entity Tag Counters data collection specifies the next available "unique" tag value for each entity type in a JT B-Rep. These are rolling tag counters that are meant to be used for assigning a unique tag when a new entity is added to a JT B-Rep.

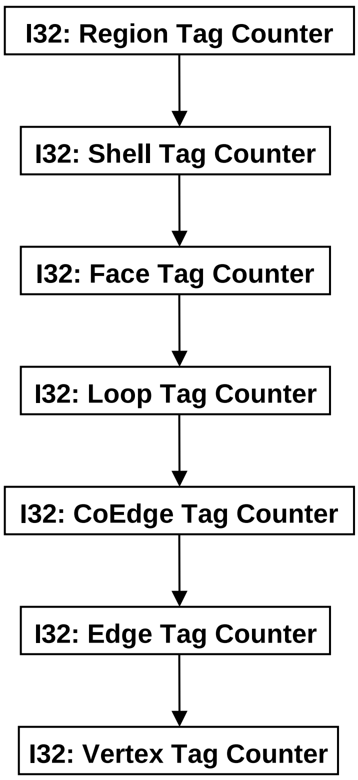


Figure 263 — Topological Entity Tag Counters data collection

I32: Region Tag Counter

Region tag Counter specifies the next available “unique” tag value for Region entity.

I32: Shell Tag Counter

Shell Tag Counter specifies the next available “unique” tag value for Shell entity.

I32: Face Tag Counter

Face Tag Counter specifies the next available “unique” tag value for Face entity.

I32: Loop Tag Counter

Loop Tag Counter specifies the next available “unique” tag value for Loop entity.

I32: CoEdge Tag Counter

CoEdge Tag Counter specifies the next available “unique” tag value for CoEdge entity.

I32: Edge Tag Counter

Edge Tag Counter specifies the next available “unique” tag value for Edge entity.

I32: Vertex Tag Counter

Vertex Tag Counter specifies the next available “unique” tag value for Vertex entity.

K.1.2.3.1 B-Rep CAD Tag Data

The B-Rep CAD Tag Data collection contains the list of persistent IDs, as defined in the CAD System, to uniquely identify individual Faces and Edges in the JT B-Rep. The existence of this B-Rep CAD Tag Data collection is dependent upon the value of previously read data field CAD Tags Flag as documented in JT B-Rep Element.

If B-Rep CAD Tag Data collection is present, there will be a CAD Tag for every Face and every Edge in the JT B-Rep and the list order will be Face CAD Tags followed by Edge CAD Tags. Therefore the total number of CAD Tags in the list should be equal to “Face Count + Edge Count” as documented in Topological Entity Counts.

Figure 264 — B-Rep CAD Tag Data collection

Complete description for Compressed CAD Tag Data can be found in Compressed CAD Tag Data.

Annex L
(deprecated)
PMI Data Segment

JT files created prior to the Version 8 format description that have been migrated to a ISO 14306:2012 format description may have PMI information represented in a PMI Data Segment. From a parsing point of view, a PMI Data Segment should be treated exactly the same as a PMI Manager Meta Data Element.

PMI Data Segments are deprecated and should not be written to JT file

Annex M

Procedural Geometry – Evaluation and Approximation

The following section describes in detail algorithms used for evaluation of procedural geometry elements of type Intersection Curve and Blend Edge Surface.

M.1 Introduction & Scope

These notes are intended to help people understand the procedural geometry that is used in JT/XT data. Specifically, we discuss procedural intersection curves (“icurves” for short), and rolling-ball blend surfaces. The chapter includes mathematical background and pseudocode that can be used to evaluate a point at a given parametric location on either an icurve or a blend surface; this evaluation procedure provides a clear and completely unambiguous definition of the geometry.

Once the evaluation functions are available, it is quite straightforward to approximate an icurve or a blend surface with b-spline geometry of some sort. This is just general-purpose approximation technology, but we explain how to do it, anyway. Internally, Parasolid makes very little use of spline approximations, so this is not an area where we invest heavily, and the algorithms given below could probably be improved. The spline geometry is easier to import into other CAD systems, and might possibly deliver better performance in some types of computations. The disadvantage, of course, is that the b-spline geometry will occupy more space, and it will only replicate the true procedural geometry within some tolerance.

M.2 Notation

Upright bold upper-case letters like **A**, **B**, **C**, **P, X** denote 3D points and vectors. Points and vectors are not the same thing, of course, but the context should make things clear. Italic lower-case letters like s, t, u, v, w denote real numbers, which are often parameter values on curves and surfaces.

M.3 Pseudocode

The pseudocode is written in a language that is roughly C#. We assume that we have Point and Vector classes that represent 3D points and vectors respectively. These classes have obvious functions, like Vector.Norm (computes the length of the given vector), and so on. We also assume that the usual arithmetic operators (+,-,*) have been overloaded in these classes, so that P - Q is the difference of two points (which is a vector), U + V is the sum of two vectors, U * V is their dot product, and so on.

The procedural geometry algorithms used inside Parasolid are extremely complex. Parasolid has a zero-regression policy. So, if we develop a new algorithm that improves 99% of all data cases, we sometimes retain the old algorithm to avoid regressions in the other 1%. There are many different paths through the code, and the descriptions below represent only the primary ones (i.e. the ones that are most often followed, and the ones that we regard as the best).

M.4 Intersection Curve

M.4.1 Intersection Curve Basics

Suppose we have two surfaces **A** and **B** with parameterizations $(s,t) \mapsto \mathbf{A}(s,t)$ and $(u,v) \mapsto \mathbf{B}(u,v)$, and we are interested in their curve of intersection, $\mathbf{C}$. Specifically, suppose we are given a parameter value w, and we want to calculate the corresponding point $\mathbf{C}(w)$ on the icurve. If we can do this for any given w, then we have a clear and unambiguous definition of the icurve, and approximating it by an explicit spline curve should be straightforward.

In a JT file, the representation of an icurve is a sequence of “chart points” $\mathbf{P}_0, \dots, \mathbf{P}_n$. To each chart point $\mathbf{P}_i$ we assign a parameter value $w = w_i$; detailed procedures for doing this are explained later, but, for now, assume that this has already been done, and that the w_i values are strictly increasing. We assume that the given parameter value w lies in the range $w_0 \leq w \leq w_n$. Therefore, we can find an index i such that $w_i \leq w \leq w_{i+1}$.

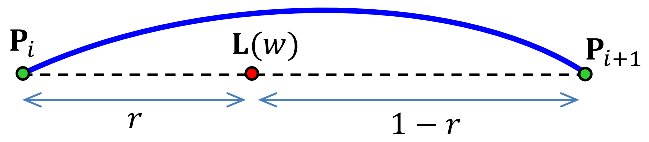
We first find a point $\mathbf{L}(w)$ on the chord $\mathbf{P}_i\mathbf{P}_{i+1}$. We let r denote the ratio

$$r = \frac{w - w_i}{w_{i+1} - w_i}$$

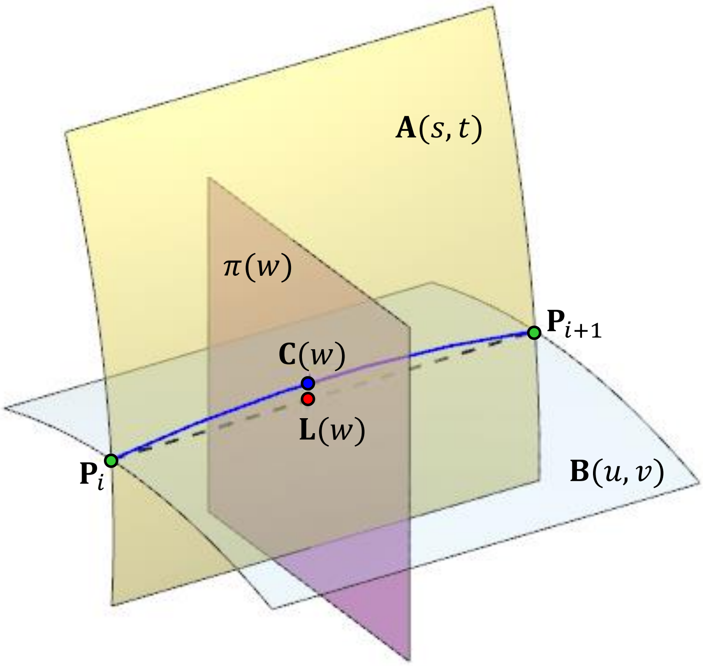
and then

$$\boldsymbol{L}(w) = (1 - r)\boldsymbol{P}_i + r\boldsymbol{P}_{i+1} = \frac{w_{i+1} - w}{w_{i+1} - w_i}\boldsymbol{P}_i + \frac{w - w_i}{w_{i+1} - w_i}\boldsymbol{P}_{i+1}$$

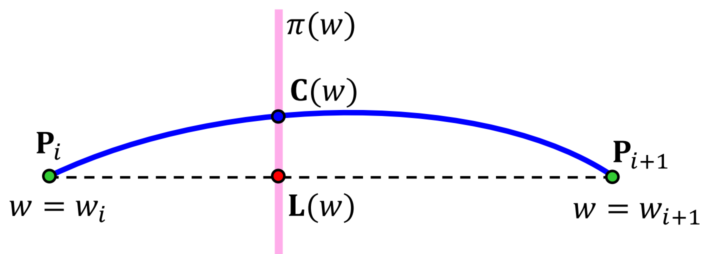
We see that $\mathbf{L}(w)$ divides the chord in the ratio r to $1 - r$, as shown below:



Next, we construct the plane $\pi(w)$ that passes through the point $\mathbf{L}(w)$ and is perpendicular to the chord $\mathbf{P}_i\mathbf{P}_{i+1}$. The point $\mathbf{C}(w)$ of the intersection curve is the intersection of the two surfaces $\mathbf{A}$ and $\mathbf{B}$ and the plane $\pi(w)$, as shown in the pictures below



The construction is clearer if we omit the two surfaces and look at the plane $\pi(w)$ edge-on:



M.4.2 Populating Chart Points

An icurve consists primarily of a sequence of "chart points". Each chart point is represented by a structure called an hvec. The name "hvec" is an abbreviation for "heptavector", so-called because the structure originally had 7 members. The members are described in the following declaration:

```
struct hvec {
   Point       CurvePoint;         // Point on curve
   double[]    st;                 // Parameter value pair (s,t) on first surface
   double[]    uv;                 // Parameter value pair (u,v) on second surface
   Vector      CurveTangent;       // Curve tangent vector (a unit vector)
   double      w;                  // Curve parameter value }
```

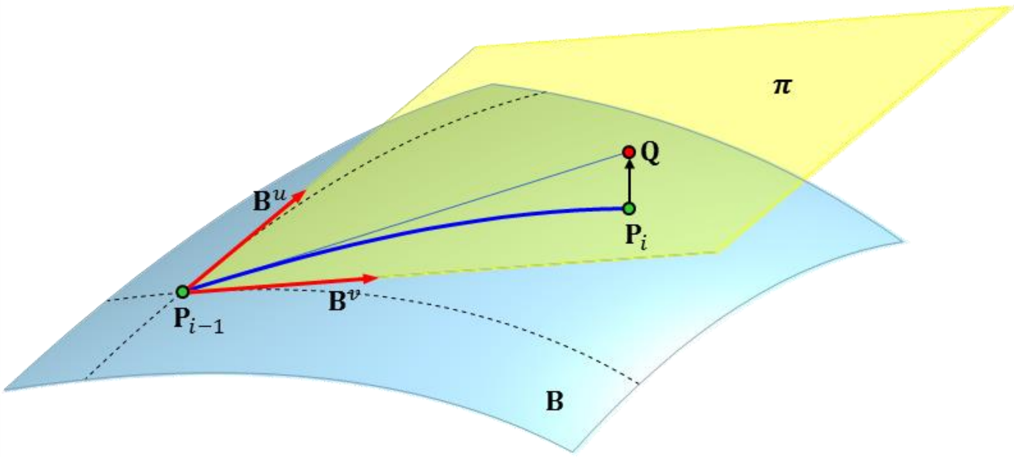
In the hvecs structures of an icurve in a JT file, only the CurvePoint fields are populated. The remaining elements of the hvec structures can all be derived from these as explained below.

M.4.2.1 Getting Surface Parameter Values

First, we need to calculate the surface parameter values at each chart point $\mathbf{P}_i$. In other words, we need to find two parameter pairs (s,t) and (u,v) such that $\mathbf{A}(s,t) = \mathbf{P}_i$ and $\mathbf{B}(u,v) = \mathbf{P}_i$. The computations for the two surfaces are similar, and independent of each other, so we just illustrate with the surface **B**.

If the surface **B** is simple and analytic, the calculations are easy, so we omit them.

If the surface **B** is complex, we need to use an iterative numerical method. The major problem with the numerical approach is that the iteration needs a good starting point. To obtain one, we use a construction based on the (parameterized) tangent plane $\boldsymbol{\pi}$ at the previous chart point, $\mathbf{P}_{i-1}$, as shown below:



Let $\mathbf{B}^u$ and $\mathbf{B}^v$ denote the partial derivatives of the surface **B** at the previous point $\mathbf{P}_{i-1}$. Then the equation of the tangent plane at $\mathbf{P}_{i-1}$ can be written as

$$\boldsymbol{\pi}(\delta u, \delta v) = \mathbf{P}_{i-1} + \delta u \mathbf{B}^u + \delta v \mathbf{B}^v$$

We want to find values of u and v such that the point $\boldsymbol{\pi}(u,v)$ on the tangent plane is as close as possible to the point $\boldsymbol{P}_i$.

This means that $\pi(u,v)$ must be the orthogonal projection of $\mathbf{P}_i$ onto the plane π, so

$$(\pi(\delta u, \delta v) - \boldsymbol{P}_i) \cdot \boldsymbol{B}^u = 0$$

$$(\pi(\delta u, \delta v) - \boldsymbol{P}_i) \cdot \boldsymbol{B}^v = 0$$

Substituting for $\pi(\delta u, \delta v)$, and rearranging, this gives

$$\delta u(\boldsymbol{B}^u \cdot \boldsymbol{B}^u) + \delta v(\boldsymbol{B}^u \cdot \boldsymbol{B}^v) = (\boldsymbol{P}_i - \boldsymbol{P}_{i-1}) \cdot \boldsymbol{B}^u$$

$$\delta u(\boldsymbol{B}^u \cdot \boldsymbol{B}^v) + \delta v(\boldsymbol{B}^v \cdot \boldsymbol{B}^v) = (\boldsymbol{P}_i - \boldsymbol{P}_{i-1}) \cdot \boldsymbol{B}^v$$

This is a system of two linear equations in the unknowns δu and δv, which we can easily solve (by Cramer's rule, for example). So, if (u_0, v_0) denote the surface parameter values at $\mathbf{P}_{i-1}$, then it is reasonable to assume that $\mathbf{B}(u_0 + \delta u\, , v_0 + \delta v) \approx \mathbf{P}_i$, so $(u, v) = (u_0 + \delta u, v_0 + \delta v)$ will be suitable starting values in an iterative algorithm that finds (u, v) such that $\mathbf{B}(u, v) = \mathbf{P}_i$.

Finding (u, v) such that $\mathbf{B}(u, v) = \mathbf{P}_i$ will be impossible, in floating point arithmetic, of course, so a more practical approach is to try to find (u, v) that minimizes $\|\mathbf{B}(u, v) - \mathbf{P}_i\|^2$. Since the point $\mathbf{P}_i$ lies very close to the surface **B**, this minimum will be very small, and this is may be helpful in locating it. The pseudocode for this process is as follows:

```
// Finds parameter values at a given point on a surface
// Input:
//    B       -- the surface
//    P       -- a point on the surface
//    nearUV -- (u,v) values such that B(u,v) is close to P. Same as (u0,v0) in notes
above.
// Returns:
//    The (u,v) values such that B(u,v) = P

double[] SurfaceParametersAtPoint(Surface B, Point P, double[] nearUV)
{
   // Calculate position and partial derivatives at previous chart point
   Point     Q = B.Position(nearUV);      // Position at nearUV = (u0,v0)
   Vector   Bu = B.DerivDu (nearUV);      // Partial deriv wrt u at nearUV
   Vector   Bv = B.DerivDv (nearUV);      // Partial deriv wrt v nearUV

   // Construct linear equations for finding uv delta values, and solve
   double a = Bu * Bu;  double b = Bu * Bv;  double h = (P - Q) * Bu;
   double c = Bu * Bv;  double d = Bv * Bv;  double k = (P - Q) * Bv;
   double[] deltaUV = LinearSystemSolve(a, b, c, d, h, k);

   // Add on delta values: startUV = nearUV + deltaUV
   double[] startUV = { nearUV[0] + deltaUV[0], nearUV[1] + deltaUV[1] };

   // Minimize to find closest point on surface, where B(u,v) = P,
   double outUV = FindMinimum(SurfaceDistance2, double[] startUV);
   return outUV;
}
```

Here we have assumed the existence of three other standard functions.

First, a simple linear system solver:

```
// Computes (x,y) that are solutions to the system:
//       a*x + b*y = h
//       c*x + d*y = k
// Input:
//   a,bc,d -- coefficients
//   h,k    -- right-hand sides
// Returns:
//   Solution (x,y) of the linear system

double[] LinearSystemSolve(double a, double b, double c, double d, double h, double k)

Next, a function that calculates (squared) distance from a point to the surface B:
```

```
// Calculates squared distance from a given point to a point on a surface
// Input:
//   surf -- the surface
//   P    -- the point
// Returns:
// Squared distance from point P to point surf(u,v)

double SurfaceDistance2(Surface surf, Point P, double[] uv)
{
   Position Q = surf.Point(uv);
   double dist2 = (P - Q) * (P - Q);
   return dist2;
}
```

and, finally, an iterative numerical minimization function

```
// Minimizes a real-valued function of two variables
// Input:
//   f     -- the real-valued function to be minimized
//   start -- the starting values of the variables, from which to start iteration
// Returns:
//   Arguments for which the function f is a local minimum

double[] FindMinimum(RealFunction2 f, double[] start)
```

There is nothing very special about the minimization function that is required here. Parasolid uses a home-grown function, but it is unremarkable. It uses a maximum of 20 iterations, and it terminates when either the function value or the step size is less than a small tolerance epsilonDistance, whose value is 10^{-8}. Examples of suitable functions can be found in the Numerical Recipes book [7], in the GNU Scientific Library [8], in MINPACK[9], or in numerous other places. Specifically, the functions frpmin or dfpmin from the Numerical Recipes book have the necessary functionality.

We do exactly the same computation to find parameter values (s,t) on the first surface, $\mathbf{A}$, such that $\mathbf{A}(s,t) = \mathbf{P}_i$. We write the (s,t) and (u,v) values into the hvec.

M.4.2.2 Special Case: the First Point

In the previous section, we calculated surface parameter values at the current chart point by "stepping" from the previous point to get to a place where we could start an iterative algorithm. Of course, if there is no previous point, this doesn't work, so we need some other way to start our iteration. Unfortunately, a brute-force global search is the only option. One possible way to do this is outlined below. Again, we use the surface **B** as an example.

The objective is to find parameter values (u,v) such that $\mathbf{B}(u,v)$ is close to the current point **P**; these (u,v) values can then be used to start an iterative minimization, as described in the previous section. The basic idea is to test the value of $\mathbf{B}(u,v)$ at some $m \times n$ locations $(u,v) = (u_i, v_j)$, and simply choose the location that's closest to **P**. The question is: what values should be used for m and n. Some suggestions are outlined below:

Surface Type	Description
Bezier patch	If the patch has degrees $p \times q$, use an array size somewhere between $p \times q$ and $2p \times 2q$.
NURBs surface	Treat each constituent Bézier patch as described above
Offset surface	Use the same array of locations as on the base surface
Extruded surface	Do a 2D calculation in a plane that's perpendicular to the extrusion direction. If the generator curve is a spline of degree p, use somewhere between p and $2p$ points on each of its Bézier segments.
Revolved surface	Again, do the computation in 2D, in a plane containing the surface's axis of rotation. If the generator curve is a spline, handle as above.

If the global searching method is implemented in a function GlobalSearch, then the pseudocode for this approach is as follows:

```
// Do global search to get a near point
double[] nearUV = GlobalSearch(B, P);

// Use this near point as input to SurfaceParametersAtPoint
double outUV = SurfaceParametersAtPoint(B, P, nearUV)
```

This basic calculation should work adequately, but there are many ways to improve it. Firstly, convex hull or boxing algorithms can be used to eliminate large portions of the surface that are clearly further from **P** than the current minimum. This will not produce better results, but it will certainly improve performance. Also, a tessellation could be used instead of an array of points — this would require point-to-triangle distance calculations rather than point-to-point calculations, but it will yield better starting values. The tessellation can be a simple one constructed from the $m \times n$ array of points described above, or more sophisticated tesselation algorithms could be used.

If stepping from the previous chart point (as described in section) fails, then the global search method described in this section can be used as a backup. In fact, the global search could be used at every point, in principle, though this is likely to result in poor performance.

M.4.2.3 Getting the Tangent Vector

The next thing that's needed in the hvec is the CurveTangent vector. This is just the unitized cross product of the two surface normals at the point, so it's easy to compute:

```
Vector normA = A.Normal(st);
Vector normB = B.Normal(uv);
Vector curveTangent = Vector.UnitCross(normA, normB);
```

Obviously this calculation won't work if the two surface normal are parallel (because the surfaces are tangent), but this should not happen at a chart point.

M.4.2.4 Getting Chart Point Parameters

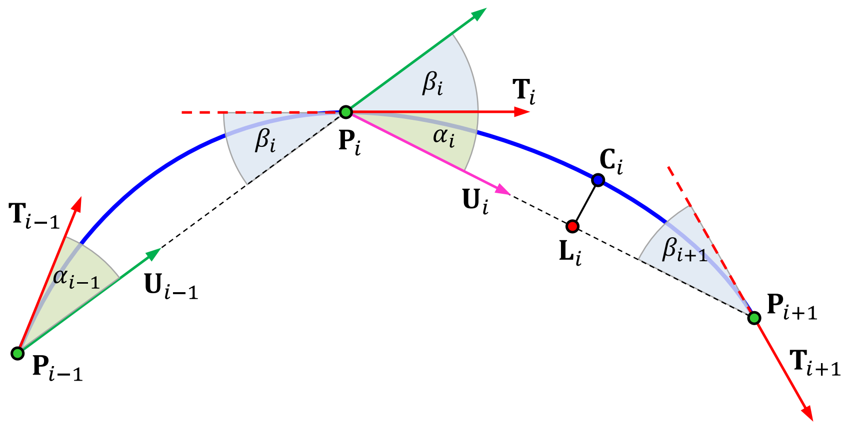
Finally, we need to assign a parameter value w_i to each chart point $\mathbf{P}_i$. These parameter values do not have any influence on the shape of the icurve — they are used only to identify an interval $[w_i, w_{i+1}]$ containing the given value w, and to compute a local parameter r (a ratio) within this interval. The computation of points on the icurve will work no matter what sequence of values $w_0, \dots, w_n$ we use (as long as they are increasing). The w_i values do not affect the shape of the icurve, they only affect the mapping $w \mapsto \mathbf{C}(w)$. So, if we only care about the shape of the icurve, and not about its parameterization, it doesn't much matter what w_i values we assign to the chart points.

The simplest option would be to make the w_i values evenly spaced — we would just set $w_i = i$ or $w_i = i/n$ for $i = 0,1, \dots, n$. Saying this another way, this scheme makes the parameter increments $h_i = w_{i+1} - w_i$ constant. Another simple option would be to space the w_i according to chord-lengths, so

$$h_i = w_{i+1} - w_i = \|\boldsymbol{P}_i - \boldsymbol{P}_{i-1}\| \qquad \text{for } i = 0,1, \dots, n-1$$

However, both of these simple schemes have a drawback: the icurve constructed this way will be G1 (it will have continuous unit tangent) but it will not necessarily be C1 (it will not have a continuous first derivative). In many situations, the lack of C1 continuity is not a problem — a jump in the length of the first derivative vector as the curve passes through a chart point is not likely to cause much trouble. In other situations, C1 continuity is desirable, and it turns out that this can be achieved by a careful choice of the parameter values w_i (or, more precisely, by the right choice of the parameter increments h_i). There is no special mathematics here: it is well-known that ***any*** piecewise G1 curve can be made C1 simply by adjusting parameter increments in this way. The details are given below.

Let $\mathbf{T}_i$ be the unit tangent of the icurve at the chart point $\mathbf{P}_i$, let $\mathbf{V}_i = \mathbf{P}_{i+1} - \mathbf{P}_i$ be the i-th chord vector, let $d_i = \|\mathbf{V}_i\|$ be its length, and let $\mathbf{U}_i = \mathbf{V}_i / d_i$ be the unit vector along the i-th chord. Also, let α_i and β_i be the angles between the tangent $\mathbf{T}_i$ and the unit chord vectors $\mathbf{U}_i$ and $\mathbf{U}_{i-1}$ respectively, as shown in the diagram below:



Our goal is to compute suitable values for the parameter increments $h_i = w_{i+1} - w_i$.

On the segment where $w_i \le w \le w_{i+1}$, the equation of the chord is

$$\boldsymbol{L}_i(w) = \frac{w_{i+1} - w}{w_{i+1} - w_i}\,\boldsymbol{P}_i + \frac{w - w_i}{w_{i+1} - w_i}\,\boldsymbol{P}_{i+1} = \boldsymbol{P}_i + \frac{w - w_i}{h_i}\boldsymbol{V}_i$$

and so

$$\boldsymbol{L}_i'(w) = \frac{\boldsymbol{V}_i}{h_i} = \frac{d_i \boldsymbol{U}_i}{h_i}$$

Since the vector $\mathbf{C}_i(w) - \mathbf{L}_i(w)$ is perpendicular to the chord $\mathbf{P}_i\mathbf{P}_{i+1}$, we have

$$[\boldsymbol{C}_i(w) - \boldsymbol{L}_i(w)] \cdot \boldsymbol{U}_i = 0$$

Differentiating with respect to w and substituting the expression for $\mathbf{L}_i'(w)$ from above gives

$$\left[\boldsymbol{C}_i'(w) - \frac{d_i \boldsymbol{U}_i}{h_i}\right] \cdot \boldsymbol{U}_i = 0$$

Then, since $\mathbf{U}_i \cdot \mathbf{U}_i = 1$, we get

$$\boldsymbol{C}_i'(w) \cdot \boldsymbol{U}_i = \frac{d_i}{h_i}$$

When $w = w_i$, we have $\mathbf{C}_i'(w_i) \cdot \mathbf{U}_i = \|\mathbf{C}_i'(w_i)\| \cos \alpha_i$, so

$$\|\boldsymbol{C}_i'(w_i)\| = \frac{d_i}{h_i\, cos\, \alpha_i}$$

Doing the same sort of calculations on the icurve segment $\mathbf{C}_{i-1}$ where $w_{i-1} \le w \le w_i$, we get

$$\boldsymbol{C}_{i-1}'(w) \cdot \boldsymbol{U}_{i-1} = \frac{d_{i-1}}{h_{i-1}}$$

and setting $w = w_i$ in this equation, we get

$$\|\boldsymbol{C}_{i-1}'(w_i)\| = \frac{d_{i-1}}{h_{i-1}\cos\beta_i}$$

To get C1 continuity at $w = w_i$, we need to have $\|\mathbf{C}_{i-1}'(w_i)\| = \|\mathbf{C}_i'(w_i)\|$, which implies that

$$\frac{d_{i-1}}{h_{i-1}\cos\beta_i} = \frac{d_i}{h_i\cos\alpha_i}$$

and hence

$$h_i = \frac{\boldsymbol{d}_i\cos\beta_i}{\boldsymbol{d}_{i-1}\cos\alpha_i}h_{i-1}$$

This is the form that's actually used in the Parasolid code. But $\cos\beta_i = (\mathbf{T}_i\cdot\mathbf{V}_{i-1})\,/d_{i-1}$ and $\cos\alpha_i = (\mathbf{T}_i\cdot\mathbf{V}_i)\,/d_i$, so this can also be written

$$h_i = \frac{d_i^2(\boldsymbol{T}_i\cdot\boldsymbol{V}_{i-1})}{d_{i-1}^2(\boldsymbol{T}_i\cdot\boldsymbol{V}_i)}h_{i-1} = \frac{(\boldsymbol{V}_i\cdot\boldsymbol{V}_i)(\boldsymbol{T}_i\cdot\boldsymbol{V}_{i-1})}{(\boldsymbol{V}_{i-1}\cdot\boldsymbol{V}_{i-1})(\boldsymbol{T}_i\cdot\boldsymbol{V}_i)}h_{i-1}$$

This is a very efficient computation since it involves no square roots or trigonometric functions.

The first parameter value w_0 and the first increment $h_0 = w_1 - w_0$ are arbitrary, but, for definiteness, their values are set using the `base_parameter` and `base_scale` properties of the chart, respectively. Once these are established, all other h_i and w_i values can be calculated recursively from the formula above.

The pseudocode to implement this is as follows:

```
// Calculates parameter values at chart points
// Input:
//    P -- array of chart points
//    T -- array of (unit) tangents at chart points
// Returns
//    Array of parameter values to be assigned to chart points
double[] ChartParameters(Position[] P, Vector[] T)
{
   int n = P.Length;

   double[] w  = new double[n];        // parameter values at chart points
   double[] h  = new double[n-1];      // parameter increments between chart points
   Vector[] V  = new Vector[n-1];      // chord vectors
   double[] c2 = new double[n-1];      // chord lengths (squared)

   w[0] = base_parameter;
   V[0] = P[1] - P[0];
   d2[0] = V[0] * V[0];
   h[0] = base_scale;
   w[1] = w[0] + h[0];

   for (int i = 1 ; i < n-1 ; i++)
   {
      V[i]  = P[i+1] - P[i];                    // chord vector
      d2[i] = V[i] * V[i];                      // squared length of chord V[i]
      double numer = d2[i]*(T[i] * V[i-1]);
      double denom = d2[i-1]*(T[i] * V[i]);
      h[i] = (numer/denom) * h[i-1];
      w[i+1] = w[i] + h[i];
   }

   return w;
}
```

This code is obviously very inefficient — several of the arrays we used are not really necessary. It is written to correspond closely with the mathematical description above, so the goal is clarity rather than efficiency.

M.4.3 Computing a Point & Tangent on an Intersection Curve

Now that the chart points are fully populated, we can describe the procedure for calculating a point and tangent at a given parameter value on our icurve.

M.4.3.1 Equations for the Point on the Intersection Curve

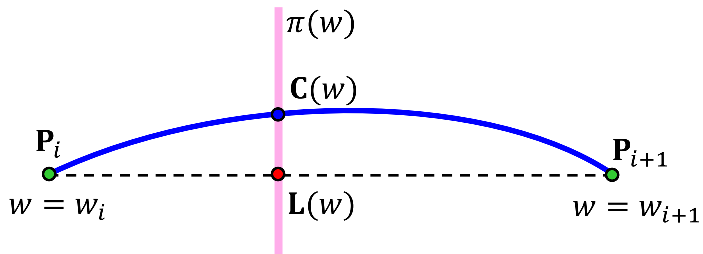
So, assume we have chart points $\mathbf{P}_0, \ldots, \mathbf{P}_n$ where the point $\mathbf{P}_i$ has curve parameter w_i and surface parameters (s_i, t_i) and (u_i, v_i). We want to calculate a point on the curve at a given parameter value w which we assume to lie in the range $w_0 \le w \le w_n$. Therefore, we can find an index i such that $w_i \le w \le w_{i+1}$.

From section L.4.1, we recall that the chord $\mathbf{P}_i\mathbf{P}_{i+1}$ has the equation

$$\boldsymbol{L}(w) = \frac{w_{i+1} - w}{w_{i+1} - w_i}\boldsymbol{P}_i + \frac{w - w_i}{w_{i+1} - w_i}\boldsymbol{P}_{i+1}$$

We construct the plane $\pi(w)$ that passes through the point $\mathbf{L}(w)$ and is perpendicular to the chord $\mathbf{P}_i\mathbf{P}_{i+1}$. A 3D point $\mathbf{X}$ lies on this plane iff $\big(\mathbf{X} - \mathbf{L}(w)\big) \cdot (\mathbf{P}_{i+1} - \mathbf{P}_i) = 0$.

The point $\mathbf{C}(w)$ of the intersection curve is the intersection of the two surfaces $\mathbf{A}$ and $\mathbf{B}$ and the plane $\pi(w)$, as shown in the picture below



To compute the point $\mathbf{C}(w)$, we have to find parameter values s, t, u, v that satisfy the equations

$$\boldsymbol{A}(s,t) = \boldsymbol{B}(u,v)$$

$$\big(\boldsymbol{A}(s,t) - \boldsymbol{L}(w)\big) \cdot (\boldsymbol{P}_{i+1} - \boldsymbol{P}_i) = 0$$

The first equation says that we have two points (one on each surface) that are identical, and the second equation says that one (and hence both) of these points lies on the plane $\pi(w)$. If we write

$$\begin{aligned} f_1(s,t,u,v) &= \boldsymbol{A}_x(s,t) - \boldsymbol{B}_x(u,v) \\ f_2(s,t,u,v) &= \boldsymbol{A}_y(s,t) - \boldsymbol{B}_y(u,v) \\ f_3(s,t,u,v) &= \boldsymbol{A}_z(s,t) - \boldsymbol{B}_z(u,v) \\ f_4(s,t,u,v) &= \big(\boldsymbol{A}(s,t) - \boldsymbol{L}(w)\big) \cdot (\boldsymbol{P}_{i+1} - \boldsymbol{P}_i) \end{aligned}$$

then our problem reduces to finding the common roots s, t, u, v of the four equations $f_i(u,v,s,t) = 0$ $(i = 1,2,3,4)$. Once we have done this, we can also obtain the icurve tangent at the parameter value w: it will be in the direction of the cross product of the two surface normals:

$$\left(\frac{\partial \boldsymbol{A}}{\partial s} \times \frac{\partial \boldsymbol{A}}{\partial t}\right) \times \left(\frac{\partial \boldsymbol{B}}{\partial u} \times \frac{\partial \boldsymbol{B}}{\partial v}\right)$$

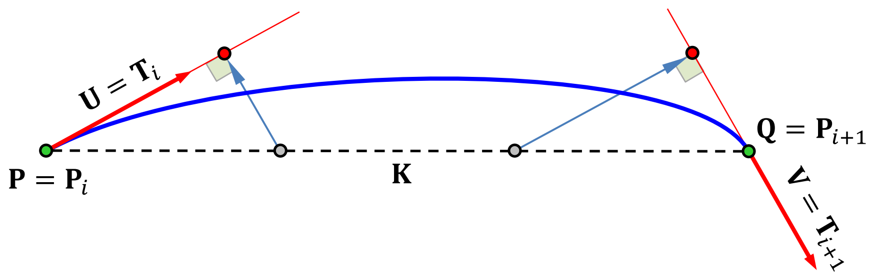
The four equations above are non-linear, of course, so they will have to be solved by an iterative numerical method. Again, there is nothing special about the root finder used in Parasolid. Suitable functions can be found in the Numerical Recipes book [7], in the GNU Scientific Library [8], or in numerous other places. Specifically, the function `newt` from the Numerical Recipes book has the required functionality.

There is no reason to fear that an iterative numerical procedure will deliver answers that are less accurate than a closed form formula. In fact, common closed-form formulas will deliver inaccurate answers if implemented poorly [10]. Accuracy depends on careful coding (avoiding loss of precision) rather than on the choice between numerical or analytical methods.

However, as in all numerical root-finding, the key is to find a good starting point. The recommended technique for doing this is described in the next section.

M.4.3.2 Estimating a Starting Point

To simplify notation, let's define $\mathbf{P} = \mathbf{P}_i$, $\mathbf{Q} = \mathbf{P}_{i+1}$, $\mathbf{U} = \mathbf{T}_i$, $\mathbf{V} = \mathbf{T}_{i+1}$, $\mathbf{K} = \mathbf{Q} - \mathbf{P}$.



We are going to construct a cubic Bezier curve that approximates the icurve and can be used to generate starting points for our iteration. The first and fourth poles of the Bezier curve will obviously be **P** and **Q** respectively. To get the second pole, we project the grey point $\mathbf{P} + ⅓\mathbf{K}$ onto the tangent line at **P**, as shown in the diagram. The result is the red point, which is $\mathbf{P} + ⅓(\mathbf{K} \cdot \mathbf{U})\mathbf{U}$. Similarly, for the third pole, we use the point $\mathbf{Q} - ⅓(\mathbf{K} \cdot \mathbf{V})\mathbf{V}$. So, in summary, our Bezier curve has poles, **P**, $\mathbf{P} + ⅓(\mathbf{K} \cdot \mathbf{U})\mathbf{U}$, $\mathbf{Q} - ⅓(\mathbf{K} \cdot \mathbf{V})\mathbf{V}$, and **Q**, so its equation is:

$$\boldsymbol{C}(r) = (1-r)^3\,\boldsymbol{P} + 3r(1-r)^2\left(\boldsymbol{P} + \frac{1}{3}\,(\boldsymbol{K} \cdot \boldsymbol{U})\boldsymbol{U}\right) + 3r^2\,(1-r)\left(\boldsymbol{Q} - \frac{1}{3}\,(\boldsymbol{K} \cdot \boldsymbol{V})\boldsymbol{V}\right) + r^3\,\boldsymbol{Q}$$

where, as before, r is the ratio

$$r = \frac{w - w_i}{w_{i+1} - w_i}$$

In fact, it is more convenient to express this cubic in Hermite form. If we define so-called blending functions h_0, h_1, k_0, k_1 by

$$h_0(t) = 1 - 3t^2 + 2t^3$$
$$h_1(t) = 3t^2 - 2t^3$$
$$k_0(t) = t - 2t^2 + t^3$$
$$k_1(t) = t^3 - t^2$$

Then the cubic curve can be written in the somewhat tidier form

$$\boldsymbol{C}(r) = h_0(r)\boldsymbol{P} + h_1(r)\boldsymbol{Q} + k_0(r)(\boldsymbol{K} \cdot \boldsymbol{U})\boldsymbol{U} + k_1(r)(\boldsymbol{K} \cdot \boldsymbol{V})\boldsymbol{V}$$

With either formula, it is easy to check that $\mathbf{C}(0) = \mathbf{P}$, $\mathbf{C}(1) = \mathbf{Q}$, and

$$\frac{d\boldsymbol{C}}{dr}(0) = (\boldsymbol{K} \cdot \boldsymbol{U})\boldsymbol{U} \quad ; \quad \frac{d\boldsymbol{C}}{dr}(1) = (\boldsymbol{K} \cdot \boldsymbol{V})\boldsymbol{V}$$

The basic function for performing this cubic interpolation is as follows:

```
// Calculate a point on a Hermite cubic curve
// Input:
//   P0 -- start point of curve
//   P1 -- end point of curve
//   V0 -- first derivative vector at start point
//   V1 -- first derivative vector at end point
//   t -- parameter value at which to evaluate (0 <= t <= 1)
// Returns
//   Point on cubic curve that intepolates P, Q ,U, V

static Position HermiteCubic(Point P0, Point P1, Vector V0, Vector V1, double t)
{
   double t2 = t*t;
   double t3 = t*t2;

   double  h0 = 1 - 3*t2 + 2*t3;
   double  h1 = 3*t2 - 2*t3;
   double  k0 = t - 2*t2 + t3;
   double  k1 = t3 - t2;

   return h0*P0 + h1*P1 + k0*V0 + k1*V1;
}
```

This function is used to compute an estimated point on an icurve as follows:

```
// Define variable names as in math notes above
Point P = chart[i].CurvePoint;     Vector U = chart[i+1].CurveTangent;
Point Q = chart[i].CurvePoint;     Vector V = chart[i+1].CurveTangent;

//Get parameter values. We assume that  wi < w < wip1
double wi   = chart[i].w;
double wip1 = chart[i+1].w;

// Compute the ratio r
double r = (w - wi)/(wip1 - wi);

// Chord and derivative vectors
Vector K = Q - P;
Vector KUU = (K*U)*U;
Vector KVV = (K*V)*V;

// Calculate point Cr =C(r) on cubic approximation
Position Cr = HermiteCubic(P, Q, KUU, KVV, r);
```

From this point $\mathbf{C}(r)$, we can obtain surface parameter values (s,t) and (u,v) using the techniques described in section L.4.2.1. We step along tangent planes to get rough estimates, and then refine these estimates by iterative numerical methods. If $r < 0.5$, we use the tangent plane at $\mathbf{P}_i$, and if $r \geq 0.5$, we use the tangent plane at $\mathbf{P}_{i+1}$.

Then we use these (s,t) and (u,v) values as the starting point in an iterative numerical root finder, as described in section L.4.3.1 .

M.4.4 Approximating an Intersection Curve

The approximation process described here is completely generic — it can be applied to any curve for which points and first derivative vectors can be computed. The procedure has no relationship to JT/XT icurves, specifically.

The basic idea is to start with cubic segments constructed from set of points on the curve (the chart points, in the case of an icurve), and then add intermediate points to split these cubic segments as needed. The splitting is continued until the cubic segments are close enough to the original icurve. An implementation of this idea is provided in section L.4.4.4.

M.4.4.1 Fitting Hermite Cubics

At the i-th chart point, we know the location $\mathbf{P}_i$, the unit tangent vector $\mathbf{T}_i$, and the parameter value w_i. As in section L.4.2.4, let $\mathbf{V}_i = \mathbf{P}_{i+1} - \mathbf{P}_i$ denote the chord vector, and let $h_i = w_{i+1} - w_i$. Then, from the discussion in section L.4.3, we know that the icurve's first derivative vector at $\mathbf{P}_i$ is given by

$$\frac{d\boldsymbol{C}}{dw}(w = w_i) = \frac{\|\boldsymbol{V}_i\|}{h_i \cos\alpha_i}\boldsymbol{T}_i = \frac{(\boldsymbol{V}_i \cdot \boldsymbol{V}_i)}{h_i(\boldsymbol{V}_i \cdot \boldsymbol{T}_i)}\boldsymbol{T}_i$$

Similarly, at the other end of the segment,

$$\frac{d\boldsymbol{C}}{dw}(w = w_{i+1}) = \frac{(\boldsymbol{V}_i \cdot \boldsymbol{V}_i)}{h_{i+1}(\boldsymbol{V}_i \cdot \boldsymbol{T}_{i+1})}\boldsymbol{T}_{i+1}$$

Note that this last equation holds true even if we have not made the special choices for the w_i values described in to make the icurve C1 continuous.

We can now use this data to construct a Hermite cubic approximation **H** of the icurve on the interval $w_i \leq w \leq w_{i+1}$. The relevant Hermite cubic construction is given in the function HermiteCubic.

Next we check to see if the Hermite cubic segment **H** is sufficiently close to the icurve **C** throughout the interval $w_i \leq w \leq w_{i+1}$; details of the method for doing this are described in the next section. If **H** and **C** are sufficiently close, then we're done; if not, we compute a point and tangent at some interior location in the interval $\bar{w} \in [w_i, w_{i+1}]$, and we treat the two intervals $[w_i, \bar{w}]$ and $[\bar{w}, w_{i+1}]$ as described above.

Continuing this testing and splitting process recursively, we eventually obtain a sequence of cubic segments that form an adequate approximation of the icurve (to within the caller-specified tolerance). Then we use well-known techniques to join these cubic segments together to get a cubic b-spline, as explained in section 5.3 below.

M.4.4.2 Estimating Approximation Error

The key to the process described in the previous section is the ability to measure the error between the original icurve **C** and a cubic approximation **H** on some interval $[w_i, w_{i+1}]$. If the approximation error is larger than some caller-specified value ε, then we have to split the segment into two and approximate each of the two pieces separately. There are many ways to measure the approximation error, and the choice we make will be a trade-off between certainty and performance.

If we were being rigorous, we would calculate either the "parametric" error

$$\text{error} = \max_{w_i \leq w \leq w_{i+1}} \|\boldsymbol{C}(w) - \boldsymbol{H}(w)\|$$

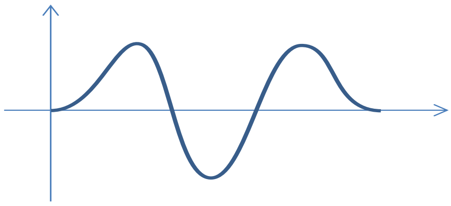
or perhaps even the "geometric" (Hausdorff) error

$$\text{error} = \max_{w_i \leq s \leq w_{i+1}} \; \min_{w_i \leq t \leq w_{i+1}} \|\boldsymbol{C}(s) - \boldsymbol{H}(t)\|$$

However, both of these are very difficult to calculate, especially the geometric error, so a simpler estimate is needed. If our estimate is overly pessimistic, we may generate more cubic segments than we really need, but this is not a great disaster. A more realistic approach is to choose some n sample locations $\bar{w}_1, \ldots, \bar{w}_n$ in $[w_i, w_{i+1}]$ and check the error only at these points. In other words, our error estimate is:

$$\text{error} = \max_{1 \leq i \leq n} \|\boldsymbol{C}(\bar{w}_i) - \boldsymbol{H}(\bar{w}_i)\|$$

Using larger values of n will reduce performance but will provide a more reliable approximation, of course. The simplest approach is to choose $n = 1$, which means that we simply use one sample location at $\bar{w} = ½(w_i + w_{i+1})$. This is especially fast because the value of $\mathbf{C}(w)$ used in measuring the error can be re-used if we find that we need to split the segment into two. Our prior experiences (and perhaps some theorems in approximation theory) tell us that the error when approximating by a Hermite cubic typically looks like this:

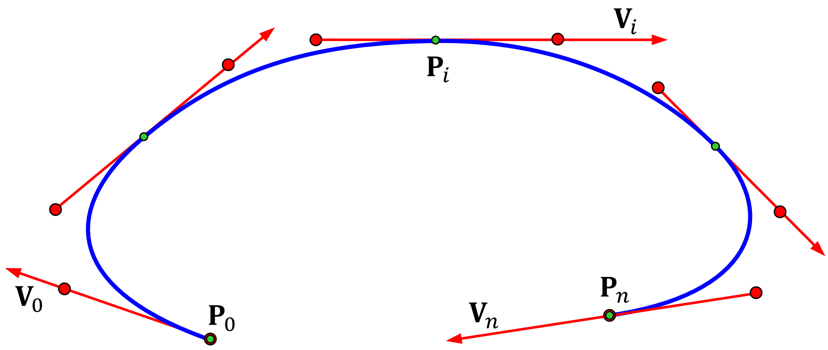


and this suggests checking the error at the values ⅓, ½, ⅔. This test is implemented in the function MeasureError in section L.4.4.4.

M.4.4.3 Constructing a Cubic B-spline

Suppose that for $i = 0,1,\dots,n$ we have a point $\mathbf{P}_i$, a derivative vector $\mathbf{V}_i$, and a parameter value w_i, and we wish to interpolate this data with a cubic b-spline. In other words, we want to construct a cubic b-spline **S** such that $\mathbf{S}(w_i) = \mathbf{P}_i$ and $\mathbf{S}'(w_i) = \mathbf{V}_i$ for $i = 0,1,\dots,n$.

Geometrically, the construction is very simple, as shown in the following diagram:



The small green points are the original points $\mathbf{P}_i$, and the red points are the poles (control points) of the cubic b-spline. The construction algorithm is as follows:

- The first pole is $\mathbf{P}_0$
- The second pole is at $\mathbf{P}_0 + \frac{1}{3}(w_1 - w_0)\mathbf{V}_0$
- Near the *i*-th point $\mathbf{P}_i$, there are two poles at $\mathbf{P}_i - \frac{1}{3}(w_i - w_{i-1})\mathbf{V}_i$ and $\mathbf{P}_i + \frac{1}{3}(w_{i+1} - w_i)\mathbf{V}_i$
- The last-but-one pole is at $\mathbf{P}_n - \frac{1}{3}(w_n - w_{n-1})\mathbf{V}_n$
- The last pole is at $\mathbf{P}_n$

The spline's knot sequence is also easy to construct. It is

$$w_0, w_0, w_0, w_0, w_1, w_1, w_2, w_2, \dots, w_i, w_i, \dots, w_{n-1}, w_{n-1}, w_n, w_n, w_n, w_n$$

As we can see, each interior knot has multiplicity =2, and the end knots have multiplicity =4. This technique for creating a spline is implemented in a Spline constructor in section L.4.4.

M.4.4.4 Approximation Code

This section provides a complete implementation of the type of approximation function described in section above. The function is completely generic and can be applied to any curve on which we can calculate points and first-derivative vectors.

As before, the code emphasizes clarity rather than efficiency.

The approximation function is CubicApproximation.Approximate. The Main function calls this function to construct an approximation of a curve described by an evaluator function IntEval. This is just a “fake” evaluator function to illustrate the process. In the usage that is of immediate interest to us here, the real evaluator function would actually return points and derivatives of an icurve, using the methods described in section L.4.2.3. The approximation process is begun from a sequence of Point/Derivative quantities, which would be read from the chart of the icurve.

The code uses *Vector* and *Position* classes with obvious properties, as described in section L.3. You have to provide these classes in order for the code to work.

The code is as follows:

```
using System.Collections.Generic;     // For List class

public class MyProgram
{
   static void Main()
   {
      // Data to be passed to the curve evaluator function (cylinder radii, in this case)
      double a = 150;
      double b = 200;
      double[] data = { a, b };

      // Evaluator function for a cylinder/cylinder intersection curve.
      CurveEvaluator eval = IntEval;

      // Get initial points to start approximation process
      double pi = System.Math.PI;
      double t0 = 0;            CurvePoint pv0 = eval(data, t0);
      double t1 = pi/4;         CurvePoint pv1 = eval(data, t1);
      double t2 = pi/2;         CurvePoint pv2 = eval(data, t2);
      CurvePoint[] initPoints = { pv0, pv1, pv2 };

      // Define the desired approximation tolerance
      double tol = 0.0015;

      // Calculate a cubic spline approximation
      Spline approx = CubicApproximation.Approximate(eval, data, initPoints, tol);

      // Write out spline data to console
      approx.Write();
   }

   /// <summary>Sample evaluator function -- for an intersection curve</summary>
   private static CurvePoint IntEval(object data, double t)
   {
      double[] ab = (double[])data;
      double a = ab[0];     // Smaller radius; cylinder x^2 + y^2 = a^2
      double b = ab[1];     // Larger radius;  cylinder y^2 + z^2 = b^2

      double cost = System.Math.Cos(t);
      double sint = System.Math.Sin(t);

      double x = a*cost;
      double y = a*sint;
      double z = System.Math.Sqrt(b*b - y*y);
      Position pt = new Position(x, y, z);

      double dx = -a*sint;
      double dy = a*cost;
      double dz = -(a*a*cost*sint) / System.Math.Sqrt(b*b - a*a*sint*sint);
      Vector deriv = new Vector(dx, dy, dz);

      return new CurvePoint(pt, deriv, t);
   }
}
```

```
/// <summary>Provides functions for approximating a given curve by a cubic
spline</summary>
public class CubicApproximation
{
   /// <summary>
   /// Constructs a cubic b-spline approximation of a curve, starting from initial
CurvePoint data
   /// </summary>
   /// <param name="eval">Evaluator function to calculate point/derivative on curve to be
approximated</param>
   /// <param name="data">Data to be passed to the evaluator function</param>
   /// <param name="initPoints">Initial list of CurvePoint objects to start
approximation</param>
   /// <param name="tol">The tolerance for the approximation process</param>
   /// <returns>A cubic b-spline approximating the given curve to within the given
tolerance</returns>
   /// <remarks>
   /// This function is suitable for approximating a JT/XT icurve. The chart points of
the icurve
   /// provide the initial list of CurvePoint items that this function receives.
   /// </remarks>
   public static Spline Approximate(CurveEvaluator eval, object data, CurvePoint[]
initPoints, double tol)
   {
      // The list of points that will be used to construct the spline approximation
      List<CurvePoint> outPoints = new List<CurvePoint>();

      // Add the first point to the list
      outPoints.Add(initPoints[0]);





      // Cycle through inter-point intervals, adding split points
      for (int i = 0 ; i <= initPoints.Length - 2 ; i++)
      {
         CurvePoint[] splitPoints = TestAndSplit(eval, data, tol, initPoints[i],
initPoints[i+1]);
         outPoints.AddRange(splitPoints);
      }

      // Create a cubic spline from the CurvePoint data
      return new Spline(outPoints.ToArray());
   }

   /// <summary>Recursively tests/splits a Hermite cubic segment until approximation
tolerance is met</summary>
   /// <param name="eval">Evaluator function to calculate point/derivative on curve to be
approximated</param>
   /// <param name="data">Data to be passed to the evaluator function</param>
   /// <param name="tol">Tolerance to be used to test whether approximation is good
enough</param>
   /// <param name="ends">The two ends of the Hermite segment to be tested</param>
   /// <returns>Array of CurvePoint items that give a good piecewise cubic
approximation</returns>
   private static CurvePoint[] TestAndSplit(CurveEvaluator eval, object data, double tol,
params CurvePoint[] ends)
   {
      // The list of CurvePoints that we will eventually output
      List<CurvePoint> outPoints = new List<CurvePoint>();

      CurvePoint pv0 = ends[0];     double t0 = pv0.T;
      CurvePoint pv1 = ends[1];     double t1 = pv1.T;

      double error = MeasureError(eval, data, ends);

      if (error < tol)  // Success, so just add end-point to pointList
```

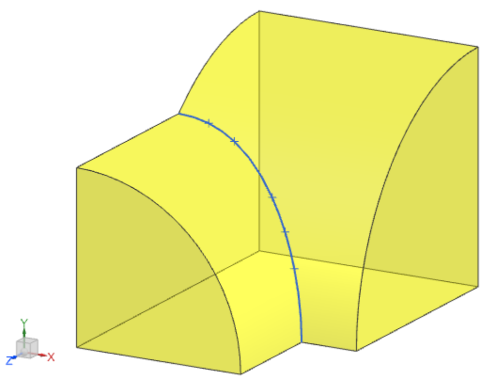
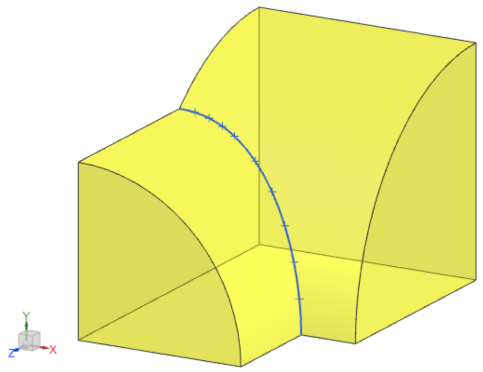
```
      {
         outPoints.Add(pv1);
      }
      else   // Split segment and procees two halves recursively
      {
         double tmid = (t0 + t1) / 2;
         CurvePoint splitPoint = eval(data, tmid);
         CurvePoint[] newPoints1 = TestAndSplit(eval, data, tol, pv0, splitPoint);
         outPoints.AddRange(newPoints1);
         CurvePoint[] newPoints2 = TestAndSplit(eval, data, tol, splitPoint, pv1);
         outPoints.AddRange(newPoints2);
      }

      return outPoints.ToArray();
   }

   /// <summary>Measures the error between a curve and a Hermite cubic
approximation</summary>
   /// <param name="eval">Evaluator function to calculate point/derivative on curve to be
approximated</param>
   /// <param name="data">Data to be passed to the evaluator function</param>
   /// <param name="ends">The two ends of the Hermite segment to be tested</param>
   /// <returns>An estimate of the error (deviation) between the curve and the
cubic</returns>
   /// <remarks>
   /// The error estimate is quite simplistic. Much more accurate estimates could be
computed,
   /// but this would also be much slower.
   /// </remarks>
   private static double MeasureError(CurveEvaluator eval, object data, params
CurvePoint[] ends)
   {
      Position P0 = ends[0].Point;     Vector V0 = ends[0].Derivative;     double t0 =
ends[0].T;
      Position P1 = ends[1].Point;     Vector V1 = ends[1].Derivative;     double t1 =
ends[1].T;

      double c = t1 - t0;
      double t;
      Position curvePoint;
      Position cubicPoint;
      double error, error1, error2;

      t = t0 + c/3;
      curvePoint = eval(data, t).Point;
      cubicPoint = HermiteCubic(P0, P1, V0, V1, t0, t1, t);
      error = Position.Distance(curvePoint, cubicPoint);

      t = t0 + c/2;
      curvePoint = eval(data, t).Point;
      cubicPoint = HermiteCubic(P0, P1, V0, V1, t0, t1, t);
      error1 = Position.Distance(curvePoint, cubicPoint);
      error = System.Math.Max(error, error1);

      t = t0 + 2*c/3;
      curvePoint = eval(data, t).Point;
      cubicPoint = HermiteCubic(P0, P1, V0, V1, t0, t1, t);
      error2 = Position.Distance(curvePoint, cubicPoint);
      error = System.Math.Max(error, error2);

      return error;
   }


   /// <summary>Calculate a point on a Hermite cubic curve</summary>
   /// <param name="P0">Start point</param>
   /// <param name="P1">End point</param>
   /// <param name="V0">Start first derivative</param>
```

```
   /// <param name="V1">End first derivative</param>
   /// <param name="t0">Start parameter</param>
   /// <param name="t1">End parameter</param>
   /// <param name="t">Parameter value at which to evaluate</param>
   /// <returns>Point on curve</returns>
   private static Position HermiteCubic(Position P0,Position P1,Vector V0,Vector
V1,double t0,double t1,double t)
   {
      double c = t1 - t0;
      double u = (t - t0)/c;
      double u2 = u*u;
      double u3 = u*u2;

      double h0 = 1 - 3*u2 + 2*u3;
      double h1 = 1 - h0;
      double k0 = u - 2*u2 + u3;
      double k1 = u3 - u2;

      return h0*P0 + h1*P1 + (k0*c)*V0 +(k1*c)*V1;
   }
}

/// <summary>Represents a location on a curve</summary>
public class CurvePoint
{
   /// <summary>Point location</summary>
   public Position Point { get; set; }

   /// <summary>Derivative vector</summary>
   public Vector Derivative { get; set; }

   /// <summary>Parameter value</summary>
   public double T { get; set; }

   /// <summary>Constructor</summary>
   /// <param name="point">Point</param>
   /// <param name="deriv">Derivative vector</param>
   /// <param name="t">Parameter value</param>
   public CurvePoint(Position point, Vector deriv, double t)
   {
      this.Point = point;
      this.Derivative = deriv;
      this.T = t;
   }
}

/// <summary>Represents a polynomial b-spline</summary>
public class Spline
{
   /// <summary>The poles (control points) of the spline</summary>
   public Position[] Poles { get; set; }

   /// <summary>The knots of the spline</summary>
   public double[] Knots { get; set; }

   /// <summary>Constructs a cubic spline from an array of CurvePoint objects</summary>
   /// <param name="points">The Hermite points</param>
   /// <remarks>
   /// The cubic spline interpolates the point and derivative values stored in the
CurvePoint array.
   /// </remarks>
   public Spline(CurvePoint[] points)
   {
      int npts = points.Length;

      List<double> knotList = new List<double>();
      List<Position> poleList = new List<Position>();
```

```
      Position P;
      Vector V;
      double h, k;

      // Initial knot has multiplicity = 4
      for (int i = 1; i <= 4; i++) knotList.Add(points[0].T);

      // Construct first two poles
      P = points[0].Point;
      V = points[0].Derivative;
      k = points[1].T - points[0].T;
      poleList.Add(P);
      poleList.Add(P + k*V/3);


      //Cycle through interior knots
      for (int i = 1; i < npts - 1; i++)
      {
         P = points[i].Point;
         V = points[i].Derivative;
         h = points[i].T - points[i - 1].T;
         k = points[i + 1].T - points[i].T;
         poleList.Add(P - h*V/3);
         poleList.Add(P + k*V/3);
         knotList.Add(points[i].T);
         knotList.Add(points[i].T);
      }

      // Construct last two poles
      P = points[npts - 1].Point;
      V = points[npts - 1].Derivative;
      h = points[npts - 1].T - points[npts - 2].T;
      poleList.Add(P - h*V/3);
      poleList.Add(P);

      //Last knot has multiplicity = 4
      for (int i = 1; i <= 4; i++) knotList.Add(points[npts - 1].T);

      this.Knots = knotList.ToArray();
      this.Poles = poleList.ToArray();
   }

   /// <summary>Write out spline knots and poles to the console</summary>
   public void Write()
   {
      System.Console.WriteLine("Knot values");
      System.Console.WriteLine("===========");

      for (int i = 0; i < this.Knots.Length; i++)
      {
         System.Console.WriteLine("t[{0}] = {1}", i, this.Knots[i].ToString("F8"));
      }

      System.Console.WriteLine("");
      System.Console.WriteLine("Poles (Control Points)");
      System.Console.WriteLine("======================");

      for (int i = 0; i < this.Poles.Length; i++)
      {
         System.Console.Write("p[{0}] = ( ", i);
         System.Console.Write(this.Poles[i].X.ToString("F8") + " , ");
         System.Console.Write(this.Poles[i].Y.ToString("F8") + " , ");
         System.Console.Write(this.Poles[i].Z.ToString("F8") + " )\n");
      }

      System.Console.ReadLine();
   }
}
```

```
/// <summary>
/// An evaluator function that calculates position and first derivative at a parameter
value on a curve
/// </summary>
/// <param name="data">Data to be used in evaluation</param>
/// <param name="t">Parameter value at which to evaluate</param>
/// <returns>PointVector containing the calculated point and first derivative</returns>
/// <remarks>
/// You write a CurveEvaluator function to provide information
/// about a curve that you're approximating with a spline.
/// </remarks>
public delegate CurvePoint CurveEvaluator(object data, double t);
```

Running the code with tolerance values of 0.02 and 0.0015 produces the following results:

<table>
<tr><td>Tolerance = 0.02</td><td>Tolerance = 0.0015</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Knot values
===========
t[0] = 0.00000000
t[1] = 0.00000000
t[2] = 0.00000000
t[3] = 0.00000000
t[4] = 0.39269908
t[5] = 0.39269908
t[6] = 0.58904862
t[7] = 0.58904862
t[8] = 0.78539816
t[9] = 0.78539816
t[10] = 1.17809725
t[11] = 1.17809725
t[12] = 1.37444679
t[13] = 1.37444679
t[14] = 1.57079633
t[15] = 1.57079633
t[16] = 1.57079633
t[17] = 1.57079633

Poles (Control Points)
======================
p[0] = (150.00000000 , 0.00000000 , 200.00000000)
p[1] = (150.00000000 , 19.63495408 , 200.00000000)
p[2] = (146.09590150 , 39.26218265 , 197.02054165)
p[3] = (134.82494406 , 66.47268096 , 188.86777357)
p[4] = (130.17473985 , 75.17260112 , 185.55245365)
p[5] = (119.26614384 , 91.49846878 , 178.06926525)
p[6] = (113.00802177 , 99.12401259 , 173.90077364)
p[7] = (92.18200800 , 119.95002636 , 160.87320146)</td><td>Knot values
===========
t[0] = 0.00000000
t[1] = 0.00000000
t[2] = 0.00000000
t[3] = 0.00000000
t[4] = 0.19634954
t[5] = 0.19634954
t[6] = 0.39269908
t[7] = 0.39269908
t[8] = 0.58904862
t[9] = 0.58904862
t[10] = 0.78539816
t[11] = 0.78539816
t[12] = 0.98174770
t[13] = 0.98174770
t[14] = 1.17809725
t[15] = 1.17809725
t[16] = 1.27627202
t[17] = 1.27627202
t[18] = 1.37444679
t[19] = 1.37444679
t[20] = 1.47262156
t[21] = 1.47262156
t[22] = 1.57079633
t[23] = 1.57079633
t[24] = 1.57079633
t[25] = 1.57079633

Poles (Control Points)
======================</td></tr>
</table>

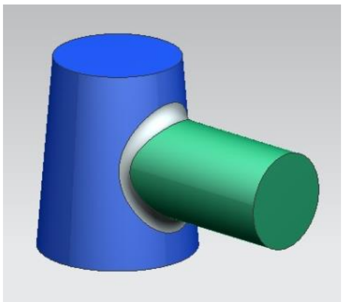
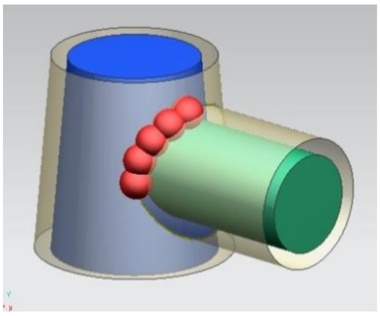
p[8] = (75.54284706 , 131.06795825 , 151.42586520)	p[0] = (150.00000000 , 0.00000000 , 200.00000000)
p[9] = (48.33234875 , 142.33891569 , 140.59439414)	p[1] = (150.00000000 , 9.81747704 , 200.00000000)
p[10] = (38.89238528 , 145.20249730 , 137.56536054)	p[2] = (149.03308682 , 19.63471133 , 199.27172642)
p[11] = (19.63471133 , 149.03308682 , 133.40589446)	p[3] = (145.20249730 , 38.89238528 , 196.42333163)
p[12] = (9.81747704 , 150.00000000 , 132.28756555)	p[4] = (142.33891569 , 48.33234875 , 194.30295229)
p[13] = (0.00000000 , 150.00000000 , 132.28756555)	p[5] = (134.82494406 , 66.47268096 , 188.86777357)
	p[6] = (130.17473985 , 75.17260112 , 185.55245365)
	p[7] = (119.26614384 , 91.49846878 , 178.06926525)
	p[8] = (113.00802177 , 99.12401259 , 173.90077364)
	p[9] = (99.12401259 , 113.00802177 , 165.21572552)
	p[10] = (91.49846878 , 119.26614384 , 160.69930611)
	p[11] = (75.17260112 , 130.17473985 , 151.99742483)
	p[12] = (66.47268096 , 134.82494406 , 147.81537485)
	p[13] = (52.86743180 , 140.46042278 , 142.39963932)
	p[14] = (48.24007148 , 142.11611878 , 140.73804072)
	p[15] = (38.84533170 , 144.96598194 , 137.80076707)
	p[16] = (34.07796679 , 146.16014468 , 136.52549402)
	p[17] = (24.44912982 , 148.07543944 , 134.44576098)
	p[18] = (19.58767265 , 148.79656849 , 133.64169964)
	p[19] = (9.81746945 , 149.75884951 , 132.56247459)
	p[20] = (4.90873852 , 150.00000000 , 132.28756555)
	p[21] = (0.00000000 , 150.00000000 , 132.28756555)

In some systems (like NX), a b-spline knot sequence is expected to start with a value of 0, and end with a value of 1. To obtain a knot sequence of this form, you can just divide each of the above knots by 1.57079633. This will not change the shape of the curve.

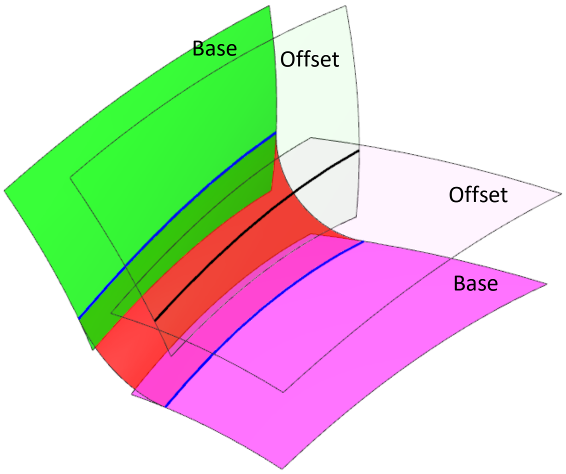
M.5 Rolling-Ball Blend Surface

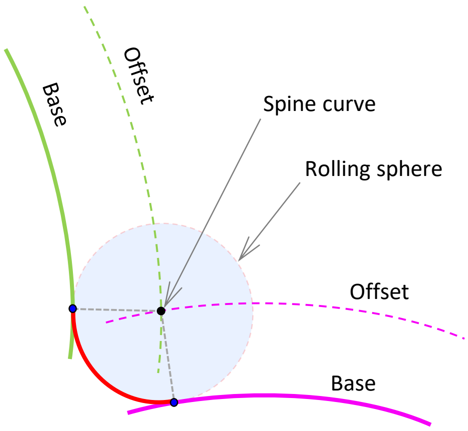
M.5.1 Computing a Point on a Blend Surface

Conceptually, a blend surface is the envelope of a spherical ball that is rolling across two base surfaces, staying in contact with both of them, as shown here:

A more detailed depiction of the construction is shown in the pictures below.





The curves shown in blue are the contact curves, where the rolling ball touches the base surfaces. Equivalently, we can think of the ball moving so that its center lies on a “spine” curve that is the intersection of two offset surfaces.

Whenever possible, simple analytic surfaces are used to perform blending operations, rather than using blend surfaces. So, many blends are modeled using cylindrical, toroidal or spherical surfaces. At the other extreme, very complex blends (e.g. varying radius ones, or ones with non-circular cross sections) are modeled using b-spline surfaces.

M.5.1.1 Stored Data

The stored data for a blend surface is as follows:

Field name	Data type	Description
type	char	Type of blend: ‘R’ or ‘E’ (see below)
surface	pointer[2]	Base surfaces (adjacent to original edge)
spine	pointer	Spine curve of the blend (locus of center of rolling ball)
range	double[2]	Offset distances to be applied to the base surfaces
thumb_weight	double[2]	Always [1,1]
boundary	pointer0[2]	Always [0, 0]
start	pointer0	Start LIMIT in certain degenerate cases
end	pointer0	End LIMIT in certain degenerate cases

The thumb-weight and boundary fields are obsolete, and can be ignored.

The two entries in the range array always have equal magnitudes, and this common magnitude is the radius of the blend surface.

M.5.1.2 Surface Equation

The equation of a rolling-ball blend surface is:

$$\boldsymbol{S}(u,v) = \boldsymbol{C}(u) + r\cos\big(v\alpha(u)\big)\,\boldsymbol{X}(u) + r\sin\big(v\alpha(u)\big)\,\boldsymbol{Y}(u)$$

where

- $\boldsymbol{C}(u)$ is the spine curve
- r is the blend radius
- $\mathbf{X}(u)$ and $\mathbf{Y}(u)$ are unit vectors that are perpendicular to the tangent vector $\mathbf{C}'(u)$, and to each other. $\mathbf{X}(u)$ is in the direction $\mathbf{S}(u,0) - \mathbf{C}(u)$.

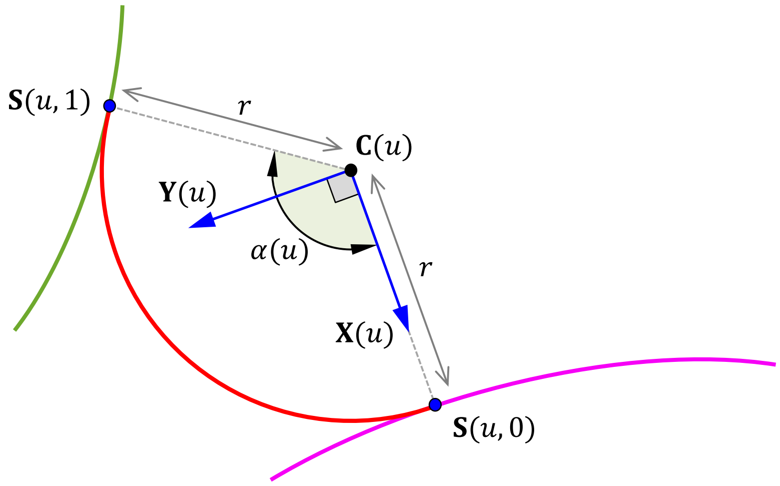
- $\alpha(u)$ is the angle subtended by points on the boundary curves at the spine

X, **Y** and α are expressed as functions of u, since their values change with u.

We note the following basic facts that are apparent from the parametric equation:

- For all u and v, we have $\|\mathbf{S}(u,v) - \mathbf{C}(u)\| = r$
- Setting $v = 0$, we get $\mathbf{S}(u,0) = \mathbf{C}(u) + r\mathbf{X}(u)$, which is a curve lying on the first base surface
- Setting $v = 1$, we get a curve $\mathbf{S}(u,1)$ lying on the second base surface
- If we fix u, we get a circular arc of radius r, with center at the point $\mathbf{C}(u)$, lying in the plane of the two vectors $\mathbf{X}(u)$ and $\mathbf{Y}(u)$

The following picture illustrates the situation:



Transmit files can contain blends of the following types:

- Type = 'R': a rolling ball blend
- Type = 'E': a cliff edge blend

For rolling ball blends, the spine curve will be the intersection of the two surfaces obtained by offsetting the supporting surfaces by an amount given by the respective entry in range[]. Note that the offsets to be applied may be positive or negative, and that the sense of the surface is significant; i.e. the offset vector is the natural unit surface normal, times the range, times –1 if the sense is negative.

For cliff edge blends, one of the surfaces will be a blended_edge with a range of [0,0]; its spine will be the cliff edge curve, and its supporting surfaces will be the surfaces of the faces adjacent to the cliff edge. Its type will be R.

The limit fields will only be non-null if the spine curve is periodic but the edge curve being blended has terminators – for example if the spine is elliptical but the blend degenerates. In this case the two LIMIT nodes, of type 'L', determine the extent of the spine.

M.5.1.3 Calculating a Point on a Blend Surface

The computation of a point on a blend surface is very similar to the computation of a point on an intersection curve, as described in section L.4.3 above.

Suppose we have two base surfaces **A** and **B** with parameterizations $(p,q) \mapsto \mathbf{A}(p,q)$ and $(s,t) \mapsto \mathbf{B}(s,t)$, and we want to calculate a point $\mathbf{S}(u,v)$ on the blend surface.

Let $\mathbf{N}_A$ and $\mathbf{N}_B$ denote the unit normal functions of the surfaces $\mathbf{A}$ and **B** respectively:

$$\boldsymbol{N}_A = \frac{\dfrac{\partial \boldsymbol{A}}{\partial p} \times \dfrac{\partial \boldsymbol{A}}{\partial q}}{\left\| \dfrac{\partial \boldsymbol{A}}{\partial p} \times \dfrac{\partial \boldsymbol{A}}{\partial q} \right\|} \quad ; \quad \boldsymbol{N}_B = \frac{\dfrac{\partial \boldsymbol{B}}{\partial s} \times \dfrac{\partial \boldsymbol{B}}{\partial t}}{\left\| \dfrac{\partial \boldsymbol{B}}{\partial s} \times \dfrac{\partial \boldsymbol{B}}{\partial t} \right\|}$$

Then we can form the two offset surfaces:

$$\overline{\boldsymbol{A}}(p,q) = \boldsymbol{A}(p,q) + r\boldsymbol{N}_A(p,q)$$

$$\overline{\boldsymbol{B}}(s,t) = \boldsymbol{B}(s,t) + r\boldsymbol{N}_B(s,t)$$

Assume for the time being that the spine curve $\mathbf{C}$ is an icurve (this is usually the case). If we use the techniques described in section L.4.3 to calculate the point $\mathbf{C}(u)$, we will obtain surface parameter values p, q, s, t such that

$$\overline{\boldsymbol{A}}(p,q) = \overline{\boldsymbol{B}}(s,t) = \boldsymbol{C}(u)$$

Although the parameter values p, q, s, t came from a computation on offset surfaces, they can also be used to calculate points on the original surfaces **A** and **B**, too, which will define the vectors $\mathbf{X}(u)$ and $\mathbf{Y}(u)$. The equation immediately above tells us that

$$\boldsymbol{A}(p,q) + r\boldsymbol{N}_A(p,q) = \boldsymbol{C}(u)$$

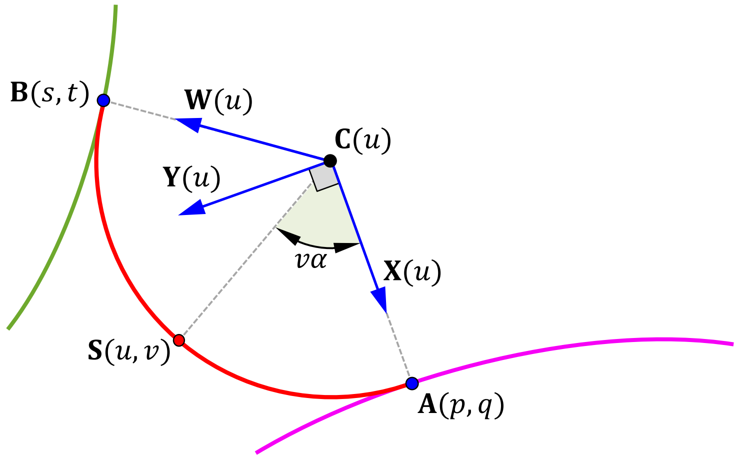
$$\boldsymbol{B}(s,t) + r\boldsymbol{N}_B(s,t) = \boldsymbol{C}(u)$$

So the point $\mathbf{A}(p,q)$ is the foot of the normal from the point $\mathbf{C}(u)$ to the surface **A**. Similarly, $\mathbf{B}(s,t)$ is the foot of the normal from the point $\mathbf{C}(u)$ to the surface **B**. The vectors $\mathbf{A}(p,q) - \mathbf{C}(u)$ and $\mathbf{B}(s,t) - \mathbf{C}(u)$ are perpendicular to the curve **C**. Then we set

$$\boldsymbol{X}(u) = \frac{\boldsymbol{A}(p,q) - \boldsymbol{C}(u)}{\|\boldsymbol{A}(p,q) - \boldsymbol{C}(u)\|} \quad ; \quad \boldsymbol{W}(u) = \frac{\boldsymbol{B}(s,t) - \boldsymbol{C}(u)}{\|\boldsymbol{B}(s,t) - \boldsymbol{C}(u)\|}$$

$$\boldsymbol{Y}(u) = \frac{\boldsymbol{X}(u) \times \big(\boldsymbol{X}(u) \times \boldsymbol{W}(u)\big)}{\big\|\boldsymbol{X}(u) \times \big(\boldsymbol{X}(u) \times \boldsymbol{W}(u)\big)\big\|}$$

so $\mathbf{X}(u)$ and $\mathbf{Y}(u)$ are orthogonal unit vectors that lie in a plane perpendicular to the curve $\mathbf{C}(u)$.



We let $\alpha(u)$ be the angle between $\mathbf{X}(u)$ and $\mathbf{W}(u)$. Then the point $\mathbf{S}(u,v)$ on the blend surface is:

$$\boldsymbol{S}(u,v) = \boldsymbol{C}(u) + r\cos\big(v\alpha(u)\big)\boldsymbol{X}(u) + r\sin\big(v\alpha(u)\big)\boldsymbol{Y}(u)$$

Alternatively, we know that $\mathbf{W}(u) = (\cos\alpha)\mathbf{X}(u) + (\sin\alpha)\mathbf{Y}(u)$. Dividing through by $\sin\alpha$, we get $\mathbf{Y}(u) = (\operatorname{cosec}\alpha)\mathbf{W}(u) - (\cot\alpha)\mathbf{X}(u)$. Substituting this into the formula above gives:

$$\boldsymbol{S}(u,v) = \boldsymbol{C}(u) + r(cos\, v\alpha - cot\,\alpha\, sin\, v\alpha)\boldsymbol{X}(u) + r(cosec\,\alpha\, sin\, v\alpha)\boldsymbol{W}(u)$$

In fact, this is the formula that is actually used in the Parasolid code.

M.5.1.4 Blend Surface Pseudocode

We assume that we already have a function that computes surface parameter values corresponding to a given parameter value on an icurve, as described in section L.4.3:

```
/// <summary>Calculates surface parameters at a location on an icurve</summary>
/// <param name="A">First  surface evaluation function, (s,t) --> A(s,t)</param>
/// <param name="B">Second surface evaluation function  (u,v) --> B(u,v)</param>
/// <param name="w">Parameter value on icurve, C</param>
/// <returns>Surface parameter values (s,t,u,v)</returns>
/// <remarks>
/// The surface parameter values (s,t,u,v) are such that
///       A(s,t) = B(u,v) = C(w)
/// </remarks>
static double[] ICurveParams(SurfaceFunction A, SurfaceFunction B, double w)
```

Also, we assume that we have functions to compute points on the offset surfaces $\bar{\mathbf{A}} = \text{offsetA}$ and $\bar{\mathbf{B}} = \text{offsetB}$. For example:

```
/// <summary>Compute point on offset surface OffsetA</summary>
/// <param name="u">Parameter value</param>
/// <param name="v">Parameter value</param>
/// <returns>Position on offset surface at parameter values (u,v)</returns>
public static Position OffsetA(double u, double v)
{
   Position P = surfaceA.Position(u,v);
   Vector N = surfaceA.UnitNormal(u,v);
   return P + d*N;
}
```

Then the code to compute a point on a blend surface is as follows:

```
/// <summary>Calculate a point on a blend surface at given parameter values
(u,v)</summary>
/// <param name="surfaceA">First surface</param>
/// <param name="surfaceB">Second surface</param>
/// <param name="r">Radius of blend surface</param>
/// <param name="u">Parameter value</param>
/// <param name="v">Parameter value</param>
/// <returns>Point S(u,v) on blend surface</returns>
public static Position Blend(Surface surfA, Surface surfB, double r, double u, double v)
{
   // Compute surface parameter values at point on blend spine,
   // which is the intersection curve of two offset surfaces
   double[] pqst = ICurveParams(OffsetA, OffsetB, u);
   double p = pqst[0];     double q = pqst[1];
   double s = pqst[2];     double t = pqst[3];

   // Compute point on icurve.
   Position Cu = OffsetA(r, p, q);

   // Compute radial vectors. Note that these are not unitized.
   Position Apq = surfA.Position(p,q);     Vector X = Apq - Cu;
   Position Bst = surfB.Position(s,t);     Vector W = Bst - Cu;

   // Angular calculations in v direction
   double denom = r*r;
   double cosA = (X*W) / denom;
   double sinA = Vector.Norm(Vector.Cross(X, W));
   double A = System.Math.Atan2(sinA, cosA);
```

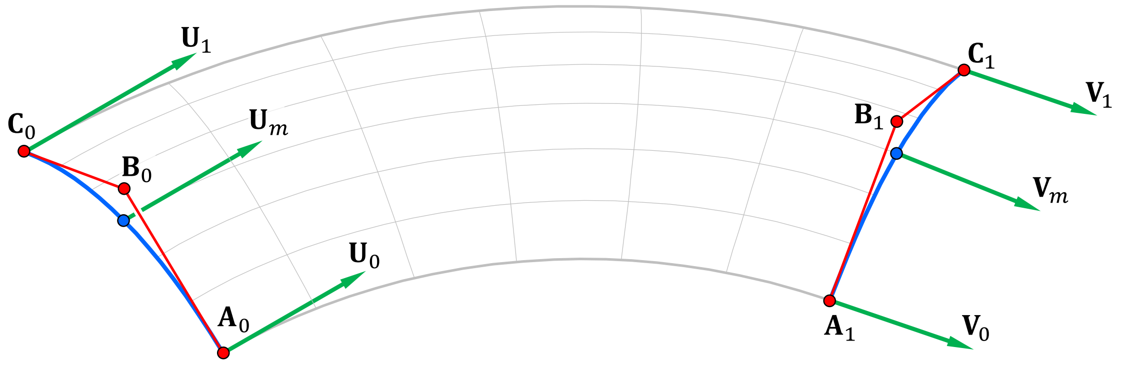
```
    double cosvA = System.Math.Cos(v*A);
    double sinvA = System.Math.Sin(v*A);
    double cotA = cosA / sinA;
    double cosecA = 1 / sinA;

    // Final point on blend surface
    Position Suv  =  Cu  +  (cosvA - cotA*sinvA)*X  +  (cosecA*sinvA)*W;
    return Suv;
}
```

M.5.2 Approximating a Blend Surface

Approximation of a blend surface can be performed using an approach that’s very similar to the spline approximation technique described in section L.4.4. The basic calculation is the construction of certain rational Bezier patches, as outlined in section L.5.2.1 below, which interpolate the circular cross-sections of the blend plus some derivatives. These Bezier patches can be assembled into a smooth NURBS surface.

M.5.2.1 The Basic Bézier Patch

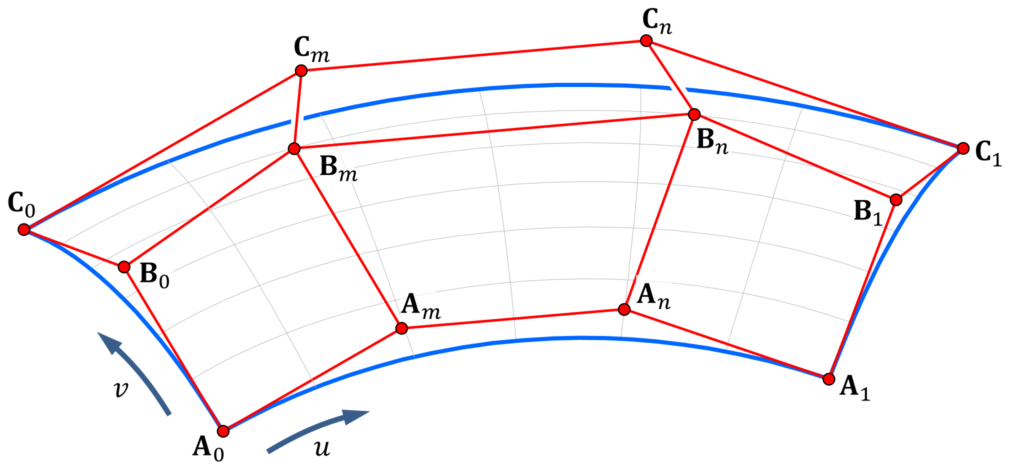
The construction starts with the following data that can easily be obtained from a blend surface:



We have:

- At one end, a circular arc, expressed as a rational quadratic Bézier curve with poles $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$. The poles $\mathbf{A}_0$ and $\mathbf{C}_0$ have weights equal to one, and the pole $\mathbf{B}_0$ has weight w_0.
- Similarly, at the other end: a circular arc, expressed as a rational quadratic Bézier curve with poles $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$. The poles $\mathbf{A}_1$ and $\mathbf{C}_1$ have weights of 1, and the pole $\mathbf{B}_1$ has weight w_1.
- Three derivative vectors $\mathbf{U}_0, \mathbf{U}_m, \mathbf{U}_1$at one end
- Three derivative vectors $\mathbf{V}_0, \mathbf{V}_m, \mathbf{V}_1$at the other end

From this data, we can construct a Bézier patch $\mathbf{S}(u, v)$, as shown here:



The patch is rational, and has degree 3 in the u-direction, and degree 2 in the v-direction. Its poles are constructed as follows:

- The poles $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$ and their weights are copied directly from the circular arc.
- The poles $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1$ and their weights are copied directly from the other circular arc.
- $\mathbf{A}_m = \mathbf{A}_0 + \frac{1}{3}\mathbf{U}_0$, and $\mathbf{C}_m = \mathbf{C}_0 + \frac{1}{3}\mathbf{U}_1$
- $\mathbf{B}_m = \mathbf{B}_0 + \frac{2}{3w_0}\left(\mathbf{U}_m - \frac{1}{4}\mathbf{U}_0 - \frac{1}{4}\mathbf{U}_1\right)$; weight = w_0
- $\mathbf{A}_n = \mathbf{A}_1 - \frac{1}{3}\mathbf{V}_0$, and $\mathbf{C}_n = \mathbf{C}_1 - \frac{1}{3}\mathbf{V}_1$
- $\mathbf{B}_n = \mathbf{B}_1 - \frac{2}{3w_1}\left(\mathbf{V}_m - \frac{1}{4}\mathbf{V}_0 - \frac{1}{4}\mathbf{V}_1\right)$; weight = w_1

With this definition, it is straightforward to verify that

- $v \mapsto \mathbf{S}(0, v)$ is a circular arc coinciding with the given one
- $v \mapsto \mathbf{S}(1, v)$ is a circular arc coinciding with the other given one
- $\mathbf{S}^u(0,0) = \mathbf{U}_0$, $\mathbf{S}^u\left(0, \frac{1}{2}\right) = \mathbf{U}_m$, and $\mathbf{S}^u(0,1) = \mathbf{U}_1$
- $\mathbf{S}^u(1,0) = \mathbf{V}_0$, $\mathbf{S}^u\left(1, \frac{1}{2}\right) = \mathbf{V}_m$, and $\mathbf{S}^u(1,1) = \mathbf{V}_1$

So, the Bézier patch interpolates the two given circular arcs and the 6 given derivative vectors.

M.5.2.2 Approximation Code

Pseudocode for approximating a blend surface is as follows:

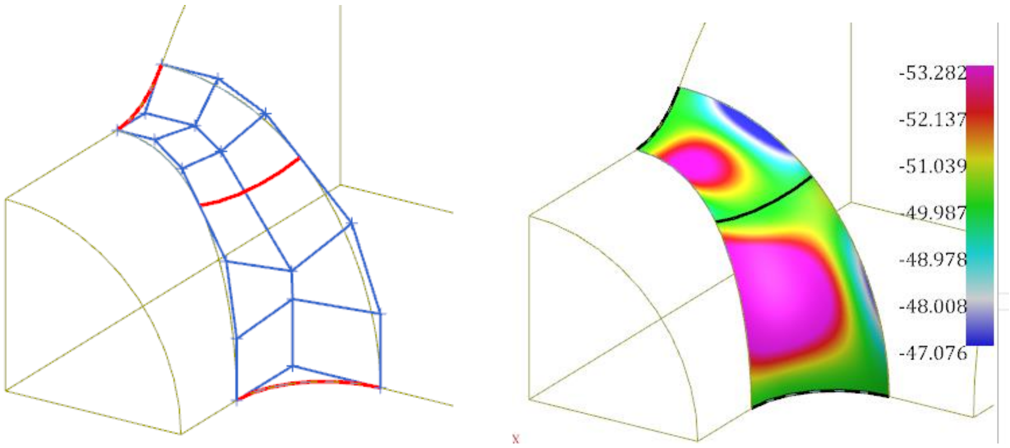
```
/// <summary>Create a b-surface approximation of a blend surface</summary>
/// <param name="blend">The blend surface to be aproximated</param>
/// <param name="nodesU">Parameter values (fractional) at which to interpolate</param>
/// <returns>The b-surface approximation</returns>
public static Bsurface ApproxBlend(BlendSurface blend, double[] nodesU)
{
   int n = nodesU.Length;

   Position[] A = new Position[2*n];      // Row of poles along edge v=0
   Position[] B = new Position[2*n];      // Middle row of poles
   Position[] C = new Position[2*n];      // Row of poles along edge v=1
   double[] wm = new double[2*n];         // Weights for middle row of poles

   double[] knotsU = new double[2*n + 4];      // Surface knots in u-direction
   double[] knotsV = { 0, 0, 0, 1, 1, 1 };     // Surface knots in v-direction

   knotsU[0] = 0;     knotsU[2*n + 2] = 1;
   knotsU[1] = 0;     knotsU[2*n + 3] = 1;

   double u0 = blend.MinU;   double u1 = blend.MaxU;
   double v0 = blend.MinV;   double v1 = blend.MaxV;
   double vm = (v0 + v1) / 2;

   double third = 1.0 / 3.0;

   double[] u = new double[n];
   for (int i = 0; i < n ; i++) u[i] = (1 - nodesU[i])*u0 + nodesU[i]*u1;

   for (int i = 0; i < n ; i++)
   {
      Spline bezierArc = blend.IsoCurveU(u[i]);    // Rational quadratic
      double w = bezierArc.Weights[1];            // Weight at its middle pole
      wm[2*i] = w;   wm[2*i + 1] = w;

      double h = (i > 0)       ?   u[i] - u[i-1]   :  0.0;
      double k = (i < n - 1)   ?   u[i+1] - u[i]   :  0.0;

      Vector U0 = blend.DerivDu(u[i], v0);    Position P0 = bezierArc.Poles[0];
      Vector Um = blend.DerivDu(u[i], vm);    Position Pm = bezierArc.Poles[1];
      Vector U1 = blend.DerivDu(u[i], v1);    Position P1 = bezierArc.Poles[2];
```

```
      Vector D = (2/w) * (Um - 0.25*U0 - 0.25*U1);

      A[2*i] = P0 - third*h*U0;    A[2*i + 1] = P0 + third*k*U0;
      B[2*i] = Pm - third*h*D;     B[2*i + 1] = Pm + third*k*D;
      C[2*i] = P1 - third*h*U1;    C[2*i + 1] = P1 + third*k*U1;

      knotsU[2*i + 2] = nodesU[i];
      knotsU[2*i + 3] = nodesU[i];;
   }

   Position[,] poles = new Position[2*n, 3];
   double[,] weights = new double[2*n, 3];

   for (int i = 0; i < 2*n; i++)
   {
        poles[i, 0] = A[i]; poles[i, 1]  = B[i];    poles[i, 2] = C[i];
      weights[i, 0] = 1;    weights[i, 1] = wm[i];  weights[i, 2] = 1;
   }

   return Bsurface(poles, weights, knotsU, knotsV);
}
```

As you can see, one of the inputs to this function is an array nodesU of parameter values at which circular cross-sections and partial derivatives are to be interpolated. The idea is that this code will be called as in section L.4.4.4: you construct an approximation, measure the error, and add new values into the nodesU array if necessary. This refinement process continues until the desired approximation error has been achieved.

M.5.2.3 Example

We use an example that's very similar to the one used in section L.4.4.4. We have two cylinders, $x^2 + y^2 = 150^2$, and $y^2 + z^2 = 200^2$, and their intersection is blended with a radius of 50. The ApproxBlend function shown above was called with nodesU = { 0, 0.6, 1 }



The image on the left shows the control points of the NURBS surface approximation that was produced, and the one on the right shows its minimum curvature values. The correct minimum curvature value (-50) is preserved near the three interpolated arcs, but not elsewhere. Since this surface has only two patches, the approximation is rather poor. Using a larger number of arcs (i.e. more entries in the nodesU array) would improve the accuracy, of course. The main approximation function should call ApproxBlend with a progressively more dense nodesU array until the desired accuracy is achieved.

For the simple surface shown above, the poles (control points) are

```
(100.00000000,   0.00000000, 250.00000000)  (70.71067812,  0.00000000, 141.42135624)  (150.00000000,   0.00000000, 200.00000000)
(100.00000000,  32.05918779, 250.00000000)  (70.71067812, 24.64162626, 141.42135624)  (150.00000000,  38.47102535, 200.00000000)
( 83.61937997,  62.80582420, 233.62398975)  (82.28854357, 40.56677690, 165.79553058)  (125.42906995,  75.36698904, 186.89919180)
( 42.38837901,  92.84163156, 208.63992245)  (38.53818429, 76.94324822, 149.78734763)  ( 63.58256851, 111.40995788, 166.91193796)
( 21.35441981, 100.00000000, 200.00000000)  (25.43275108, 89.44271910, 156.52475842)  ( 32.03162971, 120.00000000, 160.00000000)
(  0.00000000, 100.00000000, 200.00000000)  ( 0.00000000, 89.44271910, 156.52475842)  (  0.00000000, 120.00000000, 160.00000000)
```

the corresponding weights are

```
1.00000000  ;  0.70710678  ;  1.00000000
1.00000000  ;  0.70710678  ;  1.00000000
1.00000000  ;  0.83426112  ;  1.00000000
1.00000000  ;  0.83426112  ;  1.00000000
1.00000000  ;  0.89442719  ;  1.00000000
1.00000000  ;  0.89442719  ;  1.00000000
```

and the knot vectors are

```
knotsU = { 0,0,0,0,  0.6, 0.6, 1,1,1,1 }
knotsV = { 0,0,0, ,1,1,1 }
```

M.5.2.4 Derivatives of Projected Curves

The algorithm given in sections L.5.2.1 and L.5.2.2 makes use of the partial derivatives with respect to u on the blend surface, so we need a way to calculate these.

As a first step, consider the general problem of finding the derivative of a curve that is a “normal projection” of a curve **C** onto a surface **S**. More specifically, suppose we have a curve $t \mapsto \mathbf{C}(t)$ that lies above a surface $(u, v) \mapsto \mathbf{S}(u, v)$. For each given t, we can find $u = u(t)$ and $v = v(t)$ such that the vector $\mathbf{R}(t) = \mathbf{S}\big(u(t), v(t)\big) - \mathbf{C}(t)$ is normal to the surface **S**. In other words, the point $\mathbf{S}(u(t), v(t))$ is the “foot” of the normal from the point $\mathbf{C}(t)$to the surface **S**, and the curve $t \mapsto \mathbf{S}(u(t), v(t))$ is the normal projection of **C** onto **S**. The vector $\mathbf{R}(t)$ is normal to the surface **S**, so we have

$$\boldsymbol{R}(t) \cdot \boldsymbol{S}^u = 0 \quad ; \quad \boldsymbol{R}(t) \cdot \boldsymbol{S}^v = 0$$

Here, as usual, $\boldsymbol{S}^u$ and $\boldsymbol{S}^v$ denote partial derivatives of **S** with respect to u and v. Differentiating each of these equations with respect to t and rearranging, we get

$$[\boldsymbol{S}^u \cdot \boldsymbol{S}^u + \boldsymbol{R}(t) \cdot \boldsymbol{S}^{uu}] \frac{du}{dt} + [\boldsymbol{S}^u \cdot \boldsymbol{S}^v + \boldsymbol{R}(t) \cdot \boldsymbol{S}^{uv}] \frac{dv}{dt} = \frac{d\boldsymbol{C}}{dt} \cdot \boldsymbol{S}^u$$

$$[\boldsymbol{S}^u \cdot \boldsymbol{S}^v + \boldsymbol{R}(t) \cdot \boldsymbol{S}^{uv}] \frac{du}{dt} + [\boldsymbol{S}^v \cdot \boldsymbol{S}^v + \boldsymbol{R}(t) \cdot \boldsymbol{S}^{vv}] \frac{dv}{dt} = \frac{d\boldsymbol{C}}{dt} \cdot \boldsymbol{S}^v$$

This is a system of two linear equations that we can solve to get du/dt and dv/dt. Then the derivative of the projected curve $\mathbf{P}(t) = \mathbf{S}(u(t), v(t))$ is

$$\frac{d\boldsymbol{P}}{dt} = \frac{du}{dt}\boldsymbol{S}^u + \frac{dv}{dt}\boldsymbol{S}^u$$

and the derivative of the vector **R** is

$$\frac{d\boldsymbol{R}}{dt} = \frac{du}{dt}\boldsymbol{S}^u + \frac{dv}{dt}\boldsymbol{S}^u - \frac{d\boldsymbol{C}}{dt}$$

M.5.2.5 Blend Surface Derivatives

The blend surface equation we obtained at the end of section L.5.1.2 was:

$$\boldsymbol{S}(u, v) = \boldsymbol{C}(u) + r(\cos v\alpha - \cot \alpha \sin v\alpha)\boldsymbol{X}(u) + r(\operatorname{cosec} \alpha \sin v\alpha)\boldsymbol{W}(u)$$

Differentiating this with respect to u, we get

$$\begin{aligned}\frac{d\boldsymbol{S}}{du} &= \frac{d\boldsymbol{C}}{du} + \frac{d\alpha}{du}\{\sin v\alpha\,(\operatorname{cosec}^2 \alpha - v) - v \cot \alpha \cos v\alpha\} r\boldsymbol{X} \\ &\quad + \frac{d\alpha}{du}\operatorname{cosec} \alpha\,(v \cos v\alpha - \sin v\alpha \cos \alpha \operatorname{cosec} \alpha) r\boldsymbol{W} \\ &\quad + (\cos v\alpha - \sin v\alpha \cot \alpha) r\boldsymbol{X}^u + (\sin v\alpha \operatorname{cosec} \alpha) r\boldsymbol{W}^u\end{aligned}$$

We can obtain $\mathbf{X}^u$ and $\mathbf{W}^u$ using the techniques described in the previous section, so the only remaining unknown is $d\alpha/du$.

But we know that $\mathbf{X} \cdot \boldsymbol{W} = r^2 \cos\alpha$, and differentiating this with respect to u gives

$$\boldsymbol{X} \cdot \boldsymbol{W}^u + \boldsymbol{W} \cdot \boldsymbol{X}^u = -r^2 \sin\alpha \frac{d\alpha}{du}$$

and so

$$\frac{d\alpha}{du} = -\frac{\boldsymbol{X} \cdot \boldsymbol{W}^u + \boldsymbol{W} \cdot \boldsymbol{X}^u}{r^2 \sin\alpha}$$

While all of the above might be interesting mathematics, for some people, a more practical approach is to use divided difference approximations to obtain partial derivatives. The following function does this – the recommendation is to use this approach instead of the analytical one above:

```
/// <summary>Evaluates approximate first derivative on a blend surface</summary>
/// <param name="blend">The blend surface</param>
/// <param name="u">Parameter value u at which to compute</param>
/// <param name="v">Parameter value v at which to compute</param>
/// <returns>Partial derivative wrt u at parameter values (u,v)</returns>
public static Vector ApproxDerivDu(BlendSurface blend, double u, double v)
{
   Position p0, pm, p1;
   Vector deriv;

   // Step needs to be chosen carefully; not too big, and not too small
   double u0 = blend.MinU;    double u1 = blend.MaxU;
   double step = 0.000001;
   double du = (u1 - u0)*step;

   if (u < u0 + du)  // Near beginning, so use forward difference formula
   {
      p0 = blend.Position(u,v);
      pm = blend.Position(u + du, v);
      p1 = blend.Position(u + 2*du, v);
      deriv = (3 * (pm - p0) - (p1 - pm)) / (2 * du);
   }

   else if (u > u1 - du)  // Near the end, so use backward difference formula
   {
      p0 = blend.Position(u - 2*du, v);
      pm = blend.Position(u - du, v);
      p1 = blend.Position(u, v);
      deriv = (3*(p1 - pm) - (pm - p0)) / (2*du);
   }

   else  // Normal case in interior of interval [u0,u1], so use central difference
formula
   {
      p0 = blend.Position(u - du, v);
      p1 = blend.Position(u + du, v);
      deriv = (p1 - p0) / (2.0 * du);
   }

   return deriv;
}
```

There are much more sophisticated algorithms for numerical estimation of derivatives; see section 5.7 of the Numerical Recipes book for details. In particular, the discussion of the choice of the variable step is well worth reading. If step is too large, the divided difference formula has a large truncation error, and if step is too small, the calculations will be ruined by subtractive cancellation. But our only use for partial derivatives is the construction of a blend surface approximation, so we do not need to know their values very accurately, and the simple function given above is more than adequate for our purpose. In fact, for the example given in section L.5.2.3, the derivative estimates given by the ApproxDerivDu function are everywhere within 0.00000019 of the correct values, which is certainly good enough.

M.5.2.6 Approximating Blend Surface Edges

The “contact curve” edges of a blend surface are icurves formed by intersections with special “blend bound” surfaces. Blend bound surfaces are complicated and require special techniques, so it would be nice if we could avoid them; it turns out that we can. Suppose we have constructed a b-spline approximation **S** of a blend surface **B**. Then the edges of **S** can easily be constructed, and these will provide spline curve approximations of the edges of **B**.

M.6 Blend Surface Questions and Answers

This section reproduces some earlier questions and answers about blend surfaces, for easy reference.

How does the blend radius r, correspond to the data stored in the XT Data? And is the parametric representation correct – can’t r be positive for one surface and negative for another?

The blend radius *r* is the magnitude of the offset distance of each underlying surface and directly relates to the range values stored in the rolling ball blend surface data. The range value for each surface can be $+r$ or $-r$. The $+$ or $-$ indicates the direction of the offset of the underlying surface taking into account the surface sense.

When interpreting the parametric representation quoted in the XT Data Description, the current formula is correct since the blend radius r itself is always a positive value (even though the offset value in the range field may be negative):

$$\boldsymbol{R}(u,v) = \boldsymbol{C}(u) + r\,cos\big(v\alpha(u)\big)\,\boldsymbol{X}(u) + r\,sin\big(v\alpha(u)\big)\,\boldsymbol{Y}(u)$$

However, to avoid confusion you may prefer to think of the formula as follows:

$$\boldsymbol{R}(u,v) = \boldsymbol{C}(u) + |r|\,cos\big(v\alpha(u)\big)\,\boldsymbol{X}(u) + |r|\,sin\big(v\alpha(u)\big)\,\boldsymbol{Y}(u)$$

What is the spine curve of a rolling ball blend surface?

Conceptually, for a rolling ball blend surface, the spine curve describes the path of the center of the ball as it moves along the two underlying surfaces, maintaining point contact with both surfaces at any one time as it does so.

This is explained in the XT Data Description as follows:
For rolling ball blends, the spine curve will be the intersection of the two surfaces obtained by offsetting the supporting surfaces by an amount given by the respective entry in range[]. Note that the offsets to be applied may be positive or negative, and that the sense of the surface is significant; i.e. the offset vector is the natural unit surface normal, times the range, times –1 if the sense is negative.

What type of curves can be referenced as the spine of the rolling ball blend surface?

For a rolling ball blend surface that has been created using the Parasolid Kernel, the most common types of spine curves are ellipses, intersection curves, rational b-spline curves which exactly represent a portion of an ellipse.

Additionally, for cliff blends (type = E) created by the Parasolid Kernel, it is also common to see lines, b-curves and circles representing the spine of the supporting (zero radius rolling ball blend) surface.

The format does not preclude using any exact 3D curve as the spine curve within the blended_edge data structure.

How is the rolling ball blend surface parameterized?

The XT Data Description explains the parameterization as follows:

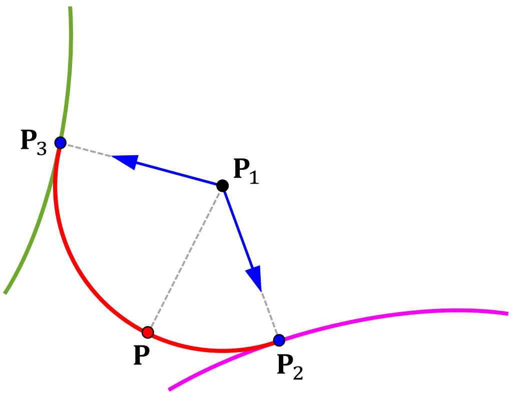
The u parameter is inherited from the spine, the constant u lines being circles perpendicular to the spine curve. The v parameter is zero [0] at the blend boundary on the first surface, and one [1] on the blend boundary on the second surface; unless the sense of the spine curve is negative, in which case it is the other way round. The v parameter is proportional to the angle around the circle.

Given an [x,y,z] point on a blend surface, how do I determine the corresponding [u,v]?

A blend surface is such that the following is always true:

- Given a point $\mathbf{P}$ on the blend surface, find the closest point $\mathbf{P}_1$ on the blend spine. The parameter of that point $\mathbf{P}_1$ on the curve, provides the u parameter of $\mathbf{P}$.
- Given the point, $\mathbf{P}_1$ on the spine curve, find the closest points $\mathbf{P}_2$ and $\mathbf{P}_3$ on each of the underlying surfaces. These points $\mathbf{P}_2$ and $\mathbf{P}_3$ will both lie on the corresponding blend boundaries.
- One of the points $\mathbf{P}_2$ and $\mathbf{P}_3$ will have v parameter [0] and the other will have v parameter [1], dependent on the sense of the spine curve.
- Points $\mathbf{P}$, $\mathbf{P}_2$ and $\mathbf{P}_3$ will all lie on the same circular arc on the underlying blend surface. The v parameter of $\mathbf{P}$ will be proportional to its angular location along the arc between $\mathbf{P}_2$ and $\mathbf{P}_3$.

The following picture illustrates further

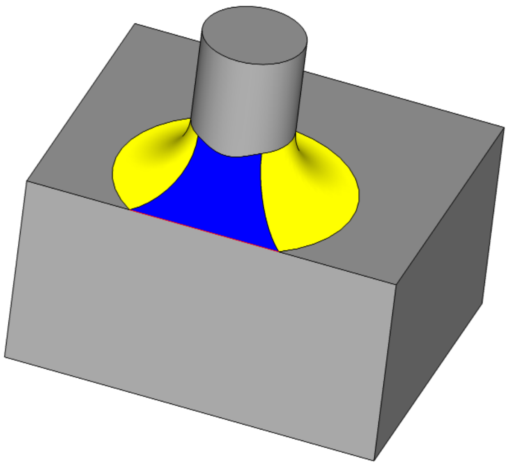


What is a ‘cliff-edge blend’? How does it relate to the blended_edge structure?

The presence of a ‘cliff-edge blend’ (referred to in the following explanation as just ‘cliff blend’) is denoted when the **blend_type** field of the blended_edge structure is set to ‘E’ (rather than ‘R’). This indicates that the blended_edge structure represents a cliff blend.

Conceptually, a cliff blend is one where a rolling ball blend surface would overflow a face boundary. In order to stay within the face boundary, the rolling ball moves along this ‘cliff’ edge whilst also maintaining contact with the other surface and such that the resultant cliff blend surface meets that other surface tangentially.

A simple example of a rolling ball blend surface (in yellow) and a cliff blend surface (in blue) are shown in the picture below. The cliff (edge/curve) is also shown (in red). Note that this is shown for representation purposes only; if such a part had been constructed using the Parasolid Kernel, the yellow surface would be simplified exactly to a torus as part of the blend construction.



For a cliff blend, the spine curve is represented in a different way to the spine of a rolling ball blend surface. This is explained in the XT Data Description as follows:

For cliff edge blends, one of the surfaces will be a blended_edge with a range of [0,0]; its spine will be the cliff edge curve, and its supporting surfaces will be the surfaces of the faces adjacent to the cliff edge. Its type will be R.

For example in the case pictured above, the elements of the structure include the following:

For the rolling ball blend surface in yellow:

- One of the elements of the **surface** array is the cylinder
- The other element of the **surface** array is the planar surface of the top face
- The **spine** is a curve representing the intersection of the offsets of the two surfaces above
- The **range** is [+/-r, +/-r] (the sign for each takes into account the surface normal and surface sense)
- The **blend_type** is R

For the cliff blend surface in blue:

- One of the elements of the surface array is the cylinder
- The other element of the surface array is a rolling ball blend surface with range = [0,0]
- Conceptually this is a representation of the cliff curve as a zero radius blend (see below for more details)
- The spine is derived in exactly the same way as for a rolling ball blend (based on the intersection of the offsets of the two surfaces)
- The range is [+/-r,+/-r]
- The blend_type is E

For the other element of the surface array above:

- One of the elements of its surface array is the front-facing planar surface adjacent to the edge

- The other element of its surface array is the top-facing planar surface adjacent to the edge in the original model
- Note that these two surfaces intersect to give the cliff curve
- The range is [0,0]
- The spine is the cliff curve (formed by intersecting the two surfaces), and can be conceptually thought of as the curve of the cliff edge
- The blend_type is R

M.7 Annex Bibliography

The following documents are referenced in this document:

[1] ISO/IS 14306:2012 – JT file format specification for 3D Visualization (2012-12-14)

[2] JT Implementor Forum – Implementation Guidelines, Version 0.9 (May 27, 2014)

[3] JT Content Harmonization (v3.0) – Progress Report and Proposal for JT and Accompanying Formats (2013-05-21)

[4] Layer Filter Support in JT Files (v2.0; Feburary, 2013) – provided by Siemens

[5] JT-v10-file-format-reference-rev-A_tcm1023-224370.pdf – JT File Format Reference Version 10.0 Rev-A – provided by Siemens

[6] JT OPEN ADVANCED MATERIALS (March, 2012) – provided by Siemens

[7] Numerical Recipes, by Press, Teukolsky, Vetterling, Flannery.

[8] http://www.gnu.org/software/gsl/

[9] http://en.wikipedia.org/wiki/MINPACK

[10] http://en.wikipedia.org/wiki/Loss_of_significance

Annex N

JT PMI Properties

The table below lists properties that can be found in JT parts that contain Generic PMI Entities .

Tables are provided for each “Generic PMI Entity type”. A full list of Generic PMI Entity Types can be found in the “Generic PMI Entities” logical collection description from the “PMI Manager Meta Data Element”.

Each table lists the property keys that are vaild for the specific Generic PMI entity Type. For example, all valid PMI properties for Dimensions can be found in the table PMI_DIM_TYPE (0x0082). In some cases, a Generic Enitity type shares a common set of Property keys. In these cases more than one Generic PMI Entity Type is listed for the table. For example, PMI_SPOT_WELD_TYPE (0x0004), PMI_ARC_SPOT_WELD_TYPE (0x0018), PMI_RES_SPOT_WELD_TYPE (0x0020), PMI_DOLLOP_TYPE (0x0028), PMI_CLINCH_TYPE (0x0040) all share the same property keys.

There are four columns in each table;

1. “Key” Property Atom Value String – the string value found in the “Key PMI Property Atom” as described in “PMI Property”. A full description of “PMI Property” can be found in the “PMI Model Views” logical collection.
2. “Value” Property Atom Value String Encoding Format - the string value found in “Value PMI Property Atom” as described in “PMI Property”. A full description of “PMI Property” can be found in the “PMI Model Views” logical collection.
3. Prequist Key – keys required for the proper use of the listed property key
4. AdditionalNote – additional information for the property key

The PMI Properties table makes use of C programming string format specifiers in order to describe the content expected by the property key as all properties are natively stored as UTF16 encoded strings. For example “%.16g” indicates an encoded double value, “%s” a string and “%d” an integer.

Note: Implementers should use the descriptions found in PMI Manager Meta Data Element as opposed to working with PMI Data Segment.

Note: The following comment applies for all properties where the value format is "0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string

- This defines an hexadecimal integer representing RGB colour where value has “0x00bbggrr” form. The low-order byte contains a value for the relative intensity of red; the second byte contains a value for the relative intensity of green; and the third byte contains a value for the relative intensity of blue. The high-order byte shall be zero. The maximum value for a single byte is 0xFF (i.e. intensity value is in the range [0:255]).

N.1 PMI_DIM_TYPE (0x0082)

Table 205 — PMI_DIM_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DimensionText[%d]	NULL		DimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
DimensionText[%d].Text	NULL	DimensionText[%d]	
DimensionText[%d].Text.Item[%d]	NULL	DimensionText[%d].Text	
DimensionText[%d].Text.Item[%d].Outline	NULL	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].Outline.Side	NULL	DimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.colour replaces DimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.filled	"%d"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.style replaces DimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.type replaces DimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineType for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.visible	"%d"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.width.name replaces DimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineName for compatiblity.
DimensionText[%d].Text.Item[%d].bold	"%d"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].font	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].italic	"%d"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].language	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].string	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textAspect	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textHeight	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].textLineWidth.name	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Outline	NULL	DimensionText[%d].Text	
DimensionText[%d].Text.Outline.Side	NULL	DimensionText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Outline.Side.width.name	"%s"	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.colour replaces DimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineColour for compatiblity.
DimensionText[%d].Text.Outline.doubleOffset	"%.16g"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.filled	"%d"	DimensionText[%d].Text.Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.style replaces DimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineStyle for compatiblity.
DimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.type replaces DimensionText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineType for compatiblity.
DimensionText[%d].Text.Outline.visible	"%d"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Outline.width.name	"%s"	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.width.name replaces DimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineName for compatiblity.
DimensionText[%d].Text.bold	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.font	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.italic	"%d"	DimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.italicAngle	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DimensionText[%d].Text	
DimensionText[%d].Text.language	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.lineFactor	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.maxExtend	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.name	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.spaceFactor	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DimensionText[%d].Text	
DimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DimensionText[%d].Text	
DimensionText[%d].Text.textAspect	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text	
DimensionText[%d].Text.textHeight	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	DimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.textLineWidth.name	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	DimensionText[%d].Text	
DimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text	
DimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DimensionText[%d].Text	
DimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	DimensionText[%d]	
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
DualDimensionText[%d]	NULL		DualDimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
DualDimensionText[%d].Text	NULL	DualDimensionText[%d]	
DualDimensionText[%d].Text.Item[%d]	NULL	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.Item[%d].Outline	NULL	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].Outline.Side	NULL	DualDimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.colour replaces DualDimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.filled	"%d"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.style replaces DualDimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.type replaces DualDimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineType for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.visible	"%d"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.width.name replaces DualDimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineName for compatiblity.
DualDimensionText[%d].Text.Item[%d].bold	"%d"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].font	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].italic	"%d"	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].language	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].string	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textAspect	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textHeight	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].textLineWidth.name	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Outline	NULL	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.Outline.Side	NULL	DualDimensionText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Outline.Side.width.name	"%s"	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.colour replaces DualDimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineColour for compatiblity.
DualDimensionText[%d].Text.Outline.doubleOffset	"%.16g"	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.filled	"%d"	DualDimensionText[%d].Text.Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.style replaces DualDimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineStyle for compatiblity.
DualDimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.type replaces DualDimensionText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineType for compatiblity.
DualDimensionText[%d].Text.Outline.visible	"%d"	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Outline.width.name	"%s"	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.width.name replaces DualDimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineName for compatiblity.
DualDimensionText[%d].Text.bold	"%d"	DualDimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.font	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.italic	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.italicAngle	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.language	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.lineFactor	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.maxExtend	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.name	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.spaceFactor	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textAspect	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textHeight	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	DualDimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.textLineWidth.name	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	DualDimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DualDimensionText[%d].Text	
DualDimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	DualDimensionText[%d]	
FeatureIdentification	"%s"		
Format	NULL		*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d]	NULL	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline	NULL	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.Side	NULL	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.Side.width.name	"%s"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.colour replaces *Format*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineColour for compatiblity.
Format.Item[%d].Outline.doubleOffset	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.filled	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Format.Item[%d].Outline.style replaces *Format*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineStyle for compatiblity.
Format.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.type replaces *Format*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineType for compatiblity.
Format.Item[%d].Outline.visible	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].Outline.width.name	"%s"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.width.name replaces *Format*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineName for compatiblity.
Format.Item[%d].bold	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].font	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italic	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italicAngle	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].language	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].spaceFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].string	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textAspect	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textHeight	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].textLineWidth.name	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.bold	"%d"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.font	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italic	"%d"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italicAngle	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.language	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.lineFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.spaceFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textAspect	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textHeight	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textLineWidth.*measurementType*	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.textLineWidth.name	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.underline	String representing enumeration such that: "0"=over, "1"=under	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
FreeState	NULL		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
GeneralToleranceISO	NULL		
GeneralToleranceISO.toleranceClass	String representing enumeration such that: "0"=fine, "1"=medium, "2"=coarse, "3"=verycoarse,	GeneralToleranceISO	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
ReferenceText[%d]	NULL		ReferenceText[%d] index expected to range from 0-1 inclusive.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Each instance of this is expected to have a different position such that ReferenceText[x].position != ReferenceText[y].position for any combination of x and y
ReferenceText[%d].Text	NULL	ReferenceText[%d]	
ReferenceText[%d].Text.Item[%d]	NULL	ReferenceText[%d].Text	
ReferenceText[%d].Text.Item[%d].Outline	NULL	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].Outline.Side	NULL	ReferenceText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.colour replaces ReferenceText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineColour for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.filled	"%d"	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.style replaces ReferenceText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineStyle for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.type replaces ReferenceText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineType for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline.visible	"%d"	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].Outline.width.name	"%s"	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.width.name replaces ReferenceText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineName for compatiblity.
ReferenceText[%d].Text.Item[%d].bold	"%d"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].font	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].italic	"%d"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].italicAngle	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].language	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineFactor	"%.16g"	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].spaceFactor	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].string	"%s"	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textAspect	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textHeight	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].textLineWidth.name	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Outline	NULL	ReferenceText[%d].Text	
ReferenceText[%d].Text.Outline.Side	NULL	ReferenceText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Outline.Side.width.name	"%s"	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.colour replaces ReferenceText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineColour for compatiblity.
ReferenceText[%d].Text.Outline.doubleOffset	"%.16g"	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.filled	"%d"	ReferenceText[%d].Text.Outline	

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.style replaces ReferenceText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineStyle for compatiblity.
ReferenceText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.type replaces ReferenceText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineType for compatiblity.
ReferenceText[%d].Text.Outline.visible	"%d"	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Outline.width.name	"%s"	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.width.name replaces ReferenceText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineName for compatiblity.
ReferenceText[%d].Text.bold	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.font	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.italic	"%d"	ReferenceText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.italicAngle	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ReferenceText[%d].Text	
ReferenceText[%d].Text.language	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.lineFactor	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.maxExtend	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.name	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.spaceFactor	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ReferenceText[%d].Text	
ReferenceText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	ReferenceText[%d].Text	
ReferenceText[%d].Text.textAspect	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text	
ReferenceText[%d].Text.textHeight	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textLineWidth.*measurementType*	"%.16g"	ReferenceText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.textLineWidth.name	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ReferenceText[%d].Text	
ReferenceText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text	
ReferenceText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	ReferenceText[%d].Text	
ReferenceText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ReferenceText[%d]	Only the before and after values are expected.
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Statistical	NULL		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text	NULL		*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.bold	"%d"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference..
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference..
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
alignment	String representing enumeration such that: "0"=middle, "1"=bottom		
allAround	"%d"		
allOver	"%d"		
angleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		
angleNumerator	"%.16g"		
appendedTextSpaceFactor	"%.16g"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
basic	"%d"		
blanked	"%d"		
bold	"%d"		
causality	String representing enumeration such that: "0"=key, "1"=functional, "2"=reference, "3"=associated, "4"=pmi		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
commaAsDecimal	"%d"		
deviation	"%s"		
diameterPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dimensionLeadingZero	"%d"		
dimensionLineBetweenArrows	"%d"		
dimensionLineTrim	"%d"		
dimensionTrailingZero	"%d"		
direction	"%.16g %.16g %.16g"		
documentation	"%d"		
dualDimensionLineCenter	"%d"		
dualLowerDeltaDenominator	"%d"		
dualPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dualPrecision	"%d"		
dualTolerancePrecision	"%d"		
dualType	String representing enumeration such that: "0"=positional, "1"=bracket, "2"=lineseparated		
dualUnit	"%s"		
dualUnitText	"%d"		
dualUpperDeltaDenominator	"%d"		
dualValueDenominator	"%d"		
featureOfSize	"%d"		
fitGrade	"%d"		
flag	"%d"		
foldLocation	"%.16g %.16g %.16g"		
font	"%s"		
fraction	"%d"		
fractionSize	String representing enumeration such that: "0"=full, "1"=twothirds, "2"=threequarters		
grade	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
group	"%s"		
inspection	"%d"		
inspectionDisplay	String representing enumeration such that: "0"=unset, "1"=none, "2"=before, "3"=after, "4"=beforeafter, "5"=all		
isReference	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
limitDisplay	String representing enumeration such that: "0"=none, "1"=limitfit		
limitFitOrder	String representing enumeration such that: "0"=valuelimitfittolerance, "1"=tolerancevaluelimitfit, "2"=valuetolerancelimitfit		
limitFitParenthesis	String representing enumeration such that: "0"=none, "1"=valuelimitfit, "2"=limitfit, "3"=tolerance, "4"=value, "5"=valuetolerance		
lineFactor	"%.16g"		
linearPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
lowerDelta	"%.16g"		
lowerDeltaDenominator	"%d"		
majorAngle	"%d"		
manual	"%d"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
narrowLeaderAngle	"%.16g"		
narrowOffset	"%.16g"		
notToScale	"%d"		
ordinateBaselineZero	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
origin	String representing enumeration such that: "0"=first, "1"=second		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
patternCount	"%d"		
pitchDiaDeviation	"%s"		
pitchDiaGrade	"%d"		
precision	"%d"		
projected	"%d"		
radialPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
referenceContent	String representing enumeration such that: "0"=value, "1"=prefix, "2"=tolerance		
referenceDisplay	String representing enumeration such that: "0"=parenthesis, "1"=reference, "2"=matched		
singleSideFirst	"%d"		
singleSideLength	"%.16g"		
singleSided	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
statisticalPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
style	String representing enumeration such that: "0"=lineardiametral, "1"=radial, "2"=controlledradial, "3"=diametral, "4"=limits, "5"=ordinate, "6"=sphericalradial, "7"=sphericaldiametral, "8"=narrow, "9"=none		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLeaderPosition	String representing enumeration such that: "0"=none, "1"=aboveleader, "2"=afterleader, "3"=abovestub, "4"=afterstub		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
threadClass	"%d"		
toleranceAngleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		
toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral		
toleranceLeadingZero	"%d"		
tolerancePrecision	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
toleranceTextSpaceFactor	"%.16g"		
toleranceTrailingZero	"%d"		
type	String representing enumeration such that: "0"=curvelength, "1"=linear, "2"=angular, "3"=radial		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
unitText	"%d"		
upperDelta	"%.16g"		
upperDeltaDenominator	"%d"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
value	"%.16g"		
valueDenominator	"%d"		
zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit		
zeroToleranceSign	String representing enumeration such that: "0"=none, "1"=plus, "2"=minus		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.2 PMI_FASTENER_TYPE (0x0230)

Table 206 — PMI_FASTENER_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red	*Text*.Item[%d].Outline	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	for the colour are encoded in the string		one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel,	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline	NULL	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of GeneralText, CoordinatedEntity[%d] or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
TextFormat[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side .width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.colour replaces TextFormat[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o utlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.dou bleOffset	"%.16g"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fillC olour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fille d	"%d"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.style replaces TextFormat[%d].Item[%d].o utlineStyle due to the increasing complexity of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
axisDirection	"%.16g %.16g %.16g"		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
counterbored	"%d"		
depth[%d]	"%.16g"		

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
diameter[%d]	"%.16g"		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		
fastenerSubType	"%s"		
flag	"%d"		
font	"%s"		
group	"%s"		
indexDirection	"%.16g %.16g %.16g"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
optDirection	"%.16g %.16g %.16g"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
position	"%.16g %.16g %.16g"		
precision	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
symbolType	"%s"		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
threaded	"%d"		
type	"%s"		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.3 PMI_SURF_FINISH_TYPE (0x0041)

Table 207 — PMI_SURF_FINISH_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=startangle, "2"=endangle		
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick,	Outline.Side	*Side* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=thin		one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick,	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=thin		
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange,	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus,	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of ProcessText, LayText

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
allAroundLeader	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
bold	"%d"		
bottomExtensionLine	"%d"		
commaAsDecimal	"%d"		
direction	"%.16g %.16g %.16g"		
flag	"%d"		
font	"%s"		
group	"%s"		
invertSymbol	"%d"		
invertText	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
jisSurfaceTexture	"%d"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lay	String representing enumeration such that: "0"=parallel, "1"=multidirectional, "2"=perpendicular, "3"=circular, "4"=particulate, "5"=angularboth, "6"=radial, "7"=userdefined		
lineFactor	"%.16g"		
lowerDelta	"%.16g"		
machiningAllowance	"%s"		
maxRoughness	"%s"		
maxRoughness2	"%s"		
minRoughness	"%s"		
minRoughness2	"%s"		
modifier	"%d"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
parenthesis	String representing enumeration such that: "0"=none, "1"=left, "2"=right, "3"=both		
precision	"%d"		
roughness	"%s"		
roughness2	"%s"		
roughness3	"%s"		
roughnessCutoff	"%s"		
roughnessSpacing	"%s"		
singleLine	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=jis, "1"=ansi_y1436_1993, "2"=iso, "3"=asme_y1436m_1996, "4"=din, "5"=gb, "6"=iso_1302_2002, "7"=eskd, "8"=din_en_iso_1302_2002		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral		
toleranceLeadingZero	"%d"		
toleranceTrailingZero	"%d"		
type	String representing enumeration such that: "0"=basic, "1"=mrr, "2"=mrp		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
upperDelta	"%.16g"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
wavinessHeight	"%s"		
wavinessSpacing	"%s"		
zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.4 PMI_FCF_TYPE (0x0081)

Table 208 — PMI_FCF_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FCFText[%d]	NULL		FCFText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that FCFText[x].position != FCFText[y].position for any

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			combination of x and y
FCFText[%d].Text	NULL	FCFText[%d]	
FCFText[%d].Text.Item[%d]	NULL	FCFText[%d].Text	
FCFText[%d].Text.Item[%d].Outline	NULL	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].Outline.Side	NULL	FCFText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCFText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Item[%d].Outline	FCFText[%d].Text.Item[%d].Outline.colour replaces FCFText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.Item[%d].outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCFText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	FCFText[%d].Text.Item[%d].Outline	
FCFText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Item[%d].Outline	
FCFText[%d].Text.Item[%d].Outline.filled	"%d"	FCFText[%d].Text.Item[%d].Outline	
FCFText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCFText[%d].Text.Item[%d].Outline	FCFText[%d].Text.Item[%d].Outline.style replaces FCFText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.Item[%d].outlineStyle for compatiblity.
FCFText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCFText[%d].Text.Item[%d].Outline	
FCFText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCFText[%d].Text.Item[%d].Outline	FCFText[%d].Text.Item[%d].Outline.type replaces FCFText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.Item[%d].outlineType for compatiblity.
FCFText[%d].Text.Item[%d].Outline.visible	"%d"	FCFText[%d].Text.Item[%d].Outline	
FCFText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	FCFText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCFText[%d].Text.Item[%d].Outline.width.na	"%s"	FCFText[%d].Text.Item[%d].Outline	FCFText[%d].Text.Item[%d].Outline.wid

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
me			th.name replaces FCFText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.Item[%d].outlineName for compatiblity.
FCFText[%d].Text.Item[%d].bold	"%d"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].font	"%s"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].italic	"%d"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].italicAngle	"%.16g"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].language	"%s"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].lineFactor	"%.16g"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld,	FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
FCFText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel,	FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCFText[%d].Text.Item[%d].spaceFactor	"%.16g"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].string	"%s"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface,	FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCFText[%d].Text.Item[%d].textAspect	"%.16g"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].textHeight	"%.16g"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCFText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCFText[%d].Text.Item[%d].textLineWidth.name	"%s"	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCFText[%d].Text.Item[%d]	
FCFText[%d].Text.Outline	NULL	FCFText[%d].Text	
FCFText[%d].Text.Outline.Side	NULL	FCFText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCFText[%d].Text.Outline.Side.width.name	"%s"	FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCFText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Outline	FCFText[%d].Text.Outline.colour replaces FCFText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCFText[%d].Text.outlineColour for compatiblity.
FCFText[%d].Text.Outline.doubleOffset	"%.16g"	FCFText[%d].Text.Outline	
FCFText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text.Outline	
FCFText[%d].Text.Outline.filled	"%d"	FCFText[%d].Text.Outline	
FCFText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCFText[%d].Text.Outline	FCFText[%d].Text.Outline.style replaces FCFText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.outlineStyle for compatiblity.
FCFText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCFText[%d].Text.Outline	
FCFText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCFText[%d].Text.Outline	FCFText[%d].Text.Outline.type replaces FCFText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.outlineType for compatiblity.
FCFText[%d].Text.Outline.visible	"%d"	FCFText[%d].Text.Outline	
FCFText[%d].Text.Outline.width.*measurementType*	"%.16g"	FCFText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCFText[%d].Text.Outline.width.name	"%s"	FCFText[%d].Text.Outline	FCFText[%d].Text.Outline.width.name

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaces FCFText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCFText[%d].Text.outlineName for compatiblity.
FCFText[%d].Text.bold	"%d"	FCFText[%d].Text	
FCFText[%d].Text.font	"%s"	FCFText[%d].Text	
FCFText[%d].Text.italic	"%d"	FCFText[%d].Text	
FCFText[%d].Text.italicAngle	"%.16g"	FCFText[%d].Text	
FCFText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCFText[%d].Text	
FCFText[%d].Text.language	"%s"	FCFText[%d].Text	
FCFText[%d].Text.lineFactor	"%.16g"	FCFText[%d].Text	
FCFText[%d].Text.maxExtend	"%d"	FCFText[%d].Text	
FCFText[%d].Text.name	"%s"	FCFText[%d].Text	
FCFText[%d].Text.spaceFactor	"%.16g"	FCFText[%d].Text	
FCFText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCFText[%d].Text	
FCFText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCFText[%d].Text	
FCFText[%d].Text.textAspect	"%.16g"	FCFText[%d].Text	
FCFText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCFText[%d].Text	
FCFText[%d].Text.textHeight	"%.16g"	FCFText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCFText[%d].Text.textLineWidth.*measurementType*	"%.16g"	FCFText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
FCFText[%d].Text.textLineWidth.name	"%s"	FCFText[%d].Text	
FCFText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCFText[%d].Text	
FCFText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCFText[%d].Text	
FCFText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	FCFText[%d].Text	
FCFText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	FCFText[%d]	
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s ; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Group	NULL		*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Primary, Secondary or Tertiary.
Group.Datum[%d]	NULL	*Group*	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].FreeState	NULL	*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel	NULL	*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d]	NULL	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline	NULL	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline.Side	NULL	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Group.Datum[%d].TextLabel.Item[%d].Outline.Side.width.name	"%s"	*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Item[%d].Outline.colour replaces *Group*.Datum[%d].TextLabel.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.Item[%d].outlineColour for compatiblity.
Group.Datum[%d].TextLabel.Item[%d].Outline.doubleOffset	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline.filled	"%d"	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed,	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
			Group.Datum[%d].TextLabel.Item[%d].Outline.style replaces *Group*.Datum[%d].TextLabel.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.Item[%d].outlineStyle for compatiblity.
Group.Datum[%d].TextLabel.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Item[%d].Outline.type replaces *Group*.Datum[%d].TextLabel.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of *Group*.Datum[%d].TextLabel.Item[%d]. outlineType for compatiblity.
Group.Datum[%d].TextLabel.Item[%d].Outline.visible	"%d"	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].Outline.width.*measurementType*	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.Datum[%d].TextLabel.Item[%d].Outline.width.name	"%s"	*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Item[%d]. Outline.width.name replaces *Group*.Datum[%d].TextLabel.Item[%d]. outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.Item[%d]. outlineName for compatiblity.
Group.Datum[%d].TextLabel.Item[%d].bold	"%d"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].font	"%s"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].italic	"%d"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.Item[%d].italicAngle	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].language	"%s"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].lineFactor	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove,	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Group.Datum[%d].TextLabel.Item[%d].spaceF	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%	*Group* must be replaced by one of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
actor		d]	Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].string	"%s"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius,	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Group.Datum[%d].TextLabel.Item[%d].textAspect	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].textHeight	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.Datum[%d].TextLabel.Item[%d].textLineWidth.name	"%s"	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline	NULL	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline.Side	NULL	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.Datum[%d].TextLabel.Outline.Side.width.*measurementType*	"%.16g"	*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Group.Datum[%d].TextLabel.Outline.Side.width.name	"%s"	*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Outline.colour replaces *Group*.Datum[%d].TextLabel.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.outlineColour for compatiblity.
Group.Datum[%d].TextLabel.Outline.doubleOffset	"%.16g"	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline.filled	"%d"	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Outline.style replaces *Group*.Datum[%d].TextLabel.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Outline.type replaces *Group*.Datum[%d].TextLabel.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.outlineType for compatiblity.
Group.Datum[%d].TextLabel.Outline.visible	"%d"	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.Outline.width.*measurementType*	"%.16g"	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.Datum[%d].TextLabel.Outline.width.name	"%s"	*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.Datum[%d].TextLabel.Outline.width.name replaces

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Group.Datum[%d].TextLabel.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.Datum[%d].TextLabel.outlineName for compatiblity.
Group.Datum[%d].TextLabel.bold	"%d"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.font	"%s"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.italic	"%d"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.italicAngle	"%.16g"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.language	"%s"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.lineFactor	"%.16g"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.maxExtend	"%d"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.name	"%s"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.spaceFactor	"%.16g"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.textAspect	"%.16g"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.textHeight	"%.16g"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.textLineWidth.*measurementType*	"%.16g"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.Datum[%d].TextLabel.textLineWidth.name	"%s"	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].TextLabel.underline	String representing enumeration such that: "0"=over, "1"=under	*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].datumRef	"%s"	*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].label	"%s"	*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.Datum[%d].projected	"%d"	*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText	NULL	*Group*	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.Item[%d]	NULL	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline	NULL	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline.Side	NULL	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Group.ExtendedText.Item[%d].Outline.Side.wi	"%s"	*Group*.ExtendedText.Item[%d].Outli	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
dth.name		ne.Side	Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.ExtendedText.Item[%d].Outline.colour replaces *Group*.ExtendedText.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.Item[%d].outlineColour for compatiblity.
Group.ExtendedText.Item[%d].Outline.doubleOffset	"%.16g"	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline.filled	"%d"	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.ExtendedText.Item[%d].Outline.style replaces *Group*.ExtendedText.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of *Group*.ExtendedText.Item[%d].outlineStyle for compatiblity.
Group.ExtendedText.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.ExtendedText.Item[%d].Outline.type replaces *Group*.ExtendedText.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.Item[%d].outlineType for compatiblity.
Group.ExtendedText.Item[%d].Outline.visible	"%d"	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].Outline.width.*measurementType*	"%.16g"	*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.ExtendedText.Item[%d].Outline.width.	"%s"	*Group*.ExtendedText.Item[%d].Outli	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
name		ne	Primary, Secondary or Tertiary.
			Group.ExtendedText.Item[%d].Outline. width.name replaces *Group*.ExtendedText.Item[%d].outline Name due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.Item[%d].outline Name for compatiblity.
Group.ExtendedText.Item[%d].bold	"%d"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].font	"%s"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].italic	"%d"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].italicAngle	"%.16g"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].language	"%s"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].lineFactor	"%.16g"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface,	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Group.ExtendedText.Item[%d].spaceFactor	"%.16g"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].string	"%s"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope,	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"62"=orientationconstraint, "63"=datumtranslation		
Group.ExtendedText.Item[%d].textAspect	"%.16g"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].textHeight	"%.16g"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].textLineWidth. *measurementType*	"%.16g"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.ExtendedText.Item[%d].textLineWidth. name	"%s"	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline	NULL	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline.Side	NULL	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Outline.Side.style	String representing enumeration such	*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Outline.Side.width.*measurementType*	"%.16g"	*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Group.ExtendedText.Outline.Side.width.name	"%s"	*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
Group.ExtendedText.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.ExtendedText.Outline.colour replaces *Group*.ExtendedText.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.Outline.doubleOffset	"%.16g"	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline.filled	"%d"	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.ExtendedText.Outline.style replaces *Group*.ExtendedText.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.outlineStyle for compatiblity.
Group.ExtendedText.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart,	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
			Group.ExtendedText.Outline.type replaces *Group*.ExtendedText.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.outlineType for compatiblity.
Group.ExtendedText.Outline.visible	"%d"	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.Outline.width.*measurementType*	"%.16g"	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
Group.ExtendedText.Outline.width.name	"%s"	*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Group.ExtendedText.Outline.width.name replaces *Group*.ExtendedText.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Group*.ExtendedText.outlineName for compatiblity.
Group.ExtendedText.bold	"%d"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.font	"%s"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.italic	"%d"	*Group*.ExtendedText	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Primary, Secondary or Tertiary.
Group.ExtendedText.italicAngle	"%.16g"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.language	"%s"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.lineFactor	"%.16g"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.maxExtend	"%d"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.name	"%s"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.spaceFactor	"%.16g"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.textAspect	"%.16g"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.textHeight	"%.16g"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.textLineWidth.*measurementType*	"%.16g"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
Group.ExtendedText.textLineWidth.name	"%s"	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Group.ExtendedText.underline	String representing enumeration such that: "0"=over, "1"=under	*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	values of blue, green and red for the colour are encoded in the string		
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such	*Text*.Item[%d]	*Text* must be replaced by one of Text

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint,	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency,	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth,	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of Text

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.	"%s"	TextFormat[%d].Item[%d].Outline.Si	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
name		de	Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright,	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such	TextFormat[%d]	

“Key” Property Atom Value String	**“Value” Property Atom Value String Encoding Format**	**Prequist Key**	**AdditionalNote**
	that: "0"=sub, "1"=super		
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ToleranceCompartment[%d]	NULL		
ToleranceCompartment[%d].*DatumGroup*.ExtendedText	NULL	ToleranceCompartment[%d].*DatumGroup*	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	NULL	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	NULL	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	NULL	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed,	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.colour replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineColour for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Ext	"%d"	ToleranceCompartment[%d].*Datum*	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
endedText.Item[%d].Outline.filled		*Group*.ExtendedText.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.style replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineStyle for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.type replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineType for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.visible	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Ext	"%.16g"	ToleranceCompartment[%d].*Datum*	*measurementType* must be replaced

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
endedText.Item[%d].Outline.width.*measurementType*		*Group*.ExtendedText.Item[%d].Outline	with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.width.name replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineName for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].bold	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].font	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].italic	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].italicAngle	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].language	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].lineFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu,	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].spaceFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].string	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch,	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textAspect	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textHeight	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textLineWidth.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	NULL	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	NULL	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed,	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.colour replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineColour for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.filled	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
ToleranceCompartment[%d].*DatumGroup*.Ext	String representing enumeration such	ToleranceCompartment[%d].*Datum*	ToleranceCompartment[%d].*DatumGr*

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
endedText.Outline.style	that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.ExtendedText.Outline	*oup*.ExtendedText.Outline.style replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineStyle for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.type replaces ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineType for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.visible	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.width.name replaces ToleranceCompartment[%d].*DatumGr*

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			oup.ExtendedText.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineName for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.bold	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.font	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.italic	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.italicAngle	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.language	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.lineFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.maxExtend	"%d"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.spaceFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.Ext	String representing enumeration such	ToleranceCompartment[%d].*Datum*	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
endedText.strikethrough	that: "0"=none, "1"=single, "2"=double	*Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textAspect	"%.16g"	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textHeight	"%.16g"	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Datum Group*.ExtendedText	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textLineWidth.name	"%s"	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.ExtendedText.underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Datum Group*.ExtendedText	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	NULL	ToleranceCompartment[%d].*Datum Group*	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].FreeState	"%d"	ToleranceCompartment[%d].*Datum Group*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	NULL	ToleranceCompartment[%d].*Datum Group*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	NULL	ToleranceCompartment[%d].*Datum Group*.Reference[%d].TextLabel	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	NULL	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	NULL	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.colour replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineColour due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineColour for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.filled	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.style replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineStyle for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Ref	String representing enumeration such	ToleranceCompartment[%d].*Datum*	ToleranceCompartment[%d].*DatumGr*

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
erence[%d].TextLabel.Item[%d].Outline.type	that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Group*.Reference[%d].TextLabel.Item[%d].Outline	*oup*.Reference[%d].TextLabel.Item[%d].Outline.type replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineType for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.visible	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.width.name replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineName for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].bold	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Ite	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		m[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].font	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].italic	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].italicAngle	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].language	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].lineFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud,	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip,	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].spaceFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].string	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi,	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"63"=datumtranslation		
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textAspect	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textHeight	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textLineWidth.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	NULL	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	NULL	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Ref	String representing enumeration such	ToleranceCompartment[%d].*Datum*	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
erence[%d].TextLabel.Outline.Side.style	that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Group*.Reference[%d].TextLabel.Outline.Side	Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.colour replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineColour for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Out	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string	line	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.filled	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.style replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineStyle for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.type replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineType for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.visible	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Out	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		line	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.width.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.width.name replaces ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineName for compatiblity.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.bold	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.font	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.italic	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.italicAngle	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.language	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Ref	"%.16g"	ToleranceCompartment[%d].*Datum*	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
erence[%d].TextLabel.lineFactor		*Group*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.maxExtend	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.spaceFactor	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textAspect	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textHeight	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	*measurementType* must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textLineWidth.name	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].datumRef	"%d %d %d %s" such that first in is the dst id, second the dst type, third the	ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.		
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].label	"%s"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].*DatumGroup*.Reference[%d].projected	"%d"	ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
ToleranceCompartment[%d].FreeState	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].*Indicator*	NULL	ToleranceCompartment[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup	NULL	ToleranceCompartment[%d].*Indicator*	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].FreeState	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.Datum	NULL	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.Item[%d].Outline.Side		*or*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.colour replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineColour for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.filled	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.style replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineStyle for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.type replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Ite

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			m[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineType for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.visible	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.width.name replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineName for compatiblity.
ToleranceCompartment[%d].*Indicator*.Datum	"%d"	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].TextLabel.Item[%d].bold		*or*.DatumGroup.Datum[%d].TextLabel.Item[%d]	BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].font	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].italic	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].italicAngle	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].language	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].lineFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip,	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev,	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].spaceFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].string	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope,	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"62"=orientationconstraint, "63"=datumtranslation		
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textAspect	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textHeight	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textLineWidth.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.colour replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineColour for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.filled	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Ou

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			tline.style replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineStyle for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.type replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineType due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineType for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.visible	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.width.name replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineName for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.bold	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.font	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.italic	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.italicAngle	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.language	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.lineFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.maxExtend	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.spaceFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLab	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		el	
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textAspect	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textHeight	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textLineWidth.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.Datum	"%s"	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.Datum[%d].datumRef		*or*.DatumGroup.Datum[%d]	BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].label	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].projected	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.colour replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineColour for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.filled	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.style replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineStyle for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.Datum	String representing enumeration such	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.Item[%d].Outline.type	that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*or*.DatumGroup.ExtendedText.Item[%d].Outline	BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*. DatumGroup.ExtendedText.Item[%d]. Outline.type replaces ToleranceCompartment[%d].*Indicator*. DatumGroup.ExtendedText.Item[%d].o utlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*. DatumGroup.ExtendedText.Item[%d].o utlineType for compatiblity.
ToleranceCompartment[%d].*Indicator*.Datum Group.ExtendedText.Item[%d].Outline.visible	"%d"	ToleranceCompartment[%d].*Indicat or*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.Datum Group.ExtendedText.Item[%d].Outline.width. *measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicat or*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.Datum	"%s"	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.Item[%d].Outline.width.name		*or*.DatumGroup.ExtendedText.Item[%d].Outline	BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.width.name replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineName for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].bold	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].font	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].italic	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].italicAngle	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].language	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].lineFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove,	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ToleranceCompartment[%d].*Indicator*.Datum	"%.16g"	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.Item[%d].spaceFactor		*or*.DatumGroup.ExtendedText.Item[%d]	BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].string	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs,	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textAspect	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textHeight	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textLineWidth.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	NULL	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.colour replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineColour for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.filled	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.style replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineStyle for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright,	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.type replaces ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineType for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.visible	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.width.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.width.name replaces ToleranceCompartment[%d].*Indicator*.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DatumGroup.ExtendedText.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineName for compatiblity.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.bold	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.font	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.italic	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.italicAngle	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.language	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.lineFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.maxExtend	"%d"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.spaceFactor	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.Datum	String representing enumeration such	ToleranceCompartment[%d].*Indicat*	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Group.ExtendedText.strikethrough	that: "0"=none, "1"=single, "2"=double	*or*.DatumGroup.ExtendedText	BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textAspect	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textHeight	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textLineWidth.name	"%s"	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.symbol	String representing enumeration such that: "0"=parallel, "1"=perpendicular, "2"=angular, "3"=including	ToleranceCompartment[%d].*Indicator*	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].*Indicator*.type	String representing enumeration such that: "0"=intersectionplane, "1"=orientationplane, "2"=collectionplane, "3"=directionfeature	ToleranceCompartment[%d].*Indicator*	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
ToleranceCompartment[%d].Statistical	"%d"	ToleranceCompartment[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Text*	NULL	ToleranceCompartment[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d]	NULL	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].Outline	NULL	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	NULL	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Item[%d].Outline.colour replaces ToleranceCompartment[%d].*Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.Item[%d].outlineColour for compatiblity.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].O	"%d"	ToleranceCompartment[%d].*Text*.It	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
utline.filled		em[%d].Outline	TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Item[%d].Outline.style replaces ToleranceCompartment[%d].*Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.Item[%d].outlineStyle for compatiblity.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Item[

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			%d].Outline.type replaces ToleranceCompartment[%d].*Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.Item[%d].outlineType for compatiblity.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.visible	"%d"	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Text*.Item[%d].Outline.width.name	"%s"	ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Item[%d].Outline.width.name replaces ToleranceCompartment[%d].*Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.Item[%d].outlineName for compatiblity.
ToleranceCompartment[%d].*Text*.Item[%d].bold	"%d"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Text*.Item[%d].font	"%s"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].italic	"%d"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].italicAngle	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].language	"%s"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].lineFactor	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing,	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
ToleranceCompartment[%d].*Text*.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip,	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ToleranceCompartment[%d].*Text*.Item[%d].spaceFactor	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].string	"%s"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees,	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Text*.Item[%d].textAspect	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].textHeight	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].textLineWidth.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Text*.Item[%d].textLineWidth.name	"%s"	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline	NULL	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline.Side	NULL	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Text*.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Outline.Side.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Text*.Outline.Side.width.name	"%s"	ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
ToleranceCompartment[%d].*Text*.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		TextSuffix.
			ToleranceCompartment[%d].*Text*.Outline.colour replaces ToleranceCompartment[%d].*Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.outlineColour for compatiblity.
ToleranceCompartment[%d].*Text*.Outline.doubleOffset	"%.16g"	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline.filled	"%d"	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Outline.style replaces ToleranceCompartment[%d].*Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceCompartment[%d].*Text*.outlineStyle for compatiblity.
ToleranceCompartment[%d].*Text*.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Outline.type replaces ToleranceCompartment[%d].*Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.outlineType for compatiblity.
ToleranceCompartment[%d].*Text*.Outline.visible	"%d"	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.Outline.width.*measurementType*	"%.16g"	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Text*.Outline.width.name	"%s"	ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			ToleranceCompartment[%d].*Text*.Outline.width.name replaces ToleranceCompartment[%d].*Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ToleranceCompartment[%d].*Text*.outlineName for compatiblity.
ToleranceCompartment[%d].*Text*.bold	"%d"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.font	"%s"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.italic	"%d"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.italicAngle	"%.16g"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].*Text*.language	"%s"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.lineFactor	"%.16g"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.maxExtend	"%d"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.name	"%s"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.spaceFactor	"%.16g"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.subscript	String representing enumeration such that: "0"=sub, "1"=super	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.textAspect	"%.16g"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.textHeight	"%.16g"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.textLineWid	"%.16g"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	**“Value” Property Atom Value String Encoding Format**	**Prequist Key**	**AdditionalNote**
th.*measurementType*			TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
ToleranceCompartment[%d].*Text*.textLineWidth.name	"%s"	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].*Text*.underline	String representing enumeration such that: "0"=over, "1"=under	ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
ToleranceCompartment[%d].commonZone	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].maxBonus	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].maxBonusValue	"%.16g"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].maxBonusValueString	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	ToleranceCompartment[%d]	
ToleranceCompartment[%d].precision	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].prefix	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].projected	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].projectedValue	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].projectionVector	"%.16g %.16g %.16g"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].suffix	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].tangentPlane	"%d"	ToleranceCompartment[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ToleranceCompartment[%d].unequal	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].unequalDisplay	String representing enumeration such that: "0"=circleu, "1"=uz	ToleranceCompartment[%d]	
ToleranceCompartment[%d].unequalValue	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].unitBasis	"%d"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].unitBasisValue	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].value	"%s"	ToleranceCompartment[%d]	
ToleranceCompartment[%d].zoneShape	String representing enumeration such that: "0"=diameter, "1"=sphericaldiameter, "2"=square	ToleranceCompartment[%d]	
ValueToCustomer	"%s"		
_____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	_____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	_____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
allOver	"%d"		
angleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
characteristic	String representing enumeration such that: "0"=profileofaline, "1"=circularrunout, "2"=perpendicularity, "3"=position, "4"=totalrunout,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=profileofasurface, "6"=cylindricity, "7"=symmetry, "8"=angularity, "9"=parallelism, "10"=concentricity, "11"=flatness, "12"=circularity, "13"=straightness, "14"=axisintersection		
commaAsDecimal	"%d"		
datumOnLeader	String representing enumeration such that: "0"=none, "1"=solid, "2"=filled		
direction	"%.16g %.16g %.16g"		
fcfTextUnderline	String representing enumeration such that: "0"=none, "1"=top, "2"=all		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
leadingZero	"%d"		
lineFactor	"%.16g"		
maxBonus	"%d"		
maxBonusPrecision	"%d"		
maxBonusValue	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
profileType	String representing enumeration such that: "0"=bilateral, "1"=unilateralinside, "2"=unilateraloutside, "3"=bilateralunequal		
profileValue	"%.16g"		
profileValue2	"%.16g"		
projectedBelow	"%d"		
projectedPrecision	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
trailingZero	"%d"		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.5 PMI_DFS_TYPE (0x0084)

Table 209 — PMI_DFS_TYPE

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DatumFeatureSymbolText[%d]	NULL		DatumFeatureSymbolText[%d] index expected to range from 0-3 inclusive.Each instance of this is expected to have a different position such that DatumFeatureSymbolText[x].position != DatumFeatureSymbolText[y].position for any combination of x and y
DatumFeatureSymbolText[%d].Text	NULL	DatumFeatureSymbolText[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d]	NULL	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.Item[%d] .Outline	NULL	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .Outline.Side	NULL	DatumFeatureSymbolText[%d].Text .Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Item[%d] .Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text .Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Item[%d] .Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DatumFeatureSymbolText[%d].Text .Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Item[%d] .Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DatumFeatureSymbolText[%d].Text .Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Item[%d] .Outline.Side.width.*measurementType*	"%.16g"	DatumFeatureSymbolText[%d].Text .Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	DatumFeatureSymbolText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	DatumFeatureSymbolText[%d].Text.Item[%d].Outline.colour replaces DatumFeatureSymbolText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.Item[%d].outlineColour for compatiblity.
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.filled	"%d"	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	DatumFeatureSymbolText[%d].Text.Item[%d].Outline.style replaces DatumFeatureSymbolText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.Item[%d].outlineStyle for compatiblity.
DatumFeatureSymbolText[%d].Text.Item[%d]	String representing enumeration such that:	DatumFeatureSymbolText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.Outline.thickness	"0"=normal, "1"=thick, "2"=thin	.Item[%d].Outline	
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	DatumFeatureSymbolText[%d].Text.Item[%d].Outline.type replaces DatumFeatureSymbolText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.Item[%d].outlineType for compatiblity.
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.visible	"%d"	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DatumFeatureSymbolText[%d].Text.Item[%d].Outline.width.name	"%s"	DatumFeatureSymbolText[%d].Text.Item[%d].Outline	DatumFeatureSymbolText[%d].Text.Item[%d].Outline.width.name replaces DatumFeatureSymbolText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.Item[%d].outlineName for compatiblity.
DatumFeatureSymbolText[%d].Text.Item[%d].bold	"%d"	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].font	"%s"	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals,	DatumFeatureSymbolText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
DatumFeatureSymbolText[%d].Text.Item[%d].italic	"%d"	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].italicAngle	"%.16g"	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].language	"%s"	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].lineFactor	"%.16g"	DatumFeatureSymbolText[%d].Text.Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp,	DatumFeatureSymbolText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"20"=surfacefinishmrpcomplete, "21"=meltthrough		
DatumFeatureSymbolText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain,	DatumFeatureSymbolText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
DatumFeatureSymbolText[%d].Text.Item[%d] .spaceFactor	"%.16g"	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .string	"%s"	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .subscript	String representing enumeration such that: "0"=sub, "1"=super	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal,	DatumFeatureSymbolText[%d].Text .Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
DatumFeatureSymbolText[%d].Text.Item[%d] .textAspect	"%.16g"	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .textHeight	"%.16g"	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .textLineWidth.*measurementType*	"%.16g"	DatumFeatureSymbolText[%d].Text .Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
DatumFeatureSymbolText[%d].Text.Item[%d] .textLineWidth.name	"%s"	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Item[%d] .underline	String representing enumeration such that: "0"=over, "1"=under	DatumFeatureSymbolText[%d].Text .Item[%d]	
DatumFeatureSymbolText[%d].Text.Outline	NULL	DatumFeatureSymbolText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DatumFeatureSymbolText[%d].Text.Outline.Side	NULL	DatumFeatureSymbolText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DatumFeatureSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DatumFeatureSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	DatumFeatureSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DatumFeatureSymbolText[%d].Text.Outline.Side.width.name	"%s"	DatumFeatureSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DatumFeatureSymbolText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text.Outline	DatumFeatureSymbolText[%d].Text.Outline.colour replaces DatumFeatureSymbolText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.outlineColour for compatiblity.
DatumFeatureSymbolText[%d].Text.Outline.doubleOffset	"%.16g"	DatumFeatureSymbolText[%d].Text.Outline	
DatumFeatureSymbolText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text.Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DatumFeatureSymbolText[%d].Text.Outline.filled	"%d"	DatumFeatureSymbolText[%d].Text.Outline	
DatumFeatureSymbolText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DatumFeatureSymbolText[%d].Text.Outline	DatumFeatureSymbolText[%d].Text.Outline.style replaces DatumFeatureSymbolText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.outlineStyle for compatiblity.
DatumFeatureSymbolText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DatumFeatureSymbolText[%d].Text.Outline	
DatumFeatureSymbolText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DatumFeatureSymbolText[%d].Text.Outline	DatumFeatureSymbolText[%d].Text.Outline.type replaces DatumFeatureSymbolText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.outlineType for compatiblity.
DatumFeatureSymbolText[%d].Text.Outline.visible	"%d"	DatumFeatureSymbolText[%d].Text.Outline	
DatumFeatureSymbolText[%d].Text.Outline.width.*measurementType*	"%.16g"	DatumFeatureSymbolText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DatumFeatureSymbolText[%d].Text.Outline.width.name	"%s"	DatumFeatureSymbolText[%d].Text.Outline	DatumFeatureSymbolText[%d].Text.Outline.width.name replaces DatumFeatureSymbolText[%d].Text.o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DatumFeatureSymbolText[%d].Text.outlineName for compatiblity.
DatumFeatureSymbolText[%d].Text.bold	"%d"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.font	"%s"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.italic	"%d"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.italicAngle	"%.16g"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.language	"%s"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.lineFactor	"%.16g"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.maxExtend	"%d"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.name	"%s"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.spaceFactor	"%.16g"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.textAspect	"%.16g"	DatumFeatureSymbolText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DatumFeatureSymbolText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.textHeight	"%.16g"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.textLineWidth.*measurementType*	"%.16g"	DatumFeatureSymbolText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DatumFeatureSymbolText[%d].Text.textLineWidth.name	"%s"	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DatumFeatureSymbolText[%d].Text	
DatumFeatureSymbolText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	DatumFeatureSymbolText[%d]	
DatumTargetReference[%d]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.		
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond,	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=bottomcentre		
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.	"%s"	UnitSymbol[%d].Item[%d].Outline.S	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
name		ide	Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright,	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart,	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
datumOnLeader	String representing enumeration such that: "0"=none, "1"=solid, "2"=filled		
flag	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
font	"%s"		
group	"%s"		
individuallyApplied	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
label	"%s"		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that:		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.6 PMI_DTARGET_TYPE (0x0088)

Table 210 — PMI_DTARGET_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurement Type*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle,	Outline	Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side. width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.colour replaces TextFormat[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		Right.
UnitSymbol[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side .width.name	"%s"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	UnitSymbol[%d].Item[%d].O utline.colour replaces UnitSymbol[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o utlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.dou bleOffset	"%.16g"	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.fillC olour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.fille d	"%d"	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	UnitSymbol[%d].Item[%d]. Outline	UnitSymbol[%d].Item[%d].O utline.style replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*me*	"%.16g"	UnitSymbol[%d].Outline.Si	*Side* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
asurementType		de	one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre,	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reference is an origin reference.
accountabilityId	"%d"		
areaShape	String representing enumeration such that: "0"=diameter, "1"=sphericaldiameter, "2"=square		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
diameter	"%.16g"		
flag	"%d"		
font	"%s"		
group	"%s"		
index	"%d"		
innerDiameter	"%.16g"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
label	"%s"		
language	"%s"		
length	"%.16g"		
lineFactor	"%.16g"		
movable	"%d"		
movableTargetStubDirection	String representing enumeration such that: "0"=left, "1"=right,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=inferred		
movableTargetStubLength	"%.16g"		
moveableModifierAngle	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
planar	"%d"		
point1	"%.16g %.16g %.16g"		
point2	"%.16g %.16g %.16g"		
precision	"%d"		
radius	"%.16g"		
shapeTextOutsideSymbol	"%d"		
size	"%s"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
terminator	String representing enumeration such that: "0"=none, "1"=x, "2"=plus		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textStubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred		
textStubLength	"%.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
type	String representing enumeration such that: "0"=cylindrical, "1"=line, "2"=arbitrary, "3"=circular, "4"=point, "5"=annular, "6"=area, "7"=rectangular, "8"=spherical		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
width	"%.16g"		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.7 PMI_SPOT_WELD_TYPE (0x0004), PMI_ARC_SPOT_WELD_TYPE (0x0018), PMI_RES_SPOT_WELD_TYPE (0x0020), PMI_DOLLOP_TYPE (0x0028), PMI_CLINCH_TYPE (0x0040)

Table 211 — PMI_SPOT_WELD_TYPE, PMI_ARC_SPOT_WELD_TYPE, PMI_RES_SPOT_WELD_TYPE, PMI_DOLLOP_TYPE, PMI_CLINCH_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Outline	Outline.style replaces outlineStyle due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart,	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline,	*Text*.Item[%d]	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle,	*Text*.Outline	*Text* must be replaced by one of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.language	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal,	*Text*	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=vertical		one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side .width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.colour replaces TextFormat[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o utlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.dou bleOffset	"%.16g"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fillC olour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fille d	"%d"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.style replaces TextFormat[%d].Item[%d].o utlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
approachDirection	"%.16g %.16g %.16g"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
clampingDirection	"%.16g %.16g %.16g"		
commaAsDecimal	"%d"		
delta	"%d"		
diameter	"%.16g"		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		
flag	"%d"		
font	"%s"		
group	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
groupId	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
jointId	"%s"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
normalDirection	"%.16g %.16g %.16g"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
position	"%.16g %.16g %.16g"		
precision	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
thickness	"%.16g"		
thicknesses	"%d"		
type	String representing enumeration such that: "0"=unset, "1"=resistance, "2"=projection		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMI_PROP_APPROACH_DIR	“%f %f %f”		
PMI_PROP_CLAMPING_DIR	“%f %f %f”		
PMI_PROP_NORMAL_DIR	“%f %f %f”		
PMI_PROP_ANCHOR_POINT	“%f %f %f”		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.8 PMI_LINE_WELD_TYPE (0x0008),PMI_GROOVE_WELD_TYPE (0x0010),PMI_FILLET_WELD_TYPE (0x0011),PMI_SLOT_WELD_TYPE (0x0012),PMI_EDGE_WELD_TYPE (0x0014),PMI_RES_SEAM_WELD_TYPE (0x0021),PMI_BEAD_TYPE (0x0022),PMI_TAPE_TYPE (0x0024)

Table 212 — PMI_LINE_WELD_TYPE, PMI_GROOVE_WELD_TYPE, PMI_FILLET_WELD_TYPE, PMI_SLOT_WELD_TYPE, PMI_EDGE_WELD_TYPE, ,PMI_RES_SEAM_WELD_TYPE, PMI_BEAD_TYPE, PMI_TAPE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
Format	NULL		*Format* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d]	NULL	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline	NULL	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.Side	NULL	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.Side.width.name	"%s"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.colour replaces *Format*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineColour for compatiblity.
Format.Item[%d].Outline.doubleOffset	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.filled	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed,	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		TextFormat[%d].
			Format.Item[%d].Outline.style replaces *Format*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineStyle for compatiblity.
Format.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.type replaces *Format*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineType for compatiblity.
Format.Item[%d].Outline.visible	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].
Format.Item[%d].Outline.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.width.name	"%s"	*Format*.Item[%d].Outline	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.width.name replaces *Format*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineName for compatiblity.
Format.Item[%d].bold	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].font	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italic	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].italicAngle	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].language	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected,	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Format.Item[%d].spaceFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].string	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis,	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Format.Item[%d].textAspect	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textHeight	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].textLineWidth.name	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Format*.Item[%d]	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.bold	"%d"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.font	"%s"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.italic	"%d"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.italicAngle	"%.16g"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.language	"%s"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.lineFactor	"%.16g"	*Format*	*Format* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.spaceFactor	"%.16g"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.textAspect	"%.16g"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.textHeight	"%.16g"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.textLineWidth.*measurementType*	"%.16g"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.textLineWidth.name	"%s"	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
Format.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.underline	String representing enumeration such that: "0"=over, "1"=under	*Format*	*Format* must be replaced by one of GrooveGapTextFormat[%d] or TextFormat[%d].
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
LeaderText[%d]	NULL		LeaderText[%d] index expected to range from 0-1 inclusive.Each instance of this is expected to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			have a different position such that LeaderText[x].position != LeaderText[y].position for any combination of x and y
LeaderText[%d].Text	NULL	LeaderText[%d]	
LeaderText[%d].Text.Item[%d]	NULL	LeaderText[%d].Text	
LeaderText[%d].Text.Item[%d].Outline	NULL	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].Outline.Side	NULL	LeaderText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	LeaderText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	LeaderText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	LeaderText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
LeaderText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	LeaderText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Item[%d].Outline	LeaderText[%d].Text.Item[%d].Outline.colour replaces LeaderText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineColour for compatiblity.
LeaderText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	LeaderText[%d].Text.Item[%d].Outline	
LeaderText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Item[%d].Outline	
LeaderText[%d].Text.Item[%d].Outline.filled	"%d"	LeaderText[%d].Text.Item[%d].Outline	
LeaderText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	LeaderText[%d].Text.Item[%d].Outline	LeaderText[%d].Text.Item[%d].Outline.style replaces LeaderText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.Item[%d].outlineStyle for compatiblity.
LeaderText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	LeaderText[%d].Text.Item[%d].Outline	
LeaderText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	LeaderText[%d].Text.Item[%d].Outline	LeaderText[%d].Text.Item[%d].Outline.type replaces LeaderText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.Item[%d].outlineType for compatiblity.
LeaderText[%d].Text.Item[%d].Outline.visible	"%d"	LeaderText[%d].Text.Item[%d].Outline	
LeaderText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	LeaderText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
LeaderText[%d].Text.Item[%d].Outline.width.name	"%s"	LeaderText[%d].Text.Item[%d].Outline	LeaderText[%d].Text.Item[%d].Outline.width.name replaces LeaderText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.Item[%d].outlineName for compatiblity.
LeaderText[%d].Text.Item[%d].bold	"%d"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].font	"%s"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].italic	"%d"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].italicAngle	"%.16g"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].language	"%s"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].lineFactor	"%.16g"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic,	LeaderText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
LeaderText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].spaceFactor	"%.16g"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].string	"%s"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub,	LeaderText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=super		
LeaderText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].textAspect	"%.16g"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].textHeight	"%.16g"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].textLineWi	"%.16g"	LeaderText[%d].Text.Item[%d]	*measurementType* must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
dth.*measurementType*			replaced with meterWidth or pixelWidth
LeaderText[%d].Text.Item[%d].textLineWidth.name	"%s"	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	LeaderText[%d].Text.Item[%d]	
LeaderText[%d].Text.Outline	NULL	LeaderText[%d].Text	
LeaderText[%d].Text.Outline.Side	NULL	LeaderText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	LeaderText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	LeaderText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	LeaderText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
LeaderText[%d].Text.Outline.Side.width.name	"%s"	LeaderText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
LeaderText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Outline	LeaderText[%d].Text.Outline.colour replaces LeaderText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			LeaderText[%d].Text.outlineColour for compatiblity.
LeaderText[%d].Text.Outline.doubleOffset	"%.16g"	LeaderText[%d].Text.Outline	
LeaderText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text.Outline	
LeaderText[%d].Text.Outline.filled	"%d"	LeaderText[%d].Text.Outline	
LeaderText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	LeaderText[%d].Text.Outline	LeaderText[%d].Text.Outline.style replaces LeaderText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.outlineStyle for compatiblity.
LeaderText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	LeaderText[%d].Text.Outline	
LeaderText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	LeaderText[%d].Text.Outline	LeaderText[%d].Text.Outline.type replaces LeaderText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.outlineType for compatiblity.
LeaderText[%d].Text.Outline.visible	"%d"	LeaderText[%d].Text.Outline	
LeaderText[%d].Text.Outline.width.*measurementType*	"%.16g"	LeaderText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
LeaderText[%d].Text.Outline.width.name	"%s"	LeaderText[%d].Text.Outline	LeaderText[%d].Text.Outline.wid

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			th.name replaces LeaderText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of LeaderText[%d].Text.outlineName for compatiblity.
LeaderText[%d].Text.bold	"%d"	LeaderText[%d].Text	
LeaderText[%d].Text.font	"%s"	LeaderText[%d].Text	
LeaderText[%d].Text.italic	"%d"	LeaderText[%d].Text	
LeaderText[%d].Text.italicAngle	"%.16g"	LeaderText[%d].Text	
LeaderText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	LeaderText[%d].Text	
LeaderText[%d].Text.language	"%s"	LeaderText[%d].Text	
LeaderText[%d].Text.lineFactor	"%.16g"	LeaderText[%d].Text	
LeaderText[%d].Text.maxExtend	"%d"	LeaderText[%d].Text	
LeaderText[%d].Text.name	"%s"	LeaderText[%d].Text	
LeaderText[%d].Text.spaceFactor	"%.16g"	LeaderText[%d].Text	
LeaderText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	LeaderText[%d].Text	
LeaderText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	LeaderText[%d].Text	
LeaderText[%d].Text.textAspect	"%.16g"	LeaderText[%d].Text	
LeaderText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	LeaderText[%d].Text	
LeaderText[%d].Text.textHeight	"%.16g"	LeaderText[%d].Text	
LeaderText[%d].Text.textLineWidth.*measurementType*	"%.16g"	LeaderText[%d].Text	*measurementType* must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
LeaderText[%d].Text.textLineWidth.name	"%s"	LeaderText[%d].Text	
LeaderText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	LeaderText[%d].Text	
LeaderText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	LeaderText[%d].Text	
LeaderText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	LeaderText[%d].Text	
LeaderText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	LeaderText[%d]	
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
SurfaceFinish[%d]	NULL		
SurfaceFinish[%d].Description	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].DisplayPlane.origin	"%.16g %.16g %.16g"	SurfaceFinish[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
SurfaceFinish[%d].DisplayPlane.xaxis	"%.16g %.16g %.16g"	SurfaceFinish[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
SurfaceFinish[%d].DisplayPlane.zaxis	"%.16g %.16g %.16g"	SurfaceFinish[%d]	This is property records the original defintion of the PMI display plane Should also have

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			'DisplayPlane.origin' and 'DisplayPlane.xaxis'
SurfaceFinish[%d].FeatureIdentification	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength,	SurfaceFinish[%d]	It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
SurfaceFinish[%d].GeneralAttribute[%d].ListValue[%d]	"%s"	SurfaceFinish[%d].GeneralAttribute[%d]	
SurfaceFinish[%d].Keyword[%d]	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].LAYER	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].Leader[%d]	NULL	SurfaceFinish[%d]	
SurfaceFinish[%d].Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	SurfaceFinish[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	SurfaceFinish[%d].Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
SurfaceFinish[%d].Leader[%d].allAroundSymbolSize	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowAngle	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowLength	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowOutSideLength	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble,	SurfaceFinish[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
SurfaceFinish[%d].Leader[%d].arrowWidth.*measurementType*	"%.16g"	SurfaceFinish[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].Leader[%d].arrowWidth.name	"%s"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].direction	"%.16g %.16g %.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].dotDiameter	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionJogAngle	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionJogEnd	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionJogOut	"%d"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionJogStart	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionLineExtension	"%.16g"	SurfaceFinish[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].Leader[%d].extensionLineGap	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].extensionWidth.*measurementType*	"%.16g"	SurfaceFinish[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].Leader[%d].extensionWidth.name	"%s"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].lineTextGap	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].offsetCentre	"%d"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].perpendicularToStub	"%d"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].perpendicularToTerminator	"%d"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].radiusToCentre	"%d"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].stubDirecti	String representing enumeration such that: "0"=left,	SurfaceFinish[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
on	"1"=right, "2"=inferred		
SurfaceFinish[%d].Leader[%d].stubLength	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].suppressed	"%d"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].tParm	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].terminator	"%.16g %.16g %.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].textOverLeaderFactor	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].textOverStubFactor	"%.16g"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Leader[%d].uvParm.U	"%.16g"	SurfaceFinish[%d].Leader[%d]	Expect to also have Leader[%d].uvParam.V.
SurfaceFinish[%d].Leader[%d].uvParm.V	"%.16g"	SurfaceFinish[%d].Leader[%d]	Expect to also have Leader[%d].uvParam.U.
SurfaceFinish[%d].Leader[%d].width.*measurementType*	"%.16g"	SurfaceFinish[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].Leader[%d].width.name	"%s"	SurfaceFinish[%d].Leader[%d]	
SurfaceFinish[%d].Outline	NULL	SurfaceFinish[%d]	
SurfaceFinish[%d].Outline.Side	NULL	SurfaceFinish[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].Outline.Side.width.*mea*	"%.16g"	SurfaceFinish[%d].Outline.Side	*Side* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
surementType			Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].Outline.Side.width.name	"%s"	SurfaceFinish[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].Outline	SurfaceFinish[%d].Outline.colour replaces SurfaceFinish[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].outlineColour for compatiblity.
SurfaceFinish[%d].Outline.doubleOffset	"%.16g"	SurfaceFinish[%d].Outline	
SurfaceFinish[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].Outline	
SurfaceFinish[%d].Outline.filled	"%d"	SurfaceFinish[%d].Outline	
SurfaceFinish[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].Outline	SurfaceFinish[%d].Outline.style replaces SurfaceFinish[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].outlineStyle for compatiblity.
SurfaceFinish[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].Outline	
SurfaceFinish[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline,	SurfaceFinish[%d].Outline	SurfaceFinish[%d].Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		SurfaceFinish[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].outlineType for compatiblity.
SurfaceFinish[%d].Outline.visible	"%d"	SurfaceFinish[%d].Outline	
SurfaceFinish[%d].Outline.width.*measurementType*	"%.16g"	SurfaceFinish[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].Outline.width.name	"%s"	SurfaceFinish[%d].Outline	SurfaceFinish[%d].Outline.width.name replaces SurfaceFinish[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].outlineName for compatiblity.
SurfaceFinish[%d].PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d]	
SurfaceFinish[%d].PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d]	
SurfaceFinish[%d].PMITextFlatToScreenOpacity	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d]	
SurfaceFinish[%d].PMITextInPlaneOpacity	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].RevisionModifier	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].SafetyClassification	"%s"	SurfaceFinish[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].*Text*	NULL	SurfaceFinish[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d]	NULL	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].Outline	NULL	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].Outline. Side	NULL	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Item[%d].Outline. Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Item[%d].Outline. Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Item[%d].Outline. Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Item[%d].Outline. Side.width.*measurementType*	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].*Text*.Item[%d].Outline.Side.width.name	"%s"	SurfaceFinish[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Item[%d].Outline.colour replaces SurfaceFinish[%d].*Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.Item[%d].outlineColour for compatiblity.
SurfaceFinish[%d].*Text*.Item[%d].Outline.doubleOffset	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].Outline.filled	"%d"	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].*Text*.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Item[%d].Outline.style replaces SurfaceFinish[%d].*Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.Item[%d].outlineStyle for compatiblity.
SurfaceFinish[%d].*Text*.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Item[%d].Outline.type replaces SurfaceFinish[%d].*Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			SurfaceFinish[%d].*Text*.Item[%d] .outlineType for compatiblity.
SurfaceFinish[%d].*Text*.Item[%d].Outline. visible	"%d"	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].Outline. width.*measurementType*	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].*Text*.Item[%d].Outline. width.name	"%s"	SurfaceFinish[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Item[%d] .Outline.width.name replaces SurfaceFinish[%d].*Text*.Item[%d] .outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.Item[%d] .outlineName for compatiblity.
SurfaceFinish[%d].*Text*.Item[%d].bold	"%d"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].font	"%s"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans,	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].italic	"%d"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].italicAng le	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].justificat ion	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].languag e	"%s"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].lineFact or	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].lineWel dSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].lineWel dSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove,	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
SurfaceFinish[%d].*Text*.Item[%d].spaceFactor	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].string	"%s"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].symbol	String representing enumeration such that:	SurfaceFinish[%d].*Text*.Item[%d	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	]	ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].textAspect	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].textHeig	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ht		]	ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].textLineWidth.*measurementType*	"%.16g"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].*Text*.Item[%d].textLineWidth.name	"%s"	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	SurfaceFinish[%d].*Text*.Item[%d]	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline	NULL	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline.Side	NULL	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed,	SurfaceFinish[%d].*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Outline.Side.width.*measurementType*	"%.16g"	SurfaceFinish[%d].*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].*Text*.Outline.Side.width.name	"%s"	SurfaceFinish[%d].*Text*.Outline.Side	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].*Text*.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Outline.colour replaces SurfaceFinish[%d].*Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.outlineCo

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			lour for compatiblity.
SurfaceFinish[%d].*Text*.Outline.doubleOffset	"%.16g"	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline.filled	"%d"	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Outline.style replaces SurfaceFinish[%d].*Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.outlineStyle for compatiblity.
SurfaceFinish[%d].*Text*.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset,	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"20"=scoredrectangle, "21"=parallelogram		
			SurfaceFinish[%d].*Text*.Outline.type replaces SurfaceFinish[%d].*Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.outlineType for compatiblity.
SurfaceFinish[%d].*Text*.Outline.visible	"%d"	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.Outline.width.*measurementType*	"%.16g"	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].*Text*.Outline.width.name	"%s"	SurfaceFinish[%d].*Text*.Outline	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			SurfaceFinish[%d].*Text*.Outline.width.name replaces SurfaceFinish[%d].*Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].*Text*.outlineName for compatiblity.
SurfaceFinish[%d].*Text*.bold	"%d"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.font	"%s"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.italic	"%d"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.italicAngle	"%.16g"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.language	"%s"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.lineFactor	"%.16g"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.maxExtend	"%d"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.name	"%s"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.spaceFactor	"%.16g"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
SurfaceFinish[%d].*Text*.subscript	String representing enumeration such that: "0"=sub, "1"=super	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.textAspect	"%.16g"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.textHeight	"%.16g"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.textLineWidth.*measurementType*	"%.16g"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].*Text*.textLineWidth.name	"%s"	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].*Text*.underline	String representing enumeration such that: "0"=over, "1"=under	SurfaceFinish[%d].*Text*	*Text* must be replaced by one of ProcessText, LayText or UnitSymbol[%d].
SurfaceFinish[%d].TextFormat[%d]	NULL	SurfaceFinish[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].TextFormat[%d].Item[%d]	NULL	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	NULL	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side	NULL	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.colour replaces SurfaceFinish[%d].TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].TextFormat[%d].Item[%d].outlineColour for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.filled	"%d"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.style replaces SurfaceFinish[%d].TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].TextFormat[%d].Item[%d].outlineStyle for compatiblity.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.type replaces SurfaceFinish[%d].TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].TextFormat[%d].Item[%d].outlineType for

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.visible	"%d"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline.width.name	"%s"	SurfaceFinish[%d].TextFormat[%d].Item[%d].Outline	SurfaceFinish[%d].TextFormat[% d].Item[%d].Outline.width.name replaces SurfaceFinish[%d].TextFormat[% d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of SurfaceFinish[%d].TextFormat[% d].Item[%d].outlineName for compatiblity.
SurfaceFinish[%d].TextFormat[%d].Item[%d].bold	"%d"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].font	"%s"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].italic	"%d"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].italicAngle	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright,	SurfaceFinish[%d].TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
SurfaceFinish[%d].TextFormat[%d].Item[%d].language	"%s"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].lineFactor	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead,	SurfaceFinish[%d].TextFormat[%d].Item[%d]	

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
SurfaceFinish[%d].TextFormat[%d].Item[%d].spaceFactor	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].string	"%s"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius,	SurfaceFinish[%d].TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
SurfaceFinish[%d].TextFormat[%d].Item[%d].textAspect	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].textHeight	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].TextFormat[%d].Item[%d].textLineWidth.name	"%s"	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	SurfaceFinish[%d].TextFormat[%d].Item[%d]	
SurfaceFinish[%d].TextFormat[%d].bold	"%d"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].font	"%s"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].italic	"%d"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].italicA ngle	"%.16g"	SurfaceFinish[%d].TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].language	"%s"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].lineFactor	"%.16g"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].spaceFactor	"%.16g"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].textAspect	"%.16g"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].textHeight	"%.16g"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	SurfaceFinish[%d].TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].TextFormat[%d].textLineWidth.name	"%s"	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	SurfaceFinish[%d].TextFormat[%d]	
SurfaceFinish[%d].ValueToCustomer	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	SurfaceFinish[%d].____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
SurfaceFinish[%d].____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	SurfaceFinish[%d].____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
SurfaceFinish[%d].____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the	SurfaceFinish[%d]	*reference* is the property value encoded also into the key

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	path.		identifying which reference is an origin reference.
SurfaceFinish[%d].accountabilityId	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].allAround	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].allAroundLeader	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove	SurfaceFinish[%d]	
SurfaceFinish[%d].blanked	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].bold	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].bottomExtensionLine	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].commaAsDecimal	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].direction	"%.16g %.16g %.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].flag	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].font	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].group	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].invertSymbol	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].invertText	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].italic	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].italicAngle	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].jisSurfaceTexture	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	SurfaceFinish[%d]	
SurfaceFinish[%d].language	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].lay	String representing enumeration such that: "0"=parallel,	SurfaceFinish[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=multidirectional, "2"=perpendicular, "3"=circular, "4"=particulate, "5"=angularboth, "6"=radial, "7"=userdefined		
SurfaceFinish[%d].lineFactor	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].lowerDelta	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].machiningAllowance	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].maxRoughness	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].maxRoughness2	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].minRoughness	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].minRoughness2	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].modifier	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].nameComponents.displayName	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].nameComponents.name	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright	SurfaceFinish[%d]	
SurfaceFinish[%d].panZoom	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].parenthesis	String representing enumeration such that: "0"=none, "1"=left, "2"=right, "3"=both	SurfaceFinish[%d]	
SurfaceFinish[%d].precision	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].roughness	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].roughness2	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].roughness3	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].roughnessCutoff	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].roughnessSpacing	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].singleLine	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].spaceFactor	"%.16g"	SurfaceFinish[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].stackFactor	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].standard	String representing enumeration such that: "0"=jis, "1"=ansi_y1436_1993, "2"=iso, "3"=asme_y1436m_1996, "4"=din, "5"=gb, "6"=iso_1302_2002, "7"=eskd, "8"=din_en_iso_1302_2002	SurfaceFinish[%d]	
SurfaceFinish[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	SurfaceFinish[%d]	
SurfaceFinish[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	SurfaceFinish[%d]	
SurfaceFinish[%d].symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d]	
SurfaceFinish[%d].textAspect	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	SurfaceFinish[%d]	
SurfaceFinish[%d].textDirection	"%.16g %.16g %.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].textHeight	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].textLineWidth.*measurementType*	"%.16g"	SurfaceFinish[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
SurfaceFinish[%d].textLineWidth.name	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].textOrigin	"%.16g %.16g %.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].textRotationPoint	"%.16g %.16g %.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	SurfaceFinish[%d]	
SurfaceFinish[%d].toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral	SurfaceFinish[%d]	
SurfaceFinish[%d].toleranceLeadingZero	"%d"	SurfaceFinish[%d]	
SurfaceFinish[%d].toleranceTrailingZero	"%d"	SurfaceFinish[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SurfaceFinish[%d].type	String representing enumeration such that: "0"=basic, "1"=mrr, "2"=mrp	SurfaceFinish[%d]	
SurfaceFinish[%d].underline	String representing enumeration such that: "0"=over, "1"=under	SurfaceFinish[%d]	
SurfaceFinish[%d].unit	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].upperDelta	"%.16g"	SurfaceFinish[%d]	
SurfaceFinish[%d].uriRefs[%d]	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].usage	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown	SurfaceFinish[%d]	
SurfaceFinish[%d].wavinessHeight	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].wavinessSpacing	"%s"	SurfaceFinish[%d]	
SurfaceFinish[%d].zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit	SurfaceFinish[%d]	
Text	NULL		*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline,	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		ProcessText or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Side must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d],

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ProcessText or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d], ProcessText or UnitSymbol[%d].
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
arrowSideAboveLine	"%d"		
arrowSideCompoundLongitudinalSize	"%s"		
arrowSideCompoundMainSize	"%s"		
arrowSideCompoundStaggeredSize	"%s"		
arrowSideCompoundWeld	"%d"		
arrowSideFinishSymbol	String representing enumeration such that: "0"=unset, "1"=unspecified, "2"=machining, "3"=grinding, "4"=chipping, "5"=none, "6"=hammering, "7"=peening, "8"=rolling, "9"=finishing		
arrowSideGrooveAngle	"%s"		
arrowSideGrooveGap	"%s"		
arrowSideLongitudinalSize	"%s"		
arrowSideMainSize	"%s"		
arrowSideStaggeredSize	"%s"		
arrowSideSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
arrowSideSupplementalSymbol2	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
arrowSideSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
contour	String representing enumeration such that: "0"=unset, "1"=closed, "2"=open, "3"=trilateral, "4"=allaround		
delta	"%d"		
fieldSpaceFactor	"%.16g"		
fieldWeld	"%d"		
fieldWeldDirection	String representing enumeration such that: "0"=up, "1"=down		
fieldWeldType	String representing enumeration such that: "0"=common, "1"=eskd		
flag	"%d"		
font	"%s"		
glue	"%d"		
grooveGapTextStyle	String representing enumeration such that: "0"=inside, "1"=outside, "2"=arrow		
group	"%s"		
identificationBelowReference	"%d"		
identificationLine	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
italic	"%d"		
italicAngle	"%.16g"		
jisArrowSidePartialPenetrationGrooveWeld	"%d"		
jisOtherSidePartialPenetrationGrooveWeld	"%d"		
jisStaggeredFilletWeld	"%d"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
lineGap	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
otherSideCompoundLongitudinalSize	"%s"		
otherSideCompoundMainSize	"%s"		
otherSideCompoundStaggeredSize	"%s"		
otherSideCompoundWeld	"%d"		
otherSideFinishSymbol	String representing enumeration such that: "0"=unset, "1"=unspecified, "2"=machining, "3"=grinding, "4"=chipping, "5"=none, "6"=hammering, "7"=peening, "8"=rolling, "9"=finishing		
otherSideGrooveAngle	"%s"		
otherSideGrooveGap	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
otherSideLongitudinalSize	"%s"		
otherSideMainSize	"%s"		
otherSideStaggeredSize	"%s"		
otherSideSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
otherSideSupplementalSymbol2	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
otherSideSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
panZoom	"%d"		
precision	"%d"		
referenceSign	"%d"		
referenceSignType	String representing enumeration such that: "0"=eferencesignfork, "1"=eferencesignbox		
solder	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
staggeredWeld	"%d"		
standard	String representing enumeration such that: "0"=ansiaws_a24-98 "1"=iso_2556 "2"=jis_z_3021 "3"=din "4"=eskd "5"=gb		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	green and red for the colour are encoded in the string		
symmetric	"%d"		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.9 PMI_MATERIAL_SPEC_TYPE (0x0231)

Table 213 — PMI_MATERIAL_SPEC_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminato r	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset,	Outline	Outline.style replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal,	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=thick, "2"=thin		of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete,	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.colour replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline,	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over,	*Text*	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=under		of Nomenclature, OpenField or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outlin e	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outlin e.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outlin e.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outlin e.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outlin e.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side .width.name	"%s"	TextFormat[%d].Item[%d].Outlin e.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outlin e	TextFormat[%d].Item[%d].Outli ne.colour replaces TextFormat[%d].Item[%d].outli neColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outli neColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
identifier	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
revision	"%s"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.10 PMI_PROCESS_SPEC_TYPE (0x0232)

Table 214 — PMI_PROCESS_SPEC_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.nam e	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.colou r replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Item[%d].Outline.width .name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalS	String representing enumeration such that: "0"=unset, "1"=convex,	*Text*.Item[%d]	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ymbol	"2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single,	*Text*.Item[%d]	*Text* must be replaced by

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=double		one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			OpenField or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Nomenclature,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			OpenField or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.font	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of Nomenclature, OpenField or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side .width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.colour replaces TextFormat[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o utlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.dou bleOffset	"%.16g"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fillC olour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fille d	"%d"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].O utline.style replaces TextFormat[%d].Item[%d].o utlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	for the colour are encoded in the string		
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
_____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	_____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	_____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
identifier	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
revision	"%s"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.11 PMI_PART_SPEC_TYPE (0x0233)

Table 215 — PMI_PART_SPEC_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminato r	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side .width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineNa

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			me due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
descriptiveModifier	"%s"		
flag	"%d"		
font	"%s"		
group	"%s"		
identifier	"%s"		
italic	"%d"		
italicAngle	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
itemName	"%s"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
nameModifier	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
revision	"%s"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.12 PMI_NOTE_TYPE (0x0100),PMI_FACE_ATTR_TYPE (0x0101),PMI_MV_LABEL_TYPE (0x0102),PMI_WELD_NOTE_TYPE (0x0309)

Table 216 — PMI_NOTE_TYPE,PMI_FACE_ATTR_TYPE,PMI_MV_LABEL_TYPE,PMI_WELD_NOTE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
NoteText[%d]	NULL		
NoteText[%d].Text	NULL	NoteText[%d]	
NoteText[%d].Text.Item[%d]	NULL	NoteText[%d].Text	
NoteText[%d].Text.Item[%d].Outline	NULL	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].Outline.Side	NULL	NoteText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d].Outline.colour replaces NoteText[%d].Text.Item[%d].outlineColour due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d]. outlineColour for compatiblity.
NoteText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.filled	"%d"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d]. Outline.style replaces NoteText[%d].Text.Item[%d]. outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d]. outlineStyle for compatiblity.
NoteText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d]. Outline.type replaces NoteText[%d].Text.Item[%d]. outlineType due to the increasing complexity of Outline semantics. We recommend continuing to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			support read/write of NoteText[%d].Text.Item[%d]. outlineType for compatiblity.
NoteText[%d].Text.Item[%d].Outline.visible	"%d"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].Outline.width.name	"%s"	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d]. Outline.width.name replaces NoteText[%d].Text.Item[%d]. outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d]. outlineName for compatiblity.
NoteText[%d].Text.Item[%d].bold	"%d"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].font	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].italic	"%d"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].italicAngle	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft,	NoteText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=bottomright, "8"=bottomcentre		
NoteText[%d].Text.Item[%d].language	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineFactor	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber,	NoteText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"57"=isoedge		
NoteText[%d].Text.Item[%d].spaceFactor	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].string	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textAspect	"%.16g"	NoteText[%d].Text.Item[%d]	

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textHeight	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].textLineWidth.name	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Outline	NULL	NoteText[%d].Text	
NoteText[%d].Text.Outline.Side	NULL	NoteText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.width. *measurementType*	"%.16g"	NoteText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Outline.Side.width.name	"%s"	NoteText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.colour replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			NoteText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineColour for compatiblity.
NoteText[%d].Text.Outline.doubleOffset	"%.16g"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.filled	"%d"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.style replaces NoteText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineStyle for compatiblity.
NoteText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.type replaces NoteText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of NoteText[%d].Text.outlineType for compatiblity.
NoteText[%d].Text.Outline.visible	"%d"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.width.*measurementType*	"%.16g"	NoteText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Outline.width.name	"%s"	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.width.name replaces NoteText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineName for compatiblity.
NoteText[%d].Text.bold	"%d"	NoteText[%d].Text	
NoteText[%d].Text.font	"%s"	NoteText[%d].Text	
NoteText[%d].Text.italic	"%d"	NoteText[%d].Text	
NoteText[%d].Text.italicAngle	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	NoteText[%d].Text	
NoteText[%d].Text.language	"%s"	NoteText[%d].Text	
NoteText[%d].Text.lineFactor	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.maxExtend	"%d"	NoteText[%d].Text	
NoteText[%d].Text.name	"%s"	NoteText[%d].Text	
NoteText[%d].Text.spaceFactor	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	NoteText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	NoteText[%d].Text	
NoteText[%d].Text.textAspect	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text	
NoteText[%d].Text.textHeight	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.textLineWidth.*measurementType*	"%.16g"	NoteText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.textLineWidth.name	"%s"	NoteText[%d].Text	
NoteText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	NoteText[%d].Text	
NoteText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text	
NoteText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	NoteText[%d].Text	
NoteText[%d].textDirection	"%.16g %.16g %.16g"	NoteText[%d]	
NoteText[%d].textOrigin	"%.16g %.16g %.16g"	NoteText[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross,	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle,	*Text*.Outline	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side. width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colou r	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].Ou tline.colour replaces TextFormat[%d].Item[%d].ou tlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].ou tlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doub	"%.16g"	TextFormat[%d].Item[%d].	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
leOffset		Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft,	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=bottomright, "8"=bottomcentre		
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=leardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
category	"%s"		
commaAsDecimal	"%d"		
datumOnLeader	String representing enumeration such that: "0"=none, "1"=solid, "2"=filled		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		
flag	"%d"		
font	"%s"		
group	"%s"		
identifier	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
revision	"%s"		
spaceFactor	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.13 PMI_BALLOON_TYPE (0x0235)

Table 217 — PMI_BALLOON_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
NoteText[%d]	NULL		
NoteText[%d].Text	NULL	NoteText[%d]	
NoteText[%d].Text.Item[%d]	NULL	NoteText[%d].Text	
NoteText[%d].Text.Item[%d].Outline	NULL	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].Outline.Side	NULL	NoteText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		
NoteText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d].Outline.colour replaces NoteText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d].outlineColour for compatiblity.
NoteText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.filled	"%d"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d].Outline.style replaces NoteText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d].outlineStyle for compatiblity.
NoteText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d].Outline.type replaces NoteText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d].outlineType for compatiblity.
NoteText[%d].Text.Item[%d].Outline.visible	"%d"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].Outline.width.name	"%s"	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d].Outline.width.name replaces NoteText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d].outlineName for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
NoteText[%d].Text.Item[%d].bold	"%d"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].font	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].italic	"%d"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].italicAngle	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].language	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineFactor	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug,	NoteText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
NoteText[%d].Text.Item[%d].spaceFactor	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].string	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity,	NoteText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
NoteText[%d].Text.Item[%d].textAspect	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textHeight	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].textLineWidth.name	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Outline	NULL	NoteText[%d].Text	
NoteText[%d].Text.Outline.Side	NULL	NoteText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.thickne ss	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.width. *measurementType*	"%.16g"	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Outline.Side.width.n ame	"%s"	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.c olour replaces NoteText[%d].Text.outlineCo lour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineCo lour for compatiblity.
NoteText[%d].Text.Outline.doubleOffse t	"%.16g"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.filled	"%d"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.st yle replaces NoteText[%d].Text.outlineSty le due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineStyle for compatiblity.
NoteText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.type replaces NoteText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineType for compatiblity.
NoteText[%d].Text.Outline.visible	"%d"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.width.*measurementType*	"%.16g"	NoteText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Outline.width.name	"%s"	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.width.name replaces NoteText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineName for compatiblity.
NoteText[%d].Text.bold	"%d"	NoteText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.font	"%s"	NoteText[%d].Text	
NoteText[%d].Text.italic	"%d"	NoteText[%d].Text	
NoteText[%d].Text.italicAngle	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	NoteText[%d].Text	
NoteText[%d].Text.language	"%s"	NoteText[%d].Text	
NoteText[%d].Text.lineFactor	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.maxExtend	"%d"	NoteText[%d].Text	
NoteText[%d].Text.name	"%s"	NoteText[%d].Text	
NoteText[%d].Text.spaceFactor	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	NoteText[%d].Text	
NoteText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	NoteText[%d].Text	
NoteText[%d].Text.textAspect	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text	
NoteText[%d].Text.textHeight	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.textLineWidth.*measurementType*	"%.16g"	NoteText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.textLineWidth.name	"%s"	NoteText[%d].Text	
NoteText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	NoteText[%d].Text	
NoteText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text	
NoteText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	NoteText[%d].Text	
NoteText[%d].textDirection	"%.16g %.16g %.16g"	NoteText[%d]	
NoteText[%d].textOrigin	"%.16g %.16g %.16g"	NoteText[%d]	
Outline	NULL		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej,	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate,	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset,	*Text*.Outline	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle,	TextFormat[%d].Item[%d].	TextFormat[%d].Item[%d].Ou

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	tline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
balloonId	"%s"		
blanked	"%d"		
bold	"%d"		
category	"%s"		
commaAsDecimal	"%d"		
datumOnLeader	String representing enumeration such that: "0"=none, "1"=solid, "2"=filled		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		
flag	"%d"		
font	"%s"		
group	"%s"		
identifier	"%s"		
italic	"%d"		
italicAngle	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
radius	"%.16g"		
revision	"%s"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.14 PMI_MEAS_PT_TYPE (0x0042), PMI_DATUM_PT_TYPE (0x0120), PMI_SURF_VEC_TYPE (0x0121), PMI_HOLE_VEC_TYPE (0x0122), PMI_TRIM_VEC_TYPE (0x0124), PMI_HEM_VEC_TYPE (0x0128)

Table 218 — PMI_MEAS_PT_TYPE, PMI_DATUM_PT_TYPE, PMI_SURF_VEC_TYPE, PMI_HOLE_VEC_TYPE, PMI_TRIM_VEC_TYPE, PMI_HEM_VEC_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		Hexadecimal integer representing RGB colour where value has “0x00bbggrr” form. The low-order byte contains a value for the relative intensity of red; the second byte contains a value for the relative intensity of green; and the third byte contains a value for the relative intensity of blue. The high-order byte shall be zero. The maximum value for a single byte is 0xFF (i.e. intensity value is in the range [0:255]).
PMITextInPlaneOpacity	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colou

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			r replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width .name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red	*Text*.Outline	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	for the colour are encoded in the string		one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visib	"%d"	TextFormat[%d].Item[%d].	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
le		Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
coordinationDirection	"%.16g %.16g %.16g"		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
level	"%s"		
lineFactor	"%.16g"		
measurementDirection	"%.16g %.16g %.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
normalDirection	"%.16g %.16g %.16g"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=bottomright		
panZoom	"%d"		
position	"%.16g %.16g %.16g"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMI_PROP_MEAS_LEVEL	“%d”		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMI_PROP_MEAS_DIR	“%f %f %f”		
PMI_PROP_COORD_DIR	“%f %f %f”		
PMI_PROP_NORMAL_DIR	“%f %f %f”		
PMI_PROP_ANCHOR_POINT	“%f %f %f”		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.15 PMI_DATUM_LOC_TYPE (0x0044),PMI_MEAS_LOC_TYPE (0x0118)

Table 219 — PMI_DATUM_LOC_TYPE, PMI_MEAS_LOC_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle,	Outline	Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width .name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev,	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline,	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of PartNumber, Note,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of PartNumber, Note,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of PartNumber, Note, CoordinatedEntity[%d] or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft,	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=bottomright, "8"=bottomcentre		
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
changeLevel	String representing enumeration such that: "0"=soft, "1"=medium, "2"=hard		
commaAsDecimal	"%d"		
coordinatePlane	String representing enumeration such that: "0"=y, "1"=xz, "2"=xy, "3"=yz, "4"=z, "5"=xyz, "6"=x		
flag	"%d"		
font	"%s"		
functionalSubscript	"%s"		
group	"%s"		
hotSpotPosition	"%.16g %.16g %.16g"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
mainEdgeType	String representing enumeration such that: "0"=ce, "1"=e, "2"=e, "3"=ek, "4"=co		
mainHoleType	String representing enumeration such that: "0"=dl, "1"=h, "2"=hk, "3"=ch, "4"=hk, "5"=ac, "6"=h, "7"=bl, "8"=ch		
mainRectSlotFreeformType	String representing enumeration such that: "0"=dl, "1"=h, "2"=hk, "3"=bl, "4"=hk, "5"=ac, "6"=h, "7"=ch		
mainSurfaceType	String representing enumeration such that: "0"=sk, "1"=co, "2"=j, "3"=s, "4"=sl, "5"=sk, "6"=cs, "7"=s, "8"=ck		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
normalDirection	"%.16g %.16g %.16g"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
pinDirection	"%.16g %.16g %.16g"		
precision	"%d"		
punchDirection	"%.16g %.16g %.16g"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subType	String representing enumeration such that: "0"=mechanicallyfastened, "1"=manufacturingandassembly, "2"=diesmolds, "3"=fixing, "4"=temporarytransferred, "5"=permanentlytransferred, "6"=coordination, "7"=correctable, "8"=auxiliaryrest, "9"=scribedcoordinationline, "10"=accessclearancehole, "11"=detached, "12"=blank, "13"=jclamps		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbol	String representing enumeration such that: "0"=rectangle, "1"=edge, "2"=hole, "3"=freeform, "4"=slot, "5"=surface		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.16 PMI_USER_DEFINED (0x0114)

Table 220 — PMI_USER_DEFINED

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright,	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlin

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		eType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth. *measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlin

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			eColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
measurementType			replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
UserDefinedSymbolText[%d]	NULL		
UserDefinedSymbolText[%d].Text	NULL	UserDefinedSymbolText[%d]	
UserDefinedSymbolText[%d].Text.Item[%d]	NULL	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.Item[%d].Outline	NULL	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side	NULL	UserDefinedSymbolText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	UserDefinedSymbolText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Item[%d].Outline	UserDefinedSymbolText[%d].Text.Item[%d].Outline.colour

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaces UserDefinedSymbolText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.Item[%d].outlineColour for compatiblity.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d].Outline	
UserDefinedSymbolText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Item[%d].Outline	
UserDefinedSymbolText[%d].Text.Item[%d].Outline.filled	"%d"	UserDefinedSymbolText[%d].Text.Item[%d].Outline	
UserDefinedSymbolText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UserDefinedSymbolText[%d].Text.Item[%d].Outline	UserDefinedSymbolText[%d].Text.Item[%d].Outline.style replaces UserDefinedSymbolText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.Item[%d].outlineStyle for compatiblity.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UserDefinedSymbolText[%d].Text.Item[%d].Outline	
UserDefinedSymbolText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright,	UserDefinedSymbolText[%d].Text.Item[%d].Outline	UserDefinedSymbolText[%d].Text.Item[%d].Outline.type replaces UserDefinedSymbolText[%d].Tex

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		t.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.Item[%d].outlineType for compatiblity.
UserDefinedSymbolText[%d].Text.Item[%d].Outline.visible	"%d"	UserDefinedSymbolText[%d].Text.Item[%d].Outline	
UserDefinedSymbolText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UserDefinedSymbolText[%d].Text.Item[%d].Outline.width.name	"%s"	UserDefinedSymbolText[%d].Text.Item[%d].Outline	UserDefinedSymbolText[%d].Text.Item[%d].Outline.width.name replaces UserDefinedSymbolText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.Item[%d].outlineName for compatiblity.
UserDefinedSymbolText[%d].Text.Item[%d].bold	"%d"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].font	"%s"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list,	UserDefinedSymbolText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=datetime, "11"=none		
UserDefinedSymbolText[%d].Text.Item[%d].italic	"%d"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].italicAngle	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].language	"%s"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].lineFactor	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot,	UserDefinedSymbolText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
UserDefinedSymbolText[%d].Text.Item[%d].spaceFactor	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].string	"%s"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity,	UserDefinedSymbolText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UserDefinedSymbolText[%d].Text.Item[%d].textAspect	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].textHeight	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	UserDefinedSymbolText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UserDefinedSymbolText[%d].Text.Item[%d].textLineWidth.name	"%s"	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UserDefinedSymbolText[%d].Text.Item[%d]	
UserDefinedSymbolText[%d].Text.Outline	NULL	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.Outline.Side	NULL	UserDefinedSymbolText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UserDefinedSymbolText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UserDefinedSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UserDefinedSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	UserDefinedSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UserDefinedSymbolText[%d].Text.Outline.Side.width.name	"%s"	UserDefinedSymbolText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UserDefinedSymbolText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Outline	UserDefinedSymbolText[%d].Text.Outline.colour replaces UserDefinedSymbolText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.outlineColour for compatiblity.
UserDefinedSymbolText[%d].Text.Outline.doubleOffset	"%.16g"	UserDefinedSymbolText[%d].Text.Outline	
UserDefinedSymbolText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UserDefinedSymbolText[%d].Text.Outline	
UserDefinedSymbolText[%d].Text.Outline.filled	"%d"	UserDefinedSymbolText[%d].Text.Outline	
UserDefinedSymbolText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed,	UserDefinedSymbolText[%d].Text.Outline	UserDefinedSymbolText[%d].Text.Outline.style replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		UserDefinedSymbolText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.outlineStyle for compatiblity.
UserDefinedSymbolText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UserDefinedSymbolText[%d].Text.Outline	
UserDefinedSymbolText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UserDefinedSymbolText[%d].Text.Outline	UserDefinedSymbolText[%d].Text.Outline.type replaces UserDefinedSymbolText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UserDefinedSymbolText[%d].Text.outlineType for compatiblity.
UserDefinedSymbolText[%d].Text.Outline.visible	"%d"	UserDefinedSymbolText[%d].Text.Outline	
UserDefinedSymbolText[%d].Text.Outline.width.*measurementType*	"%.16g"	UserDefinedSymbolText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UserDefinedSymbolText[%d].Text.Outline.width.name	"%s"	UserDefinedSymbolText[%d].Text.Outline	UserDefinedSymbolText[%d].Text.Outline.width.name replaces UserDefinedSymbolText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of UserDefinedSymbolText[%d].Text.outlineName for compatiblity.
UserDefinedSymbolText[%d].Text.bold	"%d"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.font	"%s"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.italic	"%d"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.italicAngle	"%.16g"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.language	"%s"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.lineFactor	"%.16g"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.maxExtend	"%d"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.name	"%s"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.spaceFactor	"%.16g"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.textAspect	"%.16g"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.textColo	"0x00%02x%02x%02x" such that the values of blue, green	UserDefinedSymbolText[%d].	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ur	and red for the colour are encoded in the string	Text	
UserDefinedSymbolText[%d].Text.textHeight	"%.16g"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.textLineWidth.*measurementType*	"%.16g"	UserDefinedSymbolText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
UserDefinedSymbolText[%d].Text.textLineWidth.name	"%s"	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	UserDefinedSymbolText[%d].Text	
UserDefinedSymbolText[%d].textDirection	"%.16g %.16g %.16g"	UserDefinedSymbolText[%d]	
UserDefinedSymbolText[%d].textOrigin	"%.16g %.16g %.16g"	UserDefinedSymbolText[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
filename	"%s"		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
lineStyle	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
scale	"%.16g"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.17 PMI_CIRCLE_CENTRE_TYPE (0x0238)

Table 221 — PMI_CIRCLE_CENTRE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s; %d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>; <unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle , "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.		the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=thin		
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright,	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset,	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=datetime, "11"=none		
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplemental Symbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth,	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			CoordinatedEntity[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of CoordinatedEntity[%d] or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFacto r	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrou gh	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
_____JtTkIntersectionReference____*reference0*	ORIGIN*reference1*	_____JtTkIntersectionReference____*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkIntersectionReference____*reference1*	ORIGIN*reference0*	_____JtTkIntersectionReference____*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional)

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
axisDirection	"%.16g %.16g %.16g"		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
diameter	"%.16g"		
diameterOverride	"%d"		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
normalDirection	"%.16g %.16g %.16g"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
position	"%.16g %.16g %.16g"		
precision	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.18 PMI_CSYSTEM_TYPE (0x0104)

Table 222 — PMI_PMI_CSYSTEM_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
JT END ARROW TYPE	"CONE" or "NONE"		CONE is currently the only supported arrow type
JTTK MULTICAD COLOR	%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f		
JTTK MULTICAD CUSTOM PROPERTY TYPES	*<lengthOfKey>*:*<key>*:*<lengthOfPropertyType>*:*<JtkString|JtkINT|JtkFLOAT>*:		Repeat pattern in value for each custom key
JTTK MULTICAD CUSTOM WIDTH	%d		Pixel width, draw style width should be set to 4 to honour this variable
JTTK MULTICAD DRAW STYLE	%d,%d first number is the type, second the width		type values are as follows: 0=solid, 1=dashed, 2=dotted, 3=dashed dotted, 4=phantom, 5=long dashed, 6=centerline, 7=invisible

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD LAYER	%d		
JTTK MULTICAD NAME	%s		
JTTK MULTICAD VISIBILITY	"FALSE" or "TRUE"		

N.19 PMI_REF_PT_TYPE (0x0110)

Table 223 — PMI_REF_PT_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

<table>
<tr><th>“Key” Property Atom Value String</th><th>“Value” Property Atom Value String Encoding Format</th><th>Prequist Key</th><th>AdditionalNote</th></tr>
<tr><td></td><td>"46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity</td><td></td><td></td></tr>
<tr><td>GeneralAttribute[%d].ListValue[%d]</td><td>"%s"</td><td>GeneralAttribute[%d]</td><td></td></tr>
<tr><td>JTTK MULTICAD COLOR</td><td>%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f</td><td></td><td></td></tr>
<tr><td>JTTK MULTICAD CUSTOM PROPERTY TYPES</td><td><lengthOfKey>:<key>:<lengthOfPropertyType>:<JtkString|JtkINT|JtkFLOAT>:</td><td></td><td>Repeat pattern in value for each custom key</td></tr>
<tr><td>JTTK MULTICAD CUSTOM WIDTH</td><td>%d</td><td></td><td>Pixel width, draw style width should be set to 4 to honour this variable</td></tr>
<tr><td>JTTK MULTICAD DRAW STYLE</td><td>%d,%d first number is the type, second the width</td><td></td><td>type values are as follows: 0=no marker, 1=point, 2=dot, 3=asterix, 4=circle, 5=poundsign, 6=X, 7=gridpoint, 8=square, 9=triangle,</td></tr>
</table>

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			10=diamond, 11=centerline, 12=anchor, 13=filled circled, 14=filled square, 15=large filled square, 16=circle in circle, 17=circle in square, 18=square in square, 19=filled left triangle, 20=filled right triangle, 21=filled up triangle, 22=filled down triangle, 23=filled left triangle in circle, 24=filled right triangle in circle, 25=filled up triangle in circle, 26=filled down triangle in circle, 27=filled left triangle in square 28=filled right triangle in square, 29=filled up triangle in square, 30=filled down triangle in square, 31=rounded cross, 32=filled diamond, 33=up down triangles, 34=left right triangles, 35=small wheel, 36=large wheel, 37=hollow circle, 38=arc end point, 39=big asterisk, 40=line in cricle, 41=plus in circle, 42=center of rotation, 43=invalid
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD DRAW STYLE	%d,%d first number is the type, second the width		type values are as follows: 0=solid, 1=dashed, 2=dotted, 3=dashed dotted, 4=phantom, 5=long dashed, 6=centerline, 7=invisible

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD LAYER	%d		
JTTK MULTICAD NAME	%s		
JTTK MULTICAD VISIBILITY	"FALSE" or "TRUE"		

N.20 PMI_REFAXIS_TYPE (0x0111)

Table 224 — PMI_REFAXIS_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
JT END ARROW TYPE	"CONE" or "NONE"		CONE is currently the only supported arrow type
JTTK MULTICAD COLOR	%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f		
JTTK MULTICAD CUSTOM PROPERTY TYPES	*<lengthOfKey>*:*<key>*:*<lengthOfPropertyType>*:*<JtkString|JtkINT|JtkFLOAT>*:		Repeat pattern in value for each custom key
JTTK MULTICAD CUSTOM WIDTH	%d		Pixel width, draw style width should be set to 4 to honour this variable
JTTK MULTICAD DRAW STYLE	%d,%d first number is the type, second the width		type values are as follows: 0=solid, 1=dashed, 2=dotted,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			3=dashed dotted, 4=phantom, 5=long dashed, 6=centerline, 7=invisible
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD LAYER	%d		
JTTK MULTICAD NAME	%s		
JTTK MULTICAD VISIBILITY	"FALSE" or "TRUE"		

N.21 PMI_REF_PLANE_TYPE (0x0112)

Table 225 — PMI_REF_PLANE_TYPE

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
JTTK MULTICAD COLOR	%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f		
JTTK MULTICAD CUSTOM PROPERTY TYPES	*<lengthOfKey>*:*<key>*:*<lengthOfPropertyType>*:*<JtkString|JtkINT|JtkFLOAT>*:		Repeat pattern in value for each custom key
JTTK MULTICAD CUSTOM WIDTH	%d		Pixel width, draw style width should be set to 4 to honour this variable
JTTK MULTICAD DRAW STYLE	%d,%d first number is the type, second the width		type values are as follows: 0=solid, 1=dashed, 2=dotted, 3=dashed dotted, 4=phantom, 5=long dashed, 6=centerline,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			7=invisible
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD LAYER	%d		
JTTK MULTICAD NAME	%s		
JTTK MULTICAD VISIBILITY	"FALSE" or "TRUE"		

N.22 PMI_SECTION_TYPE (0x0305)

Table 226 — PMI_SECTION_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
CLIP_CAPPING	"%d"		
CLIP_CAPPING_COLOR	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
CLIP_CAPPING_USE_BODY_COLOR	"True" or "False"		
CLIP_CURVE	"%d"		
CLIP_CURVE_COLOR	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
CLIP_CURVE_USE_BODY_COLOR	"True" or "False"		
CLIP_HATCH_ANGLE	"%.16g"		
CLIP_HATCH_ASSEMBLY_ADJACENT_METHOD	Enumerated string supporting values		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"None", "Diagonal" and "Variable"		
CLIP_HATCH_ASSEMBLY_ADJACENT_THRESHOLD	"%.16g"		
CLIP_HATCH_COLOR	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
CLIP_HATCH_MATERIAL_DEFAULT	"%s"		
CLIP_HATCH_NOMATERIAL_DEFAULT	%s		The string should be the name of the pattern to use when a intersected part has no material with a pattern definition.
CLIP_HATCH_PATTERN	%s		The string should be the name of the pattern to use.
CLIP_HATCH_REF_AXIS	"%.16g %.16g %.16g"		
CLIP_HATCH_ROTATE	"%d"		
CLIP_HATCH_SPACING	"%.16g"		
CLIP_HATCH_SPACING_ANGLE_DIR	%d %d %d %d %d		The spacing, angle and direction that the hatching of the section has
CLIP_HATCH_TYPE	"GLOBAL" or "MATERIAL"		If GLOBAL then the CLIP_HATCH_PATTERN property should also be present. If MATERIAL CLIP_HATCH_NOMATERIAL_DEFAULT should be
CLIP_NORMAL	"%.16g,%.16g,%.16g"		
CLIP_POSITION	"%.16g,%.16g,%.16g"		
CLIP_RENDER	String representing enumeration such that: "0"=shaded, "1"=wireframe		
CLIP_STYLE	Enumerated string supporting values "Both", "Near", "Far" and "None"		
CLIP_WIREFRAME	"True" or "False"		
Curve[%d]	"%d %d %d %s" such that first in is		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.		
namedFromCPS	"%d"		
removeCoincidentEntities	"%d"		

N.23 PMI_REF_CIRCLE_TYPE (0x030A)

Table 227 — PMI_REF_CIRCLE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

<table>
<tr><th>“Key” Property Atom Value String</th><th>“Value” Property Atom Value String Encoding Format</th><th>Prequist Key</th><th>AdditionalNote</th></tr>
<tr><td></td><td>"46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity</td><td></td><td></td></tr>
<tr><td>GeneralAttribute[%d].ListValue[%d]</td><td>"%s"</td><td>GeneralAttribute[%d]</td><td></td></tr>
<tr><td>JTTK MULTICAD COLOR</td><td>%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f</td><td></td><td></td></tr>
<tr><td>JTTK MULTICAD CUSTOM PROPERTY TYPES</td><td><lengthOfKey>:<key>:<lengthOfPropertyType>:<JtkString|JtkINT|JtkFLOAT>:</td><td></td><td>Repeat pattern in value for each custom key</td></tr>
<tr><td>JTTK MULTICAD CUSTOM WIDTH</td><td>%d</td><td></td><td>Pixel width, draw style width should be set to 4 to honour this variable</td></tr>
<tr><td>JTTK MULTICAD DRAW STYLE</td><td>%d,%d first number is the type, second the width</td><td></td><td>type values are as follows: 0=solid, 1=dashed, 2=dotted, 3=dashed dotted, 4=phantom, 5=long dashed, 6=centerline,</td></tr>
</table>

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			7=invisible
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD LAYER	%d		
JTTK MULTICAD NAME	%s		
JTTK MULTICAD VISIBILITY	"FALSE" or "TRUE"		
origin	"%.16g %.16g %.16g"		
radius	"%.16g"		
tParm	"%.16g %.16g"		
xaxis	"%.16g %.16g %.16g"		
zaxis	"%.16g %.16g %.16g"		

N.24 PMI_REF_CYLINDER_TYPE (0x030B)

Table 228 — PMI_REF_CYLINDER_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
JTTK MULTICAD COLOR	%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f		
JTTK MULTICAD CUSTOM PROPERTY TYPES	*<lengthOfKey>*:*<key>*:*<lengthOfPropertyType>*:*<JtkString|JtkINT|JtkFLOAT>*:		Repeat pattern in value for each custom key

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
JTTK MULTICAD CUSTOM WIDTH	%d		Pixel width, draw style width should be set to 4 to honour this variable
JTTK MULTICAD DRAW STYLE	%d,%d first number is the type, second the width		type values are as follows: 0=solid, 1=dashed, 2=dotted, 3=dashed dotted, 4=phantom, 5=long dashed, 6=centerline, 7=invisible
			width values are as follows: 0=normal, 1=thick, 2=thin, 4=custom width
JTTK MULTICAD LAYER	%d		
JTTK MULTICAD NAME	%s		
JTTK MULTICAD VISIBILITY	"FALSE" or "TRUE"		
origin	"%.16g %.16g %.16g"		
radius	"%.16g"		
uvParm	"%.16g %.16g %.16g %.16g"		
xaxis	"%.16g %.16g %.16g"		
zaxis	"%.16g %.16g %.16g"		

N.25 PMI_CALLOUT_DIM_TYPE (0X030D)

Table 229 — PMI_CALLOUT_DIM_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DimensionText[%d]	NULL		DimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
DimensionText[%d].Text	NULL	DimensionText[%d]	
DimensionText[%d].Text.Item[%d]	NULL	DimensionText[%d].Text	
DimensionText[%d].Text.Item[%d].Outline	NULL	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].Outline.Side	NULL	DimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.colour replaces DimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.filled	"%d"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.style replaces DimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse,	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		DimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineType for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.visible	"%d"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.width.name replaces DimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineName for compatiblity.
DimensionText[%d].Text.Item[%d].bold	"%d"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].font	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].italic	"%d"	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].language	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint,	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
DimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].string	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that:	DimensionText[%d].Text.Item[%d	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=sub, "1"=super	]	
DimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textAspec	"%.16g"	DimensionText[%d].Text.Item[%d	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
t		]	
DimensionText[%d].Text.Item[%d].textColou r	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textHeigh t	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textLine Width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].textLine Width.name	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textThick ness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Outline	NULL	DimensionText[%d].Text	
DimensionText[%d].Text.Outline.Side	NULL	DimensionText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.thickn ess	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.width. *measurementType*	"%.16g"	DimensionText[%d].Text.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Outline.Side.width.name	"%s"	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.colour replaces DimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineColour for compatiblity.
DimensionText[%d].Text.Outline.doubleOffset	"%.16g"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.filled	"%d"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.style replaces DimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineStyle for compatiblity.
DimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft,	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.type replaces DimensionText[%d].Text.outlineType due to the increasing complexity of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineType for compatiblity.
DimensionText[%d].Text.Outline.visible	"%d"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Outline.width.name	"%s"	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.width.name replaces DimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineName for compatiblity.
DimensionText[%d].Text.bold	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.font	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.italic	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.italicAngle	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DimensionText[%d].Text	
DimensionText[%d].Text.language	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.lineFactor	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.maxExtend	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.name	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.spaceFactor	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.strikethrough	String representing enumeration such that:	DimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=none, "1"=single, "2"=double		
DimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DimensionText[%d].Text	
DimensionText[%d].Text.textAspect	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text	
DimensionText[%d].Text.textHeight	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	DimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.textLineWidth.name	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	DimensionText[%d].Text	
DimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text	
DimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DimensionText[%d].Text	
DimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	DimensionText[%d]	
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
DualDimensionText[%d]	NULL		DualDimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
DualDimensionText[%d].Text	NULL	DualDimensionText[%d]	
DualDimensionText[%d].Text.Item[%d]	NULL	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.Item[%d].Outline	NULL	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].Outline.Side	NULL	DualDimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.colour replaces DualDimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.filled	"%d"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.style replaces DualDimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft,	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.type replaces DualDimensionText[%d].Text.Item[%d].outlineType due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineType for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.visible	"%d"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.width.name replaces DualDimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineName for compatiblity.
DualDimensionText[%d].Text.Item[%d].bold	"%d"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].font	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].italic	"%d"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].language	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove,	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
DualDimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].string	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].symb	String representing enumeration such that:	DualDimensionText[%d].Text.Ite	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ol	"0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	m[%d]	
DualDimensionText[%d].Text.Item[%d].textAspect	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textC	"0x00%02x%02x%02x" such that the values of	DualDimensionText[%d].Text.Ite	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
olour	blue, green and red for the colour are encoded in the string	m[%d]	
DualDimensionText[%d].Text.Item[%d].text Height	"%.16g"	DualDimensionText[%d].Text.Ite m[%d]	
DualDimensionText[%d].Text.Item[%d].textL ineWidth.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Ite m[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].textL ineWidth.name	"%s"	DualDimensionText[%d].Text.Ite m[%d]	
DualDimensionText[%d].Text.Item[%d].textT hickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Ite m[%d]	
DualDimensionText[%d].Text.Item[%d].unde rline	String representing enumeration such that: "0"=over, "1"=under	DualDimensionText[%d].Text.Ite m[%d]	
DualDimensionText[%d].Text.Outline	NULL	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.Outline.Side	NULL	DualDimensionText[%d].Text.Out line	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.co lour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Out line.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.st yle	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Out line.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.th ickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Out line.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.wi dth.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Out line.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Outline.Side.wi dth.name	"%s"	DualDimensionText[%d].Text.Out line.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of	DualDimensionText[%d].Text.Out	DualDimensionText[%d].Text.Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	blue, green and red for the colour are encoded in the string	line	.colour replaces DualDimensionText[%d].Text.outline Colour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outline Colour for compatiblity.
DualDimensionText[%d].Text.Outline.double Offset	"%.16g"	DualDimensionText[%d].Text.Out line	
DualDimensionText[%d].Text.Outline.fillColo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Out line	
DualDimensionText[%d].Text.Outline.filled	"%d"	DualDimensionText[%d].Text.Out line	
DualDimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Out line	DualDimensionText[%d].Text.Outline .style replaces DualDimensionText[%d].Text.outline Style due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outline Style for compatiblity.
DualDimensionText[%d].Text.Outline.thickn ess	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Out line	
DualDimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright,	DualDimensionText[%d].Text.Out line	DualDimensionText[%d].Text.Outline .type replaces DualDimensionText[%d].Text.outline Type due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		recommend continuing to support read/write of DualDimensionText[%d].Text.outline Type for compatiblity.
DualDimensionText[%d].Text.Outline.visible	"%d"	DualDimensionText[%d].Text.Out line	
DualDimensionText[%d].Text.Outline.width. *measurementType*	"%.16g"	DualDimensionText[%d].Text.Out line	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Outline.width. name	"%s"	DualDimensionText[%d].Text.Out line	DualDimensionText[%d].Text.Outline .width.name replaces DualDimensionText[%d].Text.outline Name due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outline Name for compatiblity.
DualDimensionText[%d].Text.bold	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.font	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.italic	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.italicAngle	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.language	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.lineFactor	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.maxExtend	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.name	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.spaceFactor	"%.16g"	DualDimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textAspect	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textHeight	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textLineWidth. *measurementType*	"%.16g"	DualDimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.textLineWidth. name	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textOrientatio n	String representing enumeration such that: "0"=horizontal, "1"=vertical	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DualDimensionText[%d].Text	
DualDimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	DualDimensionText[%d]	
FeatureIdentification	"%s"		
Format	NULL		*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d]	NULL	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline	NULL	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].
Format.Item[%d].Outline.Side	NULL	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.Side.width.name	"%s"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.colour replaces *Format*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineColour for compatiblity.
Format.Item[%d].Outline.doubleOffset	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.filled	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.style replaces *Format*.Item[%d].outlineStyle due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineStyle for compatiblity.
Format.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.type replaces *Format*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineType for compatiblity.
Format.Item[%d].Outline.visible	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			with meterWidth or pixelWidth
Format.Item[%d].Outline.width.name	"%s"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.width.name replaces *Format*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineName for compatiblity.
Format.Item[%d].bold	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].font	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italic	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italicAngle	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft,	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		TextFormat[%d].
Format.Item[%d].language	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove,	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Format.Item[%d].spaceFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].string	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint,	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"63"=datumtranslation		
Format.Item[%d].textAspect	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textHeight	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].textLineWidth.name	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.bold	"%d"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.font	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italic	"%d"	*Format*	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceTextFormat[%d] or TextFormat[%d].
Format.italicAngle	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.language	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.lineFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.spaceFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textAspect	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textHeight	"%.16g"	*Format*	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceTextFormat[%d] or TextFormat[%d].
Format.textLineWidth.*measurementType*	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.textLineWidth.name	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.underline	String representing enumeration such that: "0"=over, "1"=under	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
FreeState	NULL		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
GeneralToleranceISO	NULL		
GeneralToleranceISO.toleranceClass	String representing enumeration such that: "0"=fine, "1"=medium, "2"=coarse, "3"=verycoarse,	GeneralToleranceISO	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that:	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=normal, "1"=thick, "2"=thin		
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded	Outline	Outline.colour replaces outlineColour due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	in the string		complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
ParameterDimension[%d]	NULL		
ParameterDimension[%d].Description	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].DimensionText[%d]	NULL	ParameterDimension[%d]	DimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
ParameterDimension[%d].DimensionText[%d].Text	NULL	ParameterDimension[%d].DimensionText[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	NULL	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	NULL	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side	NULL	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.colour replaces ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineColou

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			r for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.filled	"%d"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.style replaces ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota,	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.type replaces ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineType for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.visible	"%d"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DimensionText[%d].Text.Item[%d].Outline.width.name replaces ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.Item[%d].outlineName for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].bold	"%d"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].font	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%	"%d"	ParameterDimension[%d].Dimen	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
d].Text.Item[%d].italic		sionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].language	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel,	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].string	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].textAspect	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].textHeight	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].textLineWidth.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].DimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DimensionText[%d].Text.Outline	NULL	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.Outline.Side	NULL	ParameterDimension[%d].DimensionText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Outline.Side.width.*measurementTyp*	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
e			
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DimensionText[%d].Text.Outline.Side.width.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Outline	ParameterDimension[%d].DimensionText[%d].Text.Outline.colour replaces ParameterDimension[%d].DimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.outlineColour for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Outline.doubleOffset	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Outline	
ParameterDimension[%d].DimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text.Outline	
ParameterDimension[%d].DimensionText[%d].Text.Outline.filled	"%d"	ParameterDimension[%d].DimensionText[%d].Text.Outline	
ParameterDimension[%d].DimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DimensionText[%d].Text.Outline	ParameterDimension[%d].DimensionText[%d].Text.Outline.style replaces ParameterDimension[%d].DimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].Dimension

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text[%d].Text.outlineStyle for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DimensionText[%d].Text.Outline	
ParameterDimension[%d].DimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].DimensionText[%d].Text.Outline	ParameterDimension[%d].DimensionText[%d].Text.Outline.type replaces ParameterDimension[%d].DimensionText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.outlineType for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.Outline.visible	"%d"	ParameterDimension[%d].DimensionText[%d].Text.Outline	
ParameterDimension[%d].DimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DimensionText[%d].Text.Outline.width.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text.Outline	ParameterDimension[%d].DimensionText[%d].Text.Outline.width.name replaces ParameterDimension[%d].DimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DimensionText[%d].Text.outlineName for compatiblity.
ParameterDimension[%d].DimensionText[%d].Text.bold	"%d"	ParameterDimension[%d].DimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DimensionText[%d].Text.font	"%s"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.italic	"%d"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.italicAngle	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.language	"%s"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.lineFactor	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.maxExtend	"%d"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.spaceFactor	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.textAspect	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.textHeight	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].DimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DimensionText[%d].Text.textLineWidth.name	"%s"	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].DimensionText[%d].Text	
ParameterDimension[%d].DimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d].DimensionText[%d]	
ParameterDimension[%d].DisplayPlane.origin	"%.16g %.16g %.16g"	ParameterDimension[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
ParameterDimension[%d].DisplayPlane.xaxis	"%.16g %.16g %.16g"	ParameterDimension[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
ParameterDimension[%d].DisplayPlane.zaxis	"%.16g %.16g %.16g"	ParameterDimension[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
ParameterDimension[%d].DualDimensionText[%d]	NULL	ParameterDimension[%d]	DualDimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position !=

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DimensionText[y].position for any combination of x and y
ParameterDimension[%d].DualDimensionText[%d].Text	NULL	ParameterDimension[%d].DualDimensionText[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	NULL	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	NULL	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side	NULL	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	in the string	Outline	.colour replaces ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.filled	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.style replaces ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.type replaces ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineType for compatiblity.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.visible	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].Outline.width.name replaces ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDime

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			nsionText[%d].Text.Item[%d].outline Name for compatiblity.
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].bold	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].font	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].italic	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].language	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd,	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake,	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].string	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal,	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].textAspect	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].textHeight	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].textLineWidth.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].DualDimensionText[%d].Text.Item[%d]	
ParameterDimension[%d].DualDimensionText[%d].Text.Outline	NULL	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionTe	NULL	ParameterDimension[%d].DualDi	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
xt[%d].Text.Outline.Side		mensionText[%d].Text.Outline	Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side.width.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.colour replaces ParameterDimension[%d].DualDimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.outlineColour for compatiblity.
ParameterDimension[%d].DualDimensionTe	"%.16g"	ParameterDimension[%d].DualDi	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
xt[%d].Text.Outline.doubleOffset		mensionText[%d].Text.Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.filled	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.style replaces ParameterDimension[%d].DualDimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.outlineStyle for compatiblity.
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.type replaces ParameterDimension[%d].DualDimensionText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.outlineType for compatiblity.
ParameterDimension[%d].DualDimensionTe	"%d"	ParameterDimension[%d].DualDi	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
xt[%d].Text.Outline.visible		mensionText[%d].Text.Outline	
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DualDimensionText[%d].Text.Outline.width.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text.Outline	ParameterDimension[%d].DualDimensionText[%d].Text.Outline.width.name replaces ParameterDimension[%d].DualDimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].DualDimensionText[%d].Text.outlineName for compatiblity.
ParameterDimension[%d].DualDimensionText[%d].Text.bold	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.font	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.italic	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.italicAngle	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.language	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.lineFactor	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].DualDimensionText[%d].Text.maxExtend	"%d"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.spaceFactor	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.textAspect	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.textHeight	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].DualDimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].DualDimensionText[%d].Text.textLineWidth.name	"%s"	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].DualDimensionText[%d].Text	
ParameterDimension[%d].DualDimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d].DualDimensionText[%d]	
ParameterDimension[%d].FeatureIdentificat	"%s"	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ion			
ParameterDimension[%d].*Format*	NULL	ParameterDimension[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d]	NULL	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].Outline	NULL	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].Outline.Side	NULL	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Format*.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Format*.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Format*.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].*Format*.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Format*.Item[%d].Outline.Side.width.name	"%s"	ParameterDimension[%d].*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Format*.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			ParameterDimension[%d].*Format*.Item[%d].Outline.colour replaces ParameterDimension[%d].*Format*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Format*.Item[%d].outlineColour for compatiblity.
ParameterDimension[%d].*Format*.Item[%d].Outline.doubleOffset	"%.16g"	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].*Format*.Item[%d].Outline.filled	"%d"	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			ParameterDimension[%d].*Format*.Item[%d].Outline.style replaces ParameterDimension[%d].*Format*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Format*.Item[%d].outlineStyle for compatiblity.
ParameterDimension[%d].*Format*.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			ParameterDimension[%d].*Format*.Item[%d].Outline.type replaces ParameterDimension[%d].*Format*.Ite

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			m[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Format*.Item[%d].outlineType for compatiblity.
ParameterDimension[%d].*Format*.Item[%d].Outline.visible	"%d"	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Format*.Item[%d].Outline.width.name	"%s"	ParameterDimension[%d].*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			ParameterDimension[%d].*Format*.Item[%d].Outline.width.name replaces ParameterDimension[%d].*Format*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Format*.Item[%d].outlineName for compatiblity.
ParameterDimension[%d].*Format*.Item[%d].bold	"%d"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].f	"%s"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ont		.Item[%d]	ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].italic	"%d"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].italicAngle	"%.16g"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].language	"%s"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].lineFactor	"%.16g"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd,	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
ParameterDimension[%d].*Format*.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake,	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ParameterDimension[%d].*Format*.Item[%d].spaceFactor	"%.16g"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].string	"%s"	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism,	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ParameterDimension[%d].*Format*.Item[%d]. textAspect	"%.16g"	ParameterDimension[%d].*Format* .Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d]. textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Format* .Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d]. textHeight	"%.16g"	ParameterDimension[%d].*Format* .Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d]. textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].*Format* .Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Format*.Item[%d].	"%s"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textLineWidth.name		.Item[%d]	ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.bold	"%d"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.font	"%s"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.italic	"%d"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.italicAngle	"%.16g"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.language	"%s"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.lineFactor	"%.16g"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.spaceFact	"%.16g"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
or			ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.textAspect	"%.16g"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.textHeight	"%.16g"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Format*.textLineWidth.name	"%s"	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
ParameterDimension[%d].*Format*.underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].FreeState	NULL	ParameterDimension[%d]	
ParameterDimension[%d].GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration,	ParameterDimension[%d]	It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
ParameterDimension[%d].GeneralAttribute[%d].ListValue[%d]	"%s"	ParameterDimension[%d].GeneralAttribute[%d]	
ParameterDimension[%d].GeneralToleranceISO	NULL	ParameterDimension[%d]	
ParameterDimension[%d].GeneralToleranceISO.toleranceClass	String representing enumeration such that: "0"=fine, "1"=medium, "2"=coarse, "3"=verycoarse,	ParameterDimension[%d].GeneralToleranceISO	
ParameterDimension[%d].Keyword[%d]	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].LAYER	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].Leader[%d]	NULL	ParameterDimension[%d]	
ParameterDimension[%d].Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	ParameterDimension[%d].Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
ParameterDimension[%d].Leader[%d].allAro	"%.16g"	ParameterDimension[%d].Leader	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
undSymbolSize		[%d]	
ParameterDimension[%d].Leader[%d].arrow Angle	"%.16g"	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow Colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow Length	"%.16g"	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow LineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow LineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow OutSideLength	"%.16g"	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow Placement	String representing enumeration such that: "0"=out, "1"=in	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow Type	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	ParameterDimension[%d].Leader [%d]	
ParameterDimension[%d].Leader[%d].arrow Width.*measurementType*	"%.16g"	ParameterDimension[%d].Leader [%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].Leader[%d].arrowWidth.name	"%s"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].direction	"%.16g %.16g %.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].dotDiameter	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionJogAngle	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionJogEnd	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionJogOut	"%d"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionJogStart	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionLineExtension	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionLineGap	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].exten	String representing enumeration such that:	ParameterDimension[%d].Leader	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
sionLineType	"0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	[%d]	
ParameterDimension[%d].Leader[%d].extensionWidth.*measurementType*	"%.16g"	ParameterDimension[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].Leader[%d].extensionWidth.name	"%s"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].lineTextGap	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].offsetCentre	"%d"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].perpendicularToStub	"%d"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].perpendicularToTerminator	"%d"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].radiusToCentre	"%d"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].stubLength	"%.16g"	ParameterDimension[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].Leader[%d].suppressed	"%d"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].tParm	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].terminator	"%.16g %.16g %.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].textOverLeaderFactor	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].textOverStubFactor	"%.16g"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Leader[%d].uvParm.U	"%.16g"	ParameterDimension[%d].Leader[%d]	Expect to also have Leader[%d].uvParam.V.
ParameterDimension[%d].Leader[%d].uvParm.V	"%.16g"	ParameterDimension[%d].Leader[%d]	Expect to also have Leader[%d].uvParam.U.
ParameterDimension[%d].Leader[%d].width.*measurementType*	"%.16g"	ParameterDimension[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].Leader[%d].width.name	"%s"	ParameterDimension[%d].Leader[%d]	
ParameterDimension[%d].Outline	NULL	ParameterDimension[%d]	
ParameterDimension[%d].Outline.Side	NULL	ParameterDimension[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].Outline.Side.thick	String representing enumeration such that:	ParameterDimension[%d].Outline	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ness	"0"=normal, "1"=thick, "2"=thin	.Side	Bottom, Left or Right.
ParameterDimension[%d].Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].Outline.Side.width.name	"%s"	ParameterDimension[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Outline	ParameterDimension[%d].Outline.colour replaces ParameterDimension[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].outlineColour for compatiblity.
ParameterDimension[%d].Outline.doubleOffset	"%.16g"	ParameterDimension[%d].Outline	
ParameterDimension[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Outline	
ParameterDimension[%d].Outline.filled	"%d"	ParameterDimension[%d].Outline	
ParameterDimension[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].Outline	ParameterDimension[%d].Outline.style replaces ParameterDimension[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].outlineStyle for compatiblity.
ParameterDimension[%d].Outline.thickness	String representing enumeration such that:	ParameterDimension[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=normal, "1"=thick, "2"=thin		
ParameterDimension[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].Outline	ParameterDimension[%d].Outline.type replaces ParameterDimension[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].outlineType for compatiblity.
ParameterDimension[%d].Outline.visible	"%d"	ParameterDimension[%d].Outline	
ParameterDimension[%d].Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].Outline.width.name	"%s"	ParameterDimension[%d].Outline	ParameterDimension[%d].Outline.width.name replaces ParameterDimension[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].outlineName for compatiblity.
ParameterDimension[%d].PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d]	
ParameterDimension[%d].PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d]	
ParameterDimension[%d].PMITextFlatToScreenOpacity	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].PMITextForegrou	"0x00%02x%02x%02x" such that the values of	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ndColor	blue, green and red for the colour are encoded in the string		
ParameterDimension[%d].PMITextInPlaneOpacity	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].ReferenceText[%d]	NULL	ParameterDimension[%d]	ReferenceText[%d] index expected to range from 0-1 inclusive.
			Each instance of this is expected to have a different position such that ReferenceText[x].position != ReferenceText[y].position for any combination of x and y
ParameterDimension[%d].ReferenceText[%d].Text	NULL	ParameterDimension[%d].ReferenceText[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	NULL	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	NULL	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side	NULL	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d	"%.16g"	ParameterDimension[%d].Refere	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
].Text.Item[%d].Outline.Side.width.*measurementType*		nceText[%d].Text.Item[%d].Outline.Side	Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.colour replaces ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].outlineColour for compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.filled	"%d"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.style replaces ParameterDimension[%d].Reference

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].Reference Text[%d].Text.Item[%d].outlineStyle for compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	ParameterDimension[%d].Reference Text[%d].Text.Item[%d].Outline.type replaces ParameterDimension[%d].Reference Text[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].Reference Text[%d].Text.Item[%d].outlineType for compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.visible	"%d"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline.width.name	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].Outline	ParameterDimension[%d].Reference Text[%d].Text.Item[%d].Outline.width.name replaces ParameterDimension[%d].Reference

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].outlineName for compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].bold	"%d"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].font	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].italic	"%d"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].italicAngle	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].language	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].lineFactor	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush,	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet,	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].spaceFactor	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].string	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry,	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].textAspect	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].textHeight	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].textLineWidth.name	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].ReferenceText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].ReferenceText[%d].Text.Item[%d]	
ParameterDimension[%d].ReferenceText[%d].Text.Outline	NULL	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side	NULL	ParameterDimension[%d].ReferenceText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side.width.name	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Outline	ParameterDimension[%d].ReferenceText[%d].Text.Outline.colour replaces ParameterDimension[%d].ReferenceText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].ReferenceText[%d].Text.outlineColour for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.doubleOffset	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].ReferenceText[%d].Text.Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Outline.filled	"%d"	ParameterDimension[%d].ReferenceText[%d].Text.Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].ReferenceText[%d].Text.Outline	ParameterDimension[%d].ReferenceText[%d].Text.Outline.style replaces ParameterDimension[%d].ReferenceText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].ReferenceText[%d].Text.outlineStyle for compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].ReferenceText[%d].Text.Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].ReferenceText[%d].Text.Outline	ParameterDimension[%d].ReferenceText[%d].Text.Outline.type replaces ParameterDimension[%d].ReferenceText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].ReferenceText[%d].Text.outlineType for compatiblity.
ParameterDimension[%d].ReferenceText[%d	"%d"	ParameterDimension[%d].Refere	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
].Text.Outline.visible		nceText[%d].Text.Outline	
ParameterDimension[%d].ReferenceText[%d].Text.Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].ReferenceText[%d].Text.Outline.width.name	"%s"	ParameterDimension[%d].ReferenceText[%d].Text.Outline	ParameterDimension[%d].ReferenceText[%d].Text.Outline.width.name replaces ParameterDimension[%d].ReferenceText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].ReferenceText[%d].Text.outlineName for compatiblity.
ParameterDimension[%d].ReferenceText[%d].Text.bold	"%d"	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.font	"%s"	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.italic	"%d"	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.italicAngle	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.language	"%s"	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.lineFactor	"%.16g"	ParameterDimension[%d].ReferenceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d	"%d"	ParameterDimension[%d].Refere	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
].Text.maxExtend		nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.name	"%s"	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.spaceFactor	"%.16g"	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.textAspect	"%.16g"	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.textHeight	"%.16g"	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].Refere nceText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].ReferenceText[%d].Text.textLineWidth.name	"%s"	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].Refere nceText[%d].Text	
ParameterDimension[%d].ReferenceText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d].Refere nceText[%d]	Only the before and after values are expected.
ParameterDimension[%d].RevisionModifier	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].SafetyClassificatio n	"%s"	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].Statistical	NULL	ParameterDimension[%d]	
ParameterDimension[%d].*Text*	NULL	ParameterDimension[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d]	NULL	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].Outline	NULL	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].Outline.Side	NULL	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].*Text*.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Text*.Item[%d].Outline.Side.width.name	"%s"	ParameterDimension[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Item[%d].Outline.colour replaces ParameterDimension[%d].*Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ParameterDimension[%d].*Text*.Item[%d].outlineColour for compatiblity.
ParameterDimension[%d].*Text*.Item[%d].Outline.doubleOffset	"%.16g"	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].Outline.filled	"%d"	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Item[%d].Outline.style replaces ParameterDimension[%d].*Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.Item[%d].outlineStyle for compatiblity.
ParameterDimension[%d].*Text*.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].*Text*.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Item[%d].Outline.type replaces ParameterDimension[%d].*Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.Item[%d].outlineType for compatiblity.
ParameterDimension[%d].*Text*.Item[%d].Outline.visible	"%d"	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Text*.Item[%d].Outline.width.name	"%s"	ParameterDimension[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Item[%d].Outline.width.name replaces ParameterDimension[%d].*Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.Item[%d].outlineName for compatiblity.
ParameterDimension[%d].*Text*.Item[%d].bold	"%d"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].font	"%s"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].italic	"%d"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].italicAngle	"%.16g"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].*Text*.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].language	"%s"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].lineFactor	"%.16g"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface,	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].spaceFactor	"%.16g"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].stri	String representing enumeration such that:	ParameterDimension[%d].*Text*.It	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
kethrough	"0"=none, "1"=single, "2"=double	em[%d]	ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].string	"%s"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction,	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ParameterDimension[%d].*Text*.Item[%d].textAspect	"%.16g"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].textHeight	"%.16g"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Text*.Item[%d].textLineWidth.name	"%s"	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].*Text*.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline	NULL	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline.Side	NULL	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Outline.Side.	String representing enumeration such that:	ParameterDimension[%d].*Text*.O	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
thickness	"0"=normal, "1"=thick, "2"=thin	utline.Side	ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Outline.Side.width.*measurementType*	"%.16g"	ParameterDimension[%d].*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Text*.Outline.Side.width.name	"%s"	ParameterDimension[%d].*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
ParameterDimension[%d].*Text*.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Outline.colour replaces ParameterDimension[%d].*Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.outlin

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			eColour for compatiblity.
ParameterDimension[%d].*Text*.Outline.doubleOffset	"%.16g"	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline.filled	"%d"	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Outline.style replaces ParameterDimension[%d].*Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.outlineStyle for compatiblity.
ParameterDimension[%d].*Text*.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline.type	String representing enumeration such that:	ParameterDimension[%d].*Text*.O	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	utline	ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			ParameterDimension[%d].*Text*.Outline.type replaces ParameterDimension[%d].*Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.outlineType for compatiblity.
ParameterDimension[%d].*Text*.Outline.visible	"%d"	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.Outline.width.*measurementType*	"%.16g"	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Text*.Outline.width.name	"%s"	ParameterDimension[%d].*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ParameterDimension[%d].*Text*.Outline.width.name replaces ParameterDimension[%d].*Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ParameterDimension[%d].*Text*.outlineName for compatiblity.
ParameterDimension[%d].*Text*.bold	"%d"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.font	"%s"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.italic	"%d"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.italicAngle	"%.16g"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.language	"%s"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.lineFactor	"%.16g"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.maxExtend	"%d"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.name	"%s"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.spaceFactor	"%.16g"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.textAspect	"%.16g"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.textColour	"0x00%02x%02x%02x" such that the values of	ParameterDimension[%d].*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	blue, green and red for the colour are encoded in the string		ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.textHeight	"%.16g"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].*Text*.textLineWidth.name	"%s"	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].*Text*.underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d].*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ParameterDimension[%d].ValueToCustomer	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].____JtTkIntersecti	ORIGIN*reference1*	ParameterDimension[%d].____Jt	ORIGIN is optional and should only

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
onReference___*reference0*		TkIntersectionReference___*reference1*	be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
ParameterDimension[%d].____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	ParameterDimension[%d].____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
ParameterDimension[%d].____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.	ParameterDimension[%d]	*reference* is the property value encoded also into the key identifying which reference is an origin reference.
ParameterDimension[%d].accountabilityId	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].alignment	String representing enumeration such that:	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=middle, "1"=bottom		
ParameterDimension[%d].allAround	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].allOver	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].angleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees	ParameterDimension[%d]	
ParameterDimension[%d].angleNumerator	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].appendedTextSpaceFactor	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove	ParameterDimension[%d]	
ParameterDimension[%d].basic	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].blanked	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].bold	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].causality	String representing enumeration such that: "0"=key, "1"=functional, "2"=reference, "3"=associated, "4"=pmi	ParameterDimension[%d]	
ParameterDimension[%d].commaAsDecimal	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].deviation	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].diameterPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d]	
ParameterDimension[%d].dimensionLeadingZero	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dimensionLineBetweenArrows	"%d"	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].dimensionLineTrim	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dimensionTrailingZero	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].direction	"%.16g %.16g %.16g"	ParameterDimension[%d]	
ParameterDimension[%d].documentation	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualDimensionLineCenter	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualLowerDeltaDenominator	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d]	
ParameterDimension[%d].dualPrecision	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualTolerancePrecision	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualType	String representing enumeration such that: "0"=positional, "1"=bracket, "2"=lineseparated	ParameterDimension[%d]	
ParameterDimension[%d].dualUnit	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].dualUnitText	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualUpperDeltaDenominator	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].dualValueDenominator	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].featureOfSize	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].fitGrade	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].flag	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].foldLocation	"%.16g %.16g %.16g"	ParameterDimension[%d]	
ParameterDimension[%d].font	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].fraction	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].fractionSize	String representing enumeration such that:	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=full, "1"=twothirds, "2"=threequarters		
ParameterDimension[%d].grade	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].group	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].inspection	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].inspectionDisplay	String representing enumeration such that: "0"=unset, "1"=none, "2"=before, "3"=after, "4"=beforeafter, "5"=all	ParameterDimension[%d]	
ParameterDimension[%d].isReference	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].italic	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].italicAngle	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ParameterDimension[%d]	
ParameterDimension[%d].language	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].limitDisplay	String representing enumeration such that: "0"=none, "1"=limitfit	ParameterDimension[%d]	
ParameterDimension[%d].limitFitOrder	String representing enumeration such that: "0"=valuelimitfittolerance, "1"=tolerancevaluelimitfit, "2"=valuetolerancelimitfit	ParameterDimension[%d]	
ParameterDimension[%d].limitFitParenthesis	String representing enumeration such that: "0"=none, "1"=valuelimitfit, "2"=limitfit, "3"=tolerance, "4"=value, "5"=valuetolerance	ParameterDimension[%d]	
ParameterDimension[%d].lineFactor	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].linearPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d]	
ParameterDimension[%d].lowerDelta	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].lowerDeltaDenominator	"%d"	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].majorAngle	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].manual	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].nameComponents .displayName	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].nameComponents .name	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].narrowLeaderAng le	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].narrowOffset	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].notToScale	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].ordinateBaselineZ ero	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].origin	String representing enumeration such that: "0"=first, "1"=second	ParameterDimension[%d]	
ParameterDimension[%d].originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright	ParameterDimension[%d]	
ParameterDimension[%d].panZoom	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].parameter	String representing enumeration such that: "0"=diameter, "1"=depth, "2"=counterborediameter, "3"=counterboredepth, "4"=countersinkdiameter, "5"=countersinkangle, "6"=taperangle, "7"=fit, "8"=screwtype, "9"=screwsize, "10"=threadsize, "11"=threaddepth, "12"=startchamferdiameter, "13"=startchamferangle, "14"=endchamferdiameter,	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"15"=endchamferangle, "16"=pitch, "17"=angle, "18"=minordiameter, "19"=majordiameter, "20"=tapdrilldiameter, "21"=length, "22"=tapdrillsize, "23"=symbolicthreadcallout, "24"=shaftsize, "25"=threadpitch, "26"=form, "27"=internalexternalsymbol		
ParameterDimension[%d].patternCount	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].pitchDiaDeviation	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].pitchDiaGrade	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].precision	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].projected	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].radialPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d]	
ParameterDimension[%d].referenceContent	String representing enumeration such that: "0"=value, "1"=prefix, "2"=tolerance	ParameterDimension[%d]	
ParameterDimension[%d].referenceDisplay	String representing enumeration such that: "0"=parenthesis, "1"=reference, "2"=matched	ParameterDimension[%d]	
ParameterDimension[%d].singleSideFirst	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].singleSideLength	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].singleSided	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].spaceFactor	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].stackFactor	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009	ParameterDimension[%d]	
ParameterDimension[%d].statisticalPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ParameterDimension[%d]	
ParameterDimension[%d].style	String representing enumeration such that: "0"=lineardiametral, "1"=radial, "2"=controlledradial, "3"=diametral, "4"=limits, "5"=ordinate, "6"=sphericalradial, "7"=sphericaldiametral, "8"=narrow, "9"=none	ParameterDimension[%d]	
ParameterDimension[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ParameterDimension[%d]	
ParameterDimension[%d].symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d]	
ParameterDimension[%d].textAspect	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ParameterDimension[%d]	
ParameterDimension[%d].textDirection	"%.16g %.16g %.16g"	ParameterDimension[%d]	
ParameterDimension[%d].textHeight	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].textLeaderPosition	String representing enumeration such that: "0"=none, "1"=aboveleader, "2"=afterleader, "3"=abovestub, "4"=afterstub	ParameterDimension[%d]	
ParameterDimension[%d].textLineWidth.*measurementType*	"%.16g"	ParameterDimension[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ParameterDimension[%d].textLineWidth.name	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ParameterDimension[%d]	
ParameterDimension[%d].textOrigin	"%.16g %.16g %.16g"	ParameterDimension[%d]	
ParameterDimension[%d].textRotationPoint	"%.16g %.16g %.16g"	ParameterDimension[%d]	
ParameterDimension[%d].textThickness	String representing enumeration such that:	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=normal, "1"=thick, "2"=thin		
ParameterDimension[%d].threadClass	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].toleranceAngleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees	ParameterDimension[%d]	
ParameterDimension[%d].toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral	ParameterDimension[%d]	
ParameterDimension[%d].toleranceLeadingZero	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].tolerancePrecision	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].toleranceTextSpaceFactor	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].toleranceTrailingZero	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].type	String representing enumeration such that: "0"=curvelength, "1"=linear, "2"=angular, "3"=radial	ParameterDimension[%d]	
ParameterDimension[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ParameterDimension[%d]	
ParameterDimension[%d].unit	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].unitText	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].upperDelta	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].upperDeltaDenominator	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].uriRefs[%d]	"%s"	ParameterDimension[%d]	
ParameterDimension[%d].usage	"%s"	ParameterDimension[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ParameterDimension[%d].valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown	ParameterDimension[%d]	
ParameterDimension[%d].value	"%.16g"	ParameterDimension[%d]	
ParameterDimension[%d].valueDenominator	"%d"	ParameterDimension[%d]	
ParameterDimension[%d].zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit	ParameterDimension[%d]	
ParameterDimension[%d].zeroToleranceSign	String representing enumeration such that: "0"=none, "1"=plus, "2"=minus	ParameterDimension[%d]	
ReferenceText[%d]	NULL		ReferenceText[%d] index expected to range from 0-1 inclusive.
			Each instance of this is expected to have a different position such that ReferenceText[x].position != ReferenceText[y].position for any combination of x and y
ReferenceText[%d].Text	NULL	ReferenceText[%d]	
ReferenceText[%d].Text.Item[%d]	NULL	ReferenceText[%d].Text	
ReferenceText[%d].Text.Item[%d].Outline	NULL	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].Outline.Side	NULL	ReferenceText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Si	String representing enumeration such that:	ReferenceText[%d].Text.Item[%d]	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
de.thickness	"0"=normal, "1"=thick, "2"=thin	.Outline.Side	Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.colour replaces ReferenceText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineColour for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.filled	"%d"	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.style replaces ReferenceText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].ou

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			tlineStyle for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.type replaces ReferenceText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineType for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline.visible	"%d"	ReferenceText[%d].Text.Item[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].Outline.width.name	"%s"	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.width.name replaces ReferenceText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineName for compatiblity.
ReferenceText[%d].Text.Item[%d].bold	"%d"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].font	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list,	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=datetime, "11"=none		
ReferenceText[%d].Text.Item[%d].italic	"%d"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].italicAngle	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].language	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineFactor	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint,	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
ReferenceText[%d].Text.Item[%d].spaceFactor	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].string	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].symbol	String representing enumeration such that:	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
ReferenceText[%d].Text.Item[%d].textAspect	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textColou	"0x00%02x%02x%02x" such that the values of	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
r	blue, green and red for the colour are encoded in the string		
ReferenceText[%d].Text.Item[%d].textHeight	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].textLineWidth.name	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Outline	NULL	ReferenceText[%d].Text	
ReferenceText[%d].Text.Outline.Side	NULL	ReferenceText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Outline.Side.width.name	"%s"	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.colo

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	blue, green and red for the colour are encoded in the string		ur replaces ReferenceText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineColour for compatiblity.
ReferenceText[%d].Text.Outline.doubleOffset	"%.16g"	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.filled	"%d"	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.style replaces ReferenceText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineStyle for compatiblity.
ReferenceText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota,	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.type replaces ReferenceText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineType

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		for compatiblity.
ReferenceText[%d].Text.Outline.visible	"%d"	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Outline.width.name	"%s"	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.width.name replaces ReferenceText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outlineName for compatiblity.
ReferenceText[%d].Text.bold	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.font	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.italic	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.italicAngle	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ReferenceText[%d].Text	
ReferenceText[%d].Text.language	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.lineFactor	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.maxExtend	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.name	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.spaceFactor	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ReferenceText[%d].Text	
ReferenceText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	ReferenceText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.textAspect	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text	
ReferenceText[%d].Text.textHeight	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textLineWidth.*measurementType*	"%.16g"	ReferenceText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.textLineWidth.name	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ReferenceText[%d].Text	
ReferenceText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text	
ReferenceText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	ReferenceText[%d].Text	
ReferenceText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ReferenceText[%d]	Only the before and after values are expected.
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Statistical	NULL		
Text	NULL		*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing,	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent,	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance,	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ValueToCustomer	"%s"		

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
alignment	String representing enumeration such that: "0"=middle, "1"=bottom		
allAround	"%d"		
allOver	"%d"		
angleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		
angleNumerator	"%.16g"		
appendedTextSpaceFactor	"%.16g"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
basic	"%d"		
blanked	"%d"		
bold	"%d"		
causality	String representing enumeration such that: "0"=key, "1"=functional, "2"=reference, "3"=associated, "4"=pmi		
commaAsDecimal	"%d"		
deviation	"%s"		
diameterPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dimensionLeadingZero	"%d"		
dimensionLineBetweenArrows	"%d"		
dimensionLineTrim	"%d"		
dimensionTrailingZero	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
direction	"%.16g %.16g %.16g"		
documentation	"%d"		
dualDimensionLineCenter	"%d"		
dualLowerDeltaDenominator	"%d"		
dualPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dualPrecision	"%d"		
dualTolerancePrecision	"%d"		
dualType	String representing enumeration such that: "0"=positional, "1"=bracket, "2"=lineseparated		
dualUnit	"%s"		
dualUnitText	"%d"		
dualUpperDeltaDenominator	"%d"		
dualValueDenominator	"%d"		
featureOfSize	"%d"		
fitGrade	"%d"		
flag	"%d"		
foldLocation	"%.16g %.16g %.16g"		
font	"%s"		
fraction	"%d"		
fractionSize	String representing enumeration such that: "0"=full, "1"=twothirds, "2"=threequarters		
grade	"%d"		
group	"%s"		
inspection	"%d"		
inspectionDisplay	String representing enumeration such that: "0"=unset, "1"=none, "2"=before, "3"=after, "4"=beforeafter, "5"=all		
isReference	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
limitDisplay	String representing enumeration such that: "0"=none, "1"=limitfit		
limitFitOrder	String representing enumeration such that: "0"=valuelimitfittolerance, "1"=tolerancevaluelimitfit, "2"=valuetolerancelimitfit		
limitFitParenthesis	String representing enumeration such that: "0"=none, "1"=valuelimitfit, "2"=limitfit, "3"=tolerance, "4"=value, "5"=valuetolerance		
lineFactor	"%.16g"		
linearPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
lowerDelta	"%.16g"		
lowerDeltaDenominator	"%d"		
majorAngle	"%d"		
manual	"%d"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
narrowLeaderAngle	"%.16g"		
narrowOffset	"%.16g"		
notToScale	"%d"		
ordinateBaselineZero	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
origin	String representing enumeration such that: "0"=first, "1"=second		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
parameterLineFactor	"%.16g"		
parameterSpaceFactor	"%.16g"		
patternCount	"%d"		
pitchDiaDeviation	"%s"		
pitchDiaGrade	"%d"		
precision	"%d"		
projected	"%d"		
radialPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
referenceContent	String representing enumeration such that: "0"=value, "1"=prefix, "2"=tolerance		
referenceDisplay	String representing enumeration such that: "0"=parenthesis, "1"=reference, "2"=matched		
singleSideFirst	"%d"		
singleSideLength	"%.16g"		
singleSided	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=gm_addendum_1994, "8"=asme_y145_2009		
statisticalPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
style	String representing enumeration such that: "0"=leardiametral, "1"=radial, "2"=controlledradial, "3"=diametral, "4"=limits, "5"=ordinate, "6"=sphericalradial, "7"=sphericaldiametral, "8"=narrow, "9"=none		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLeaderPosition	String representing enumeration such that: "0"=none, "1"=aboveleader, "2"=afterleader, "3"=abovestub, "4"=afterstub		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
threadClass	"%d"		
toleranceAngleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		
toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral		
toleranceLeadingZero	"%d"		
tolerancePrecision	"%d"		
toleranceTextSpaceFactor	"%.16g"		
toleranceTrailingZero	"%d"		
type	String representing enumeration such that: "0"=curvelength, "1"=linear, "2"=angular, "3"=radial		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
unitText	"%d"		
upperDelta	"%.16g"		
upperDeltaDenominator	"%d"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
value	"%.16g"		
valueDenominator	"%d"		
zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit		
zeroToleranceSign	String representing enumeration such that: "0"=none, "1"=plus, "2"=minus		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.26 PMI_COORDINATE_NOTE_TYPE (0x0239)

Table 230 — PMI_COORDINATE_NOTE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
CoordinateSystem.origin	"%.16g %.16g %.16g"		Should not be used along side 'coordinateSystemRef' Should also have 'CoordinateSystem.xaxis'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			and 'CoordinateSystem.zaxis'
CoordinateSystem.xaxis	"%.16g %.16g %.16g"		Should not be used along side 'coordinateSystemRef' Should also have 'CoordinateSystem.origin' and 'CoordinateSystem.zaxis'
CoordinateSystem.zaxis	"%.16g %.16g %.16g"		Should not be used along side 'coordinateSystemRef' Should also have 'CoordinateSystem.origin' and 'CoordinateSystem.xaxis'
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted		It is expected that you will

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea,		also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	for the colour are encoded in the string		
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth,	Outline	Outline.type replaces outlineType due to the increasing complexity of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft,	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=bottomright, "8"=bottomcentre		
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side	String representing enumeration such that: "0"=normal, "1"=thick,	UnitSymbol[%d].Item[%d].	*Side* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.thickness	"2"=thin	Outline.Side	one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side .width.name	"%s"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	UnitSymbol[%d].Item[%d].O utline.colour replaces UnitSymbol[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o utlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.dou bleOffset	"%.16g"	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.fillC olour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.fille d	"%d"	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d]. Outline	UnitSymbol[%d].Item[%d].O utline.style replaces UnitSymbol[%d].Item[%d].o utlineStyle due to the

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
category	"%s"		
commaAsDecimal	"%d"		
coordinateSystemRef	"%d"		Reference to an existing CoordinateSystem
			Should not be used along side 'CoordinateSystem.origin', 'CoordinateSystem.xaxis' and 'CoordinateSystem.zaxis'
flag	"%d"		
font	"%s"		
group	"%s"		
iOverride	"%.16g"		
iPrefix	"%s"		
iSuffix	"%s"		
identifier	"%s"		
includeI	"%d"		
includeJ	"%d"		
includeK	"%d"		
includeLabel	"%d"		
includeLevel	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
includeX	"%d"		
includeY	"%d"		
includeZ	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
jOverride	"%.16g"		
jPrefix	"%s"		
jSuffix	"%s"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
kOverride	"%.16g"		
kPrefix	"%s"		
kSuffix	"%s"		
labelPrefix	"%s"		
labelSuffix	"%s"		
language	"%s"		
levelPrefix	"%s"		
levelSuffix	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
precision	"%d"		
revision	"%s"		
spaceFactor	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textBox	"%d"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
xOverride	"%.16g"		
xPrefix	"%s"		
xSuffix	"%s"		
yOverride	"%.16g"		
yPrefix	"%s"		
ySuffix	"%s"		
zOverride	"%.16g"		
zPrefix	"%s"		
zSuffix	"%s"		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.27 PMI_ATTRIBUTE_NOTE_TYPE (0x0240)

Table 231 — PMI_ATTRIBUTE_NOTE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
AttributeName[%d]	"%s"		
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=full		
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminato r	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side .width.name	"%s"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	UnitSymbol[%d].Item[%d].O utline.colour replaces UnitSymbol[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineNa

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			me due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
includeLabels	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMI_PROP_NOTE_HAS_URL	Encoded enumeration such that “0” = False “1” = True		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.28 PMI_BUNDLE_DRESSING_NOTE_TYPE (0x0241)

Table 232 — PMI_BUNDLE_DRESSING_NOTE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
AttributeName[%d]	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
includeLabels	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
locationOnCurve	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
precision	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
type	String representing enumeration such that: "0"=bundlenote, "1"=dressingnote		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.29 PMI_CUTTING_PLANE_SYMBOL_TYPE (0x0242)

Table 233 — PMI_CUTTING_PLANE_SYMBOL_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ConstraintPlane.origin	"%.16g %.16g %.16g"	ConstraintPlane	
ConstraintPlane.xaxis	"%.16g %.16g %.16g"	ConstraintPlane	
ConstraintPlane.zaxis	"%.16g %.16g %.16g"	ConstraintPlane	
ConstraintPlane[%d]	"%.16g %.16g %.16g"	type	Expects type property to be plane
ConstraintPlane[%d].materialSide	"%.16g %.16g %.16g"	ConstraintPlane[%d]	
ConstraintPlane[%d].origin	"%.16g %.16g %.16g"	ConstraintPlane[%d]	
ConstraintPlane[%d].xaxis	"%.16g %.16g %.16g"	ConstraintPlane[%d]	
ConstraintPlane[%d].zaxis	"%.16g %.16g %.16g"	ConstraintPlane[%d]	
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visib	"%d"	TextFormat[%d].Item[%d].	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
le		Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side	String representing enumeration such that: "0"=unset,	UnitSymbol[%d].Item[%d].	*Side* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.style	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side .width.name	"%s"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colo ur	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	UnitSymbol[%d].Item[%d].O utline.colour replaces UnitSymbol[%d].Item[%d].o utlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o utlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.dou bleOffset	"%.16g"	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.fillC olour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.fille d	"%d"	UnitSymbol[%d].Item[%d]. Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset,	UnitSymbol[%d].Item[%d].	UnitSymbol[%d].Item[%d].O

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	utline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyl

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			e for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.typ e replaces UnitSymbol[%d].outlineTyp e due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineTyp e for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measur ementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.wid th.name replaces UnitSymbol[%d].outlineNa me due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineNa me for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			%d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			key identifying which reference is an origin reference.
accountabilityId	"%d"		
aftDistance	"%.16g"	type	Expects type property to be closed
aftThroughAll	"%d"	type	Expects type property to be closed
arrowPlacement1	String representing enumeration such that: "0"=unset, "1"=origin, "2"=top, "3"=bottomright, "4"=topright		
arrowPlacement2	String representing enumeration such that: "0"=unset, "1"=origin, "2"=top, "3"=bottomright, "4"=topright		
arrowTowardsCutter	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
curveRef	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	type	Expects type property to not be plane
flag	"%d"		
font	"%s"		
foreDistance	"%.16g"	type	Expects type property to be closed
foreThroughAll	"%d"	type	Expects type property to be closed

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
lineStyle	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
materialSide	"%d"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
planeColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
shadedPlane	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=unset, "1"=iso, "2"=asme_y1441m		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textPlaneOrientation	String representing enumeration such that: "0"=unset, "1"=parallel, "2"=perpendicular		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
twoArrows	"%d"		
type	String representing enumeration such that: "0"=plane, "1"=open, "2"=closed		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that "0" = False "1" = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.30 PMI_CROSSHATCH_TYPE (0x0243)

Table 234 — PMI_CROSSHATCH_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
CrosshatchPattern.angle	"%.16g"	CrosshatchPattern	
CrosshatchPattern.pattern	String representing enumeration such that: "0"=unset, "1"=rubber, "2"=lead, "3"=user, "4"=steel, "5"=brass, "6"=solid, "7"=iron, "8"=aluminum, "9"=glass, "10"=electricalwinding, "11"=refractory, "12"=thermalinsulation	CrosshatchPattern	
CrosshatchPattern.rotate	"%d"	CrosshatchPattern	
CrosshatchPattern.spacing	"%.16g"	CrosshatchPattern	
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Outline	Outline.style replaces outlineStyle due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side. width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth,	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		utlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=bottomright, "8"=bottomcentre		
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWid th.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWid th.name	"%s"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyl

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			e due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineNa

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			me for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.31 PMI_E_MARKING_TYPE (0x0244)

Table 235 — PMI_E_MARKING_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	an entity.		of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
NoteText[%d]	NULL		
NoteText[%d].Text	NULL	NoteText[%d]	
NoteText[%d].Text.Item[%d]	NULL	NoteText[%d].Text	
NoteText[%d].Text.Item[%d].Outline	NULL	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].Outline.Side	NULL	NoteText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	NoteText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d].Outline.colour replaces NoteText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d].outlineColour for compatiblity.
NoteText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.filled	"%d"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.sty	String representing enumeration such that: "0"=unset,	NoteText[%d].Text.Item[%	NoteText[%d].Text.Item[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
le	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	d].Outline	Outline.style replaces NoteText[%d].Text.Item[%d]. outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d]. outlineStyle for compatiblity.
NoteText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d]. Outline.type replaces NoteText[%d].Text.Item[%d]. outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.Item[%d]. outlineType for compatiblity.
NoteText[%d].Text.Item[%d].Outline.visible	"%d"	NoteText[%d].Text.Item[%d].Outline	
NoteText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].Outline.width.name	"%s"	NoteText[%d].Text.Item[%d].Outline	NoteText[%d].Text.Item[%d]. Outline.width.name replaces NoteText[%d].Text.Item[%d]. outlineName due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of NoteText[%d].Text.Item[%d]. outlineName for compatiblity.
NoteText[%d].Text.Item[%d].bold	"%d"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].font	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].italic	"%d"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].italicAngle	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].language	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineFactor	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	NoteText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].spaceFactor	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].string	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis,	NoteText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
NoteText[%d].Text.Item[%d].textAspect	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textHeight	"%.16g"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	NoteText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Item[%d].textLineWidth.name	"%s"	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	NoteText[%d].Text.Item[%d]	
NoteText[%d].Text.Outline	NULL	NoteText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.Outline.Side	NULL	NoteText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.thickne ss	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.Side.width. *measurementType*	"%.16g"	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Outline.Side.width.n ame	"%s"	NoteText[%d].Text.Outline .Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
NoteText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.c olour replaces NoteText[%d].Text.outlineCo lour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineCo lour for compatiblity.
NoteText[%d].Text.Outline.doubleOffse t	"%.16g"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.filled	"%d"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset,	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.st

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		yle replaces NoteText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineStyle for compatiblity.
NoteText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.type replaces NoteText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of NoteText[%d].Text.outlineType for compatiblity.
NoteText[%d].Text.Outline.visible	"%d"	NoteText[%d].Text.Outline	
NoteText[%d].Text.Outline.width.*measurementType*	"%.16g"	NoteText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.Outline.width.name	"%s"	NoteText[%d].Text.Outline	NoteText[%d].Text.Outline.width.name replaces NoteText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of NoteText[%d].Text.outlineName for compatiblity.
NoteText[%d].Text.bold	"%d"	NoteText[%d].Text	
NoteText[%d].Text.font	"%s"	NoteText[%d].Text	
NoteText[%d].Text.italic	"%d"	NoteText[%d].Text	
NoteText[%d].Text.italicAngle	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	NoteText[%d].Text	
NoteText[%d].Text.language	"%s"	NoteText[%d].Text	
NoteText[%d].Text.lineFactor	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.maxExtend	"%d"	NoteText[%d].Text	
NoteText[%d].Text.name	"%s"	NoteText[%d].Text	
NoteText[%d].Text.spaceFactor	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	NoteText[%d].Text	
NoteText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	NoteText[%d].Text	
NoteText[%d].Text.textAspect	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	NoteText[%d].Text	
NoteText[%d].Text.textHeight	"%.16g"	NoteText[%d].Text	
NoteText[%d].Text.textLineWidth.*measurementType*	"%.16g"	NoteText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
NoteText[%d].Text.textLineWidth.name	"%s"	NoteText[%d].Text	
NoteText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	NoteText[%d].Text	
NoteText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	NoteText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
NoteText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	NoteText[%d].Text	
NoteText[%d].textDirection	"%.16g %.16g %.16g"	NoteText[%d]	
NoteText[%d].textOrigin	"%.16g %.16g %.16g"	NoteText[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Text	NULL		*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width. name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline,	*Text*.Item[%d]	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of Text[%d], AltText[%d] or UnitSymbol[%d].
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.	String representing enumeration such that: "0"=normal, "1"=thick,	TextFormat[%d].Item[%d].	*Side* must be replaced by one

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
thickness	"2"=thin	Outline.Side	of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side. width.name	"%s"	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colou r	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].Ou tline.colour replaces TextFormat[%d].Item[%d].ou tlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].ou tlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doub leOffset	"%.16g"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.fillCol our	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d]. Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline	TextFormat[%d].Item[%d].Ou tline.style replaces TextFormat[%d].Item[%d].ou tlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
category	"%s"		
commaAsDecimal	"%d"		
datumOnLeader	String representing enumeration such that: "0"=none, "1"=solid, "2"=filled		
displayType	String representing enumeration such that: "0"=flatToSurface "1"=flatToScreen		
flag	"%d"		
font	"%s"		
group	"%s"		
identifier	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
revision	"%s"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
type	String representing enumeration such that: "0"=exportcontrol, "1"=productproprietaryinfo, "2"=governmentsecurityinfo		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.32 PMI_REGION_TYPE (0x0246)

Table 236 — PMI_REGION_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Area[%d]	NULL		
Area[%d].ConstraintPlane.origin	"%.16g %.16g %.16g"	Area[%d].ConstraintPlane	
Area[%d].ConstraintPlane.xaxis	"%.16g %.16g %.16g"	Area[%d].ConstraintPlane	
Area[%d].ConstraintPlane.zaxis	"%.16g %.16g %.16g"	Area[%d].ConstraintPlane	
Area[%d].CurveRef[%d]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Area[%d]	
Area[%d].diameter	"%.16g"	Area[%d]	
Area[%d].height	"%.16g"	Area[%d]	
Area[%d].innerDiameter	"%.16g"	Area[%d]	
Area[%d].insidePoint	"%.16g %.16g %.16g"	Area[%d]	
Area[%d].length	"%.16g"	Area[%d]	
Area[%d].originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright	Area[%d]	
Area[%d].type	String representing enumeration such that: "0"=rectangular, "1"=circular, "2"=annular, "3"=cylindrical, "4"=general	Area[%d]	
Area[%d].width	"%.16g"	Area[%d]	
CrosshatchPattern.angle	"%.16g"	CrosshatchPattern	
CrosshatchPattern.pattern	String representing enumeration such that: "0"=unset, "1"=rubber, "2"=lead, "3"=user, "4"=steel, "5"=brass, "6"=solid, "7"=iron, "8"=aluminum, "9"=glass, "10"=electricalwinding, "11"=refractory, "12"=thermalinsulation	CrosshatchPattern	
CrosshatchPattern.rotate	"%d"	CrosshatchPattern	
CrosshatchPattern.spacing	"%.16g"	CrosshatchPattern	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
Face[%d]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.		
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	for the colour are encoded in the string		
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side .width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side	"%s"	UnitSymbol[%d].Item[%d].	*Side* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.width.name		Outline.Side	one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle,	UnitSymbol[%d].Item[%d].	UnitSymbol[%d].Item[%d].O

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	utline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWid th.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColo

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ur due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineTyp

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			e for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
conformsToFaces	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.33 PMI_CENTERLINE_TYPE (0x0306)

Table 237 — PMI_CENTERLINE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot,	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart		
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
Position[%d]	"%.16g %.16g %.16g"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.	String representing enumeration such that: "0"=normal, "1"=thick,	TextFormat[%d].Item[%d].	*Side* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
thickness	"2"=thin	Outline.Side	one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethroug h	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle,	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColo

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ur due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineTyp

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			e for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
angle	"%.16g"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
center	"%.16g %.16g %.16g"		
centercrossSize	"%.16g"		
centerlineType	String representing enumeration such that: "0"=centerline2d, "1"=centerline3d, "2"=centermark, "3"=circularcenterline, "4"=boltcirclecenterline, "5"=symmetriccenterline, "6"=offsetcenterline		
circularCenterlineMethod	String representing enumeration such that: "0"=center, "1"=points		
commaAsDecimal	"%d"		
displayOption	String representing enumeration such that: "0"=linenone, "1"=line, "2"=linewithextension, "3"=point		
flag	"%d"		
font	"%s"		
fullCircle	"%d"		
group	"%s"		
individualExtensions	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
overrun	"%.16g"		
panZoom	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
shortDash	"%.16g"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
width.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
width.name	"%s"		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.34 PMI_FIT_DESIGNATION_TYPE (0x0307)

Table 238 — PMI_FIT_DESIGNATION_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
FitDesignationText[%d]	NULL		FitDesignationText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that FitDesignationText[x].position != FitDesignationText[y].position for any combination of x and y
FitDesignationText[%d].Text	NULL	FitDesignationText[%d]	
FitDesignationText[%d].Text.Item[%d]	NULL	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.Item[%d].Outline	NULL	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].Outline.Side	NULL	FitDesignationText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FitDesignationText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FitDesignationText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FitDesignationText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FitDesignationText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FitDesignationText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	FitDesignationText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Item[%d].Outline	FitDesignationText[%d].Text.Item[%d].Outline.colour replaces FitDesignationText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.Item[%d].outlineColour for compatiblity.
FitDesignationText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	FitDesignationText[%d].Text.Item[%d].Outline	
FitDesignationText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Item[%d].Outline	
FitDesignationText[%d].Text.Item[%d].Outline.filled	"%d"	FitDesignationText[%d].Text.Item[%d].Outline	
FitDesignationText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FitDesignationText[%d].Text.Item[%d].Outline	FitDesignationText[%d].Text.Item[%d].Outline.style replaces FitDesignationText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FitDesignationText[%d].Text.Item[%d].outlineStyle for compatiblity.
FitDesignationText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FitDesignationText[%d].Text.Item[%d].Outline	
FitDesignationText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FitDesignationText[%d].Text.Item[%d].Outline	FitDesignationText[%d].Text.Item[%d].Outline.type replaces FitDesignationText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.Item[%d].outlineType for compatiblity.
FitDesignationText[%d].Text.Item[%d].Outline.visible	"%d"	FitDesignationText[%d].Text.Item[%d].Outline	
FitDesignationText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	FitDesignationText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FitDesignationText[%d].Text.Item[%d].Outline.width.name	"%s"	FitDesignationText[%d].Text.Item[%d].Outline	FitDesignationText[%d].Text.Item[%d].Outline.width.name replaces FitDesignationText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.Item[%d].outlineName for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
FitDesignationText[%d].Text.Item[%d].bold	"%d"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].font	"%s"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].italic	"%d"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].italicAngle	"%.16g"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].language	"%s"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].lineFactor	"%.16g"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel,	FitDesignationText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FitDesignationText[%d].Text.Item[%d].spaceFactor	"%.16g"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].string	"%s"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc,	FitDesignationText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FitDesignationText[%d].Text.Item[%d].textAspect	"%.16g"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].textHeight	"%.16g"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	FitDesignationText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FitDesignationText[%d].Text.Item[%d].textLineWidth.name	"%s"	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FitDesignationText[%d].Text.Item[%d]	
FitDesignationText[%d].Text.Outline	NULL	FitDesignationText[%d].Tex	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		t	
FitDesignationText[%d].Text.Outline.Side	NULL	FitDesignationText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FitDesignationText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FitDesignationText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	FitDesignationText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FitDesignationText[%d].Text.Outline.Side.width.name	"%s"	FitDesignationText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FitDesignationText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Outline	FitDesignationText[%d].Text.Outline.colour replaces FitDesignationText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.outlineColour for compatiblity.
FitDesignationText[%d].Text.Outline.doubleOffset	"%.16g"	FitDesignationText[%d].Text.Outline	
FitDesignationText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text.Outline	
FitDesignationText[%d].Text.Outline.filled	"%d"	FitDesignationText[%d].Tex	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		t.Outline	
FitDesignationText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FitDesignationText[%d].Text.Outline	FitDesignationText[%d].Text.Outline.style replaces FitDesignationText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.outlineStyle for compatiblity.
FitDesignationText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FitDesignationText[%d].Text.Outline	
FitDesignationText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FitDesignationText[%d].Text.Outline	FitDesignationText[%d].Text.Outline.type replaces FitDesignationText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.outlineType for compatiblity.
FitDesignationText[%d].Text.Outline.visible	"%d"	FitDesignationText[%d].Text.Outline	
FitDesignationText[%d].Text.Outline.width.*measurementType*	"%.16g"	FitDesignationText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FitDesignationText[%d].Text.Outline.width.name	"%s"	FitDesignationText[%d].Text.Outline	FitDesignationText[%d].Text.Outline.width.name replaces FitDesignationText[%d].Text.outlineName due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of FitDesignationText[%d].Text.outlineName for compatiblity.
FitDesignationText[%d].Text.bold	"%d"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.font	"%s"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.italic	"%d"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.italicAngle	"%.16g"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.language	"%s"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.lineFactor	"%.16g"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.maxExtend	"%d"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.name	"%s"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.spaceFactor	"%.16g"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.textAspect	"%.16g"	FitDesignationText[%d].Tex	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		t	
FitDesignationText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.textHeight	"%.16g"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.textLineWidth.*measurementType*	"%.16g"	FitDesignationText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
FitDesignationText[%d].Text.textLineWidth.name	"%s"	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FitDesignationText[%d].Text	
FitDesignationText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	FitDesignationText[%d].Text	
FitDesignationText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	FitDesignationText[%d]	
Format	NULL		*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d]	NULL	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline	NULL	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.Side	NULL	*Format*.Item[%d].Outline	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.Side.width.nam	"%s"	*Format*.Item[%d].Outline.Si	*Format* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
e		de	one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.colour replaces *Format*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineColour for compatiblity.
Format.Item[%d].Outline.doubleOffset	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.filled	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.style	String representing enumeration such that: "0"=unset,	*Format*.Item[%d].Outline	*Format* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.style replaces *Format*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineStyle for compatiblity.
Format.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.type replaces *Format*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Format.Item[%d].outlineType for compatiblity.
Format.Item[%d].Outline.visible	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.width.name	"%s"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.width.name replaces *Format*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineName for compatiblity.
Format.Item[%d].bold	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].font	"%s"	*Format*.Item[%d]	*Format* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italic	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italicAngle	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].language	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer,	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
Format.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].spaceFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*.Item[%d]	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].string	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d].textAspect	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textHeight	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].textLineWidth.name	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.bold	"%d"	*Format*	*Format* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.font	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italic	"%d"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italicAngle	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.language	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.lineFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.spaceFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.strikethrough	String representing enumeration such that: "0"=none,	*Format*	*Format* must be replaced by

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=single, "2"=double		one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textAspect	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textHeight	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textLineWidth.*measurementType*	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.textLineWidth.name	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*	*Format* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ToleranceTextFormat[%d] or TextFormat[%d].
Format.underline	String representing enumeration such that: "0"=over, "1"=under	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
FreeState	NULL		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
SafetyClassification	"%s"		
Statistical	NULL		
Text	NULL		*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			support read/write of *Text*.Item[%d].outlineStyle for compatiblity.
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic,	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.
Text.Outline.thickness	String representing enumeration such that: "0"=normal,	*Text*.Outline	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=thick, "2"=thin		ValueText or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.bold	"%d"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ValueText or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of ValueText or UnitSymbol[%d].
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
alignment	String representing enumeration such that: "0"=middle, "1"=bottom		
allAround	"%d"		
allOver	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
diameterPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dimensionLineBetweenArrows	"%d"		
dimensionLineSplit	String representing enumeration such that: "0"=none, "1"=limitfit, "2"=valuelimitfit		
dimensionLineTrim	"%d"		
direction	"%.16g %.16g %.16g"		
dislayClearance	"%d"		
displayTolerance	"%d"		
featureOfSize	"%d"		
flag	"%d"		
foldLocation	"%.16g %.16g %.16g"		
font	"%s"		
group	"%s"		
holeDeviation	"%s"		
holeGrade	"%d"		
holeLowerDetla	"%.16g"		
holeUpperDelta	"%.16g"		
horizontalLine	"%d"		
inspection	"%d"		
inspectionDisplay	String representing enumeration such that: "0"=unset, "1"=none, "2"=before, "3"=after, "4"=beforeafter, "5"=all		
isReference	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
limitDisplay	String representing enumeration such that: "0"=none,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=limitfit		
limitFitAlignment	String representing enumeration such that: "0"=middle, "1"=bottom		
limitFitOrder	String representing enumeration such that: "0"=valuelimitfittolerance, "1"=tolerancevaluelimitfit, "2"=valuetolerancelimitfit		
limitFitParenthesis	String representing enumeration such that: "0"=none, "1"=valuelimitfit, "2"=limitfit, "3"=tolerance, "4"=value, "5"=valuetolerance		
lineFactor	"%.16g"		
linearPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
manual	"%d"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
narrowLeaderAngle	"%.16g"		
narrowOffset	"%.16g"		
notToScale	"%d"		
origin	String representing enumeration such that: "0"=first, "1"=second		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
patternCount	"%d"		
pitchDiaDeviation	"%s"		
pitchDiaGrade	"%d"		
pitchDiameterLowerDelta	"%.16g"		
pitchDiameterUpperDelta	"%.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
precision	"%d"		
radialPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
referenceDisplay	String representing enumeration such that: "0"=parenthesis, "1"=reference, "2"=matched		
shaftDeviation	"%s"		
shaftGrade	"%d"		
shaftLowerDelta	"%.16g"		
shaftUpperDelta	"%.16g"		
singleLine	"%d"		
singleSideFirst	"%d"		
singleSideLength	"%.16g"		
singleSided	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
style	String representing enumeration such that: "0"=lineardiametral, "1"=radial, "2"=controlledradial, "3"=diametral, "4"=limits, "5"=ordinate, "6"=sphericalradial, "7"=sphericaldiametral, "8"=narrow, "9"=none		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLeaderPosition	String representing enumeration such that: "0"=none, "1"=aboveleader, "2"=afterleader, "3"=abovestub, "4"=afterstub		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral		
toleranceLeadingZero	"%d"		
tolerancePrecision	"%d"		
toleranceTrailingZero	"%d"		
type	String representing enumeration such that: "0"=curvelength, "1"=linear, "2"=angular, "3"=radial		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
unitText	"%d"		
uriRefs[%d]	"%s"		
usage	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
value	"%.16g"		
valueLeadingZero	"%d"		
valueTrailingZero	"%d"		
zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.35 PMI_COMPOSITE_FCF_TYPE (0x0308)

Table 239 — PMI_COMPOSITE_FCF_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d]	NULL		
FCF[%d].Description	"%s"	FCF[%d]	
FCF[%d].DisplayPlane.origin	"%.16g %.16g %.16g"	FCF[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
FCF[%d].DisplayPlane.xaxis	"%.16g %.16g %.16g"	FCF[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
FCF[%d].DisplayPlane.zaxis	"%.16g %.16g %.16g"	FCF[%d]	This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FCF[%d].FCFText[%d]	NULL	FCF[%d]	FCFText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that FCFText[x].position != FCFText[y].position for any combination of x and y
FCF[%d].FCFText[%d].Text	NULL	FCF[%d].FCFText[%d]	
FCF[%d].FCFText[%d].Text.Item[%d]	NULL	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.Item[%d].Outline	NULL	FCF[%d].FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side	NULL	FCF[%d].FCFText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].FCFText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text.Item[%d].Outline	FCF[%d].FCFText[%d].Text.Item[%d].Outline.colour replaces FCF[%d].FCFText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.Item[%d].outlineColour for compatiblity.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d].Outline	
FCF[%d].FCFText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	FCF[%d].FCFText[%d].Text.Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		
FCF[%d].FCFText[%d].Text.Item[%d].Outline.filled	"%d"	FCF[%d].FCFText[%d].Text.Item[%d].Outline	
FCF[%d].FCFText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].FCFText[%d].Text.Item[%d].Outline	FCF[%d].FCFText[%d].Text.Item[%d].Outline.style replaces FCF[%d].FCFText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.Item[%d].outlineStyle for compatiblity.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].FCFText[%d].Text.Item[%d].Outline	
FCF[%d].FCFText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].FCFText[%d].Text.Item[%d].Outline	FCF[%d].FCFText[%d].Text.Item[%d].Outline.type replaces FCF[%d].FCFText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.Item[%d].outlineType for compatiblity.
FCF[%d].FCFText[%d].Text.Item[%d].Outline.visible	"%d"	FCF[%d].FCFText[%d].Text.Item[%d].Outline	
FCF[%d].FCFText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].FCFText[%d].Text.Item[%d].Outline.width.name	"%s"	FCF[%d].FCFText[%d].Text.Item[%d].Outline	FCF[%d].FCFText[%d].Text.Item[%d].Outline.width.name replaces FCF[%d].FCFText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.Item[%d].outlineName for compatiblity.
FCF[%d].FCFText[%d].Text.Item[%d].bold	"%d"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].font	"%s"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].italic	"%d"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].italicAngle	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].language	"%s"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].lineFactor	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud,	FCF[%d].FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
FCF[%d].FCFText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove,	FCF[%d].FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].FCFText[%d].Text.Item[%d].spaceFactor	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].string	"%s"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi,	FCF[%d].FCFText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].FCFText[%d].Text.Item[%d].textAspect	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].textHeight	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].FCFText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].FCFText[%d].Text.Item[%d].textLineWidth.name	"%s"	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].FCFText[%d].Text.Item[%d]	
FCF[%d].FCFText[%d].Text.Outline	NULL	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.Outline.Side	NULL	FCF[%d].FCFText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Outline.Side.width.	"%.16g"	FCF[%d].FCFText[%d].Text.Outline.Si	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
measurementType		de	Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].FCFText[%d].Text.Outline.Side.width.name	"%s"	FCF[%d].FCFText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].FCFText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text.Outline	FCF[%d].FCFText[%d].Text.Outline.colour replaces FCF[%d].FCFText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.outlineColour for compatiblity.
FCF[%d].FCFText[%d].Text.Outline.doubleOffset	"%.16g"	FCF[%d].FCFText[%d].Text.Outline	
FCF[%d].FCFText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text.Outline	
FCF[%d].FCFText[%d].Text.Outline.filled	"%d"	FCF[%d].FCFText[%d].Text.Outline	
FCF[%d].FCFText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].FCFText[%d].Text.Outline	FCF[%d].FCFText[%d].Text.Outline.style replaces FCF[%d].FCFText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.outlineStyle for compatiblity.
FCF[%d].FCFText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].FCFText[%d].Text.Outline	
FCF[%d].FCFText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle,	FCF[%d].FCFText[%d].Text.Outline	FCF[%d].FCFText[%d].Text.Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		FCF[%d].FCFText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.outlineType for compatiblity.
FCF[%d].FCFText[%d].Text.Outline.visible	"%d"	FCF[%d].FCFText[%d].Text.Outline	
FCF[%d].FCFText[%d].Text.Outline.width.*measurementType*	"%.16g"	FCF[%d].FCFText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].FCFText[%d].Text.Outline.width.name	"%s"	FCF[%d].FCFText[%d].Text.Outline	FCF[%d].FCFText[%d].Text.Outline.width.name replaces FCF[%d].FCFText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].FCFText[%d].Text.outlineName for compatiblity.
FCF[%d].FCFText[%d].Text.bold	"%d"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.font	"%s"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.italic	"%d"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.italicAngle	"%.16g"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].FCFText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].FCFText[%d].Text.language	"%s"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.lineFactor	"%.16g"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.maxExtend	"%d"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.name	"%s"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.spaceFactor	"%.16g"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.textAspect	"%.16g"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.textHeight	"%.16g"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.textLineWidth.*measurementType*	"%.16g"	FCF[%d].FCFText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].FCFText[%d].Text.textLineWidth.name	"%s"	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].FCFText[%d].Text	
FCF[%d].FCFText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	FCF[%d].FCFText[%d]	
FCF[%d].FeatureIdentification	"%s"	FCF[%d]	
FCF[%d].GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value	FCF[%d]	It is expected that you will also have an additional property: key = <name> and

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration,		value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitma ss, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant , "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
FCF[%d].GeneralAttribute[%d].ListValue[%d]	"%s"	FCF[%d].GeneralAttribute[%d]	
FCF[%d].*Group*	NULL	FCF[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d]	NULL	FCF[%d].*Group*	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].FreeState	NULL	FCF[%d].*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel	NULL	FCF[%d].*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	NULL	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline	NULL	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side	NULL	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
].Outline.Side.width.name		.Item[%d].Outline.Side	Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.It em[%d].Outline.colour replaces FCF[%d].*Group*.Datum[%d].TextLabel.It em[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.It em[%d].outlineColour for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.filled	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.It em[%d].Outline.style replaces FCF[%d].*Group*.Datum[%d].TextLabel.It em[%d].outlineStyle due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].outlineStyle for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.type replaces FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].outlineType for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.visible	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.width.name	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].Outline.width.name replaces FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].outlineName for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].bold	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].font	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].italic	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].italicAngle	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright,	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].language	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].lineFactor	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove,	FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
].spaceFactor		.Item[%d]	Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].string	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle,	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].textAspect	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].textHeight	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].textLineWidth.name	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].*Group*.Datum[%d].TextLabel .Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.Datum[%d].TextLabel.Outline	NULL	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side	NULL	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side.width.name	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.Outline.colour replaces FCF[%d].*Group*.Datum[%d].TextLabel.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.outlineColour for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.doubleOffset	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.filled	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.Outline.style replaces FCF[%d].*Group*.Datum[%d].TextLabel.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of FCF[%d].*Group*.Datum[%d].TextLabel.outlineStyle for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.Outline.type replaces FCF[%d].*Group*.Datum[%d].TextLabel.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.outlineType for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.visible	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.width.*measurementType*	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.Datum[%d].TextLabel.Outline.width.name	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.Datum[%d].TextLabel.Outline.width.name replaces FCF[%d].*Group*.Datum[%d].TextLabel.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.Datum[%d].TextLabel.outlineName for compatiblity.
FCF[%d].*Group*.Datum[%d].TextLabel.bold	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.font	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.italic	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.italicAngle	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.language	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.lineFactor	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.maxExte	"%d"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
nd			Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.name	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.spaceFactor	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.textAspect	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.textHeight	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.textLineWidth.*measurementType*	"%.16g"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.Datum[%d].TextLabel.textLineWidth.name	"%s"	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].TextLabel.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].*Group*.Datum[%d].TextLabel	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].datumRef	"%s"	FCF[%d].*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].label	"%s"	FCF[%d].*Group*.Datum[%d]	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	FCF[%d].*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.Datum[%d].projected	"%d"	FCF[%d].*Group*.Datum[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText	NULL	FCF[%d].*Group*	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d]	NULL	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline	NULL	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side	NULL	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.ExtendedText.Item[%d].Outline.colour replaces FCF[%d].*Group*.ExtendedText.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.Item[%d].outlineColour for compatiblity.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.filled	"%d"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed,	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
			FCF[%d].*Group*.ExtendedText.Item[%d].Outline.style replaces FCF[%d].*Group*.ExtendedText.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.Item[%d].outlineStyle for compatiblity.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.ExtendedText.Item[%d].Outline.type replaces FCF[%d].*Group*.ExtendedText.Item[%d].outlineType due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.Item[%d].outlineType for compatiblity.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.visible	"%d"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.ExtendedText.Item[%d].Outline.width.name	"%s"	FCF[%d].*Group*.ExtendedText.Item[%d].Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.ExtendedText.Item[%d].Outline.width.name replaces FCF[%d].*Group*.ExtendedText.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.Item[%d].outlineName for compatiblity.
FCF[%d].*Group*.ExtendedText.Item[%d].bold	"%d"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].font	"%s"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.ExtendedText.Item[%d].italic	"%d"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].italicAngle	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].language	"%s"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].lineFactor	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].lineW	String representing enumeration such	FCF[%d].*Group*.ExtendedText.Item[%	*Group* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
eldSymbol	that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode,	d]	Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].*Group*.ExtendedText.Item[%d].spaceFactor	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].string	"%s"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity,	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].*Group*.ExtendedText.Item[%d].textAspect	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].textHeight	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			with meterWidth or pixelWidth
FCF[%d].*Group*.ExtendedText.Item[%d].textLineWidth.name	"%s"	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].*Group*.ExtendedText.Item[%d]	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline	NULL	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline.Side	NULL	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.ExtendedText.Outline.Side.width.name	"%s"	FCF[%d].*Group*.ExtendedText.Outline.Side	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Group*.ExtendedText.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.ExtendedText.Outline.colour replaces FCF[%d].*Group*.ExtendedText.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.outlineColour for compatiblity.
FCF[%d].*Group*.ExtendedText.Outline.doubleOffset	"%.16g"	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline.filled	"%d"	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCF[%d].*Group*.ExtendedText.Outline.style replaces FCF[%d].*Group*.ExtendedText.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.outlineStyle for compatiblity.
FCF[%d].*Group*.ExtendedText.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.ExtendedText.Outline.type replaces FCF[%d].*Group*.ExtendedText.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.ExtendedText.Outline.visible	"%d"	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.Outline.width.*measurementType*	"%.16g"	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.ExtendedText.Outline.width.name	"%s"	FCF[%d].*Group*.ExtendedText.Outline	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			FCF[%d].*Group*.ExtendedText.Outline.width.name replaces FCF[%d].*Group*.ExtendedText.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Group*.ExtendedText.outlineName for compatiblity.
FCF[%d].*Group*.ExtendedText.bold	"%d"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.font	"%s"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.italic	"%d"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.italicAngle	"%.16g"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.language	"%s"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Group*.ExtendedText.lineFactor	"%.16g"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.maxExtend	"%d"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.name	"%s"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.spaceFactor	"%.16g"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.textAspect	"%.16g"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.textHeight	"%.16g"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.textLineWidth.*measurementType*	"%.16g"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Group*.ExtendedText.textLineWidth.name	"%s"	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.
FCF[%d].*Group*.ExtendedText.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].*Group*.ExtendedText	*Group* must be replaced by one of Primary, Secondary or Tertiary.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].Keyword[%d]	"%s"	FCF[%d]	
FCF[%d].LAYER	"%d"	FCF[%d]	
FCF[%d].Leader[%d]	NULL	FCF[%d]	
FCF[%d].Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	FCF[%d].Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
FCF[%d].Leader[%d].allAroundSymbolSize	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowAngle	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowLength	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowOutSideLength	"%.16g"	FCF[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].arrowWidth.*measurementType*	"%.16g"	FCF[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].Leader[%d].arrowWidth.name	"%s"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].direction	"%.16g %.16g %.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].dotDiameter	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionJogAngle	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionJogEnd	"%.16g"	FCF[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].Leader[%d].extensionJogOut	"%d"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionJogStart	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionLineExtension	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionLineGap	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].extensionWidth.*measurementType*	"%.16g"	FCF[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].Leader[%d].extensionWidth.name	"%s"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].lineTextGap	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].offsetCentre	"%d"	FCF[%d].Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].Leader[%d].perpendicularToStub	"%d"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].perpendicularToTerminator	"%d"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].radiusToCentre	"%d"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].stubLength	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].suppressed	"%d"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].tParm	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].terminator	"%.16g %.16g %.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].textOverLeaderFactor	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].textOverStubFactor	"%.16g"	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].Leader[%d]	
FCF[%d].Leader[%d].uvParm.U	"%.16g"	FCF[%d].Leader[%d]	Expect to also have Leader[%d].uvParam.V.
FCF[%d].Leader[%d].uvParm.V	"%.16g"	FCF[%d].Leader[%d]	Expect to also have Leader[%d].uvParam.U.
FCF[%d].Leader[%d].width.*measurementType*	"%.16g"	FCF[%d].Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].Leader[%d].width.name	"%s"	FCF[%d].Leader[%d]	
FCF[%d].Outline	NULL	FCF[%d]	
FCF[%d].Outline.Side	NULL	FCF[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].Outline.Side.style	String representing enumeration such	FCF[%d].Outline.Side	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		Bottom, Left or Right.
FCF[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].Outline.Side.width.name	"%s"	FCF[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].Outline	FCF[%d].Outline.colour replaces FCF[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].outlineColour for compatiblity.
FCF[%d].Outline.doubleOffset	"%.16g"	FCF[%d].Outline	
FCF[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].Outline	
FCF[%d].Outline.filled	"%d"	FCF[%d].Outline	
FCF[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].Outline	FCF[%d].Outline.style replaces FCF[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].outlineStyle for compatiblity.
FCF[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].Outline	FCF[%d].Outline.type replaces FCF[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].outlineType for compatiblity.
FCF[%d].Outline.visible	"%d"	FCF[%d].Outline	
FCF[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].Outline.width.name	"%s"	FCF[%d].Outline	FCF[%d].Outline.width.name replaces FCF[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].outlineName for compatiblity.
FCF[%d].PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d]	
FCF[%d].PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d]	
FCF[%d].PMITextFlatToScreenOpacity	"%s"	FCF[%d]	replaces PMITextBackgroundOpacity
FCF[%d].PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the	FCF[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		
FCF[%d].PMITextInPlaneOpacity	"%s"	FCF[%d]	
FCF[%d].RevisionModifier	"%s"	FCF[%d]	
FCF[%d].SafetyClassification	"%s"	FCF[%d]	
FCF[%d].*Text*	NULL	FCF[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d]	NULL	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline	NULL	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline.Side	NULL	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Text*.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Text*.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			FCF[%d].*Text*.Item[%d].Outline.colour replaces FCF[%d].*Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Text*.Item[%d].outlineColour for compatiblity.
FCF[%d].*Text*.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline.filled	"%d"	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed,	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		
			FCF[%d].*Text*.Item[%d].Outline.style replaces FCF[%d].*Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Text*.Item[%d].outlineStyle for compatiblity.
FCF[%d].*Text*.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			FCF[%d].*Text*.Item[%d].Outline.type replaces FCF[%d].*Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCF[%d].*Text*.Item[%d].outlineType for compatiblity.
FCF[%d].*Text*.Item[%d].Outline.visible	"%d"	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Text*.Item[%d].Outline.width.name	"%s"	FCF[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			FCF[%d].*Text*.Item[%d].Outline.width.name replaces FCF[%d].*Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Text*.Item[%d].outlineName for compatiblity.
FCF[%d].*Text*.Item[%d].bold	"%d"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].font	"%s"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].italic	"%d"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].italicAngle	"%.16g"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Text*.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].language	"%s"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].lineFactor	"%.16g"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface,	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].*Text*.Item[%d].spaceFactor	"%.16g"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].string	"%s"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane,	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].*Text*.Item[%d].textAspect	"%.16g"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].textHeight	"%.16g"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Text*.Item[%d].textLineWidth.name	"%s"	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].textThickness	String representing enumeration such	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	that: "0"=normal, "1"=thick, "2"=thin		UnitSymbol[%d].
FCF[%d].*Text*.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].*Text*.Item[%d]	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline	NULL	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline.Side	NULL	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Text*.Outline.Side.width.name	"%s"	FCF[%d].*Text*.Outline.Side	*Text* must be replaced by one of Text or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].*Text*.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			FCF[%d].*Text*.Outline.colour replaces FCF[%d].*Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Text*.outlineColour for compatiblity.
FCF[%d].*Text*.Outline.doubleOffset	"%.16g"	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline.filled	"%d"	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			FCF[%d].*Text*.Outline.style replaces FCF[%d].*Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCF[%d].*Text*.outlineStyle for compatiblity.
FCF[%d].*Text*.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			FCF[%d].*Text*.Outline.type replaces FCF[%d].*Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Text*.outlineType for compatiblity.
FCF[%d].*Text*.Outline.visible	"%d"	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.Outline.width.*measurementType*	"%.16g"	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Text*.Outline.width.name	"%s"	FCF[%d].*Text*.Outline	*Text* must be replaced by one of Text or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			FCF[%d].*Text*.Outline.width.name replaces FCF[%d].*Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].*Text*.outlineName for compatiblity.
FCF[%d].*Text*.bold	"%d"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.font	"%s"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.italic	"%d"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.italicAngle	"%.16g"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.language	"%s"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.lineFactor	"%.16g"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.maxExtend	"%d"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.name	"%s"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.spaceFactor	"%.16g"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].*Text*.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.textAspect	"%.16g"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.textHeight	"%.16g"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.textLineWidth.*measurementType*	"%.16g"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].*Text*.textLineWidth.name	"%s"	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].*Text*.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].*Text*	*Text* must be replaced by one of Text or UnitSymbol[%d].
FCF[%d].TextFormat[%d]	NULL	FCF[%d]	
FCF[%d].TextFormat[%d].Item[%d]	NULL	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].Item[%d].Outline	NULL	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].Outline.Side	NULL	FCF[%d].TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	FCF[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		
FCF[%d].TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	FCF[%d].TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].TextFormat[%d].Item[%d].Outline	FCF[%d].TextFormat[%d].Item[%d].Outline.colour replaces FCF[%d].TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].TextFormat[%d].Item[%d].outlineColour for compatiblity.
FCF[%d].TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].TextFormat[%d].Item[%d].Outline	
FCF[%d].TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].TextFormat[%d].Item[%d].Outline	
FCF[%d].TextFormat[%d].Item[%d].Outline.filled	"%d"	FCF[%d].TextFormat[%d].Item[%d].Outline	
FCF[%d].TextFormat[%d].Item[%d].Outline.sty	String representing enumeration such	FCF[%d].TextFormat[%d].Item[%d].O	FCF[%d].TextFormat[%d].Item[%d].Outli

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
le	that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	utline	ne.style replaces FCF[%d].TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].TextFormat[%d].Item[%d].outlineStyle for compatiblity.
FCF[%d].TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].TextFormat[%d].Item[%d].Outline	
FCF[%d].TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].TextFormat[%d].Item[%d].Outline	FCF[%d].TextFormat[%d].Item[%d].Outline.type replaces FCF[%d].TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].TextFormat[%d].Item[%d].outlineType for compatiblity.
FCF[%d].TextFormat[%d].Item[%d].Outline.visible	"%d"	FCF[%d].TextFormat[%d].Item[%d].Outline	
FCF[%d].TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].TextFormat[%d].Item[%d].Outline.width.name	"%s"	FCF[%d].TextFormat[%d].Item[%d].Outline	FCF[%d].TextFormat[%d].Item[%d].Outline.width.name replaces FCF[%d].TextFormat[%d].Item[%d].outlineName due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].TextFormat[%d].Item[%d].outlineName for compatiblity.
FCF[%d].TextFormat[%d].Item[%d].bold	"%d"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].font	"%s"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].italic	"%d"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].italicAngle	"%.16g"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].language	"%s"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].lineFactor	"%.16g"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd,	FCF[%d].TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
FCF[%d].TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip,	FCF[%d].TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].TextFormat[%d].Item[%d].spaceFactor	"%.16g"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].string	"%s"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis,	FCF[%d].TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].TextFormat[%d].Item[%d].textAspect	"%.16g"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the	FCF[%d].TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	values of blue, green and red for the colour are encoded in the string		
FCF[%d].TextFormat[%d].Item[%d].textHeight	"%.16g"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].TextFormat[%d].Item[%d].textLineWidth.name	"%s"	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].TextFormat[%d].Item[%d]	
FCF[%d].TextFormat[%d].bold	"%d"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].font	"%s"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].italic	"%d"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].italicAngle	"%.16g"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].language	"%s"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].lineFactor	"%.16g"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].spaceFactor	"%.16g"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].textAspect	"%.16g"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	FCF[%d].TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		
FCF[%d].TextFormat[%d].textHeight	"%.16g"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].TextFormat[%d].textLineWidth.name	"%s"	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].TextFormat[%d]	
FCF[%d].TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].TextFormat[%d]	
FCF[%d].ToleranceCompartment[%d]	NULL	FCF[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumG*	String representing enumeration such	FCF[%d].ToleranceCompartment[%d]	*Side* must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
roup.ExtendedText.Item[%d].Outline.Side.thickness	that: "0"=normal, "1"=thick, "2"=thin	.*DatumGroup*.ExtendedText.Item[%d].Outline.Side	Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.style replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.type replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].bold	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].font	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].italic	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].language	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth,	]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].string	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumG*	"%.16g"	FCF[%d].ToleranceCompartment[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
roup.ExtendedText.Item[%d].textHeight		.*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Item[%d].textLineWidth.*m easurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Item[%d].textLineWidth.n ame	"%s"	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Outline	NULL	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Outline. Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumG roup*.ExtendedText.Outline.Side.width.*measur*	"%.16g"	FCF[%d].ToleranceCompartment[%d] .*DatumGroup*.ExtendedText.Outline.	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ementType		Side	
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.style replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineStyle due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.type replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineName due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.bold	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.font	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.italic	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.language	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.maxExtend	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.ExtendedText	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].FreeState	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumG*	NULL	FCF[%d].ToleranceCompartment[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
roup.Reference[%d].TextLabel.Item[%d].Outline		.*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineColour due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.style replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outli	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLab	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.It

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ne.type	"3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	el.Item[%d].Outline	em[%d].Outline.type replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].bold	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLab	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		el.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].font	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].italic	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].language	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].string	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.style replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond,	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.type replaces FCF[%d].ToleranceCompartment[%d].*D*

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		*atumGroup*.Reference[%d].TextLabel.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.bold	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumG*	"%s"	FCF[%d].ToleranceCompartment[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
roup.Reference[%d].TextLabel.font		.*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.italic	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.language	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.maxExtend	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].TextLabel	
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].datumRef	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	semantically tie a singlar semantic aspect with an entity.		
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].label	"%s"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d].projected	"%d"	FCF[%d].ToleranceCompartment[%d].*DatumGroup*.Reference[%d]	*DatumGroup* must be replaced with PrimaryDatum, SecondaryDatum or TertiaryDatum.
FCF[%d].ToleranceCompartment[%d].FreeState	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].*Indicator*	NULL	FCF[%d].ToleranceCompartment[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].FreeState	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].T	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.Outline.Side		extLabel.Item[%d].Outline	
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.style replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLa

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			bel.Item[%d].Outline.type replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].bold	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].font	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].italic	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].language	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicato*	"%.16g"	FCF[%d].ToleranceCompartment[%d]	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
r.DatumGroup.Datum[%d].TextLabel.Item[%d].lineFactor		.*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].T	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.strikethrough		extLabel.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].string	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicato*	"0x00%02x%02x%02x" such that the	FCF[%d].ToleranceCompartment[%d]	*Indicator* must be replaced by one of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
r.DatumGroup.Datum[%d].TextLabel.Outline.fillColour	values of blue, green and red for the colour are encoded in the string	.*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.style replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.type replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*In*

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			dicator.DatumGroup.Datum[%d].TextLabel.Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.bold	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.font	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.italic	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.language	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.maxExtend	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].TextLabel	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].datumRef	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].label	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d].projected	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.Datum[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outlin	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedTex	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
e.Side		t.Item[%d].Outline	
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*In*

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			dicator.DatumGroup.ExtendedText.Item [%d].Outline.style replaces FCF[%d].ToleranceCompartment[%d].*In dicator*.DatumGroup.ExtendedText.Item [%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*In dicator*.DatumGroup.ExtendedText.Item [%d].outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*In dicator*.DatumGroup.ExtendedText.Item [%d].Outline.type replaces FCF[%d].ToleranceCompartment[%d].*In*

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			dicator.DatumGroup.ExtendedText.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].outlineName for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].bold	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].font	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].italic	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].language	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].string	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicato*	String representing enumeration such	FCF[%d].ToleranceCompartment[%d]	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
r.DatumGroup.ExtendedText.Item[%d].subscript	that: "0"=sub, "1"=super	.*Indicator*.DatumGroup.ExtendedText.Item[%d]	BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d]	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Item[%d].underli	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedTex	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ne		t.Item[%d]	
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.Side	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.style replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset,	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"20"=scoredrectangle, "21"=parallelogram		
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.type replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineName due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.bold	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.font	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.italic	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.language	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.maxExtend	"%d"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedTex	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
		t	
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicato*	String representing enumeration such	FCF[%d].ToleranceCompartment[%d]	*Indicator* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
r.DatumGroup.ExtendedText.textOrientation	that: "0"=horizontal, "1"=vertical	.*Indicator*.DatumGroup.ExtendedText	BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*Indicator*.DatumGroup.ExtendedText	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.symbol	String representing enumeration such that: "0"=parallel, "1"=perpendicular, "2"=angular, "3"=including	FCF[%d].ToleranceCompartment[%d].*Indicator*	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].*Indicator*.type	String representing enumeration such that: "0"=intersectionplane, "1"=orientationplane, "2"=collectionplane, "3"=directionfeature	FCF[%d].ToleranceCompartment[%d].*Indicator*	*Indicator* must be replaced by one of BeforeIndicator or AfterIndicator.
FCF[%d].ToleranceCompartment[%d].Statistical	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].*Text*	NULL	FCF[%d].ToleranceCompartment[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	NULL	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	NULL	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Ite	"0x00%02x%02x%02x" such that the	FCF[%d].ToleranceCompartment[%d]	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
m[%d].Outline.colour	values of blue, green and red for the colour are encoded in the string	.*Text*.Item[%d].Outline	TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.style replaces FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineStyle due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.type replaces FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineType for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].bold	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].font	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans,	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].italic	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].language	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete,	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam,	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].string	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength,	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*Text*.Item[%d]	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline	NULL	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side	NULL	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side.width.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Outline.Side	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			Side must be replaced by one of Top, Bottom, Left or Right.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	colour are encoded in the string		TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Outline.colour replaces FCF[%d].ToleranceCompartment[%d].*Text*.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.outlineColour for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.doubleOffset	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.filled	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Outline.style replaces FCF[%d].ToleranceCompartment[%d].*Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			FCF[%d].ToleranceCompartment[%d].*Text*.outlineStyle for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Outline.type replaces FCF[%d].ToleranceCompartment[%d].*Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.outlineType for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.visible	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.Out	"%.16g"	FCF[%d].ToleranceCompartment[%d]	*Text* must be replaced by one of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
line.width.*measurementType*		.*Text*.Outline	TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Text*.Outline.width.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*.Outline	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			FCF[%d].ToleranceCompartment[%d].*Text*.Outline.width.name replaces FCF[%d].ToleranceCompartment[%d].*Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of FCF[%d].ToleranceCompartment[%d].*Text*.outlineName for compatiblity.
FCF[%d].ToleranceCompartment[%d].*Text*.bold	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.font	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.italic	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.italicAngle	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright,	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
FCF[%d].ToleranceCompartment[%d].*Text*.language	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.lineFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.maxExtend	"%d"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.spaceFactor	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.textAspect	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.textHeight	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.textLineWidth.*measurementType*	"%.16g"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
			measurementType must be replaced with meterWidth or pixelWidth
FCF[%d].ToleranceCompartment[%d].*Text*.textLineWidth.name	"%s"	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].*Text*.underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d].ToleranceCompartment[%d].*Text*	*Text* must be replaced by one of TextUnitBasisValue, TextPrefix or TextSuffix.
FCF[%d].ToleranceCompartment[%d].commonZone	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].maxBonus	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].maxBonusValue	"%.16g"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].maxBonusValueString	"%s"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].modifier	String representing enumeration such that: "0"=lmc, "1"=mmc, "2"=rfs	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].precision	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].prefix	"%s"	FCF[%d].ToleranceCompartment[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].ToleranceCompartment[%d].projected	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].projectedValue	"%s"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].projectionVector	"%.16g %.16g %.16g"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].suffix	"%s"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].tangentPlane	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].unequal	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].unequalDisplay	String representing enumeration such that: "0"=circleu, "1"=uz	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].unequalValue	"%s"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].unitBasis	"%d"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].unitBasisValue	"%s"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].value	"%s"	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ToleranceCompartment[%d].zoneShape	String representing enumeration such that: "0"=diameter, "1"=sphericaldiameter, "2"=square	FCF[%d].ToleranceCompartment[%d]	
FCF[%d].ValueToCustomer	"%s"	FCF[%d]	
FCF[%d]._____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	FCF[%d]._____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
FCF[%d].____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	FCF[%d].____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
FCF[%d].____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.	FCF[%d]	*reference* is the property value encoded also into the key identifying which reference is an origin reference.
FCF[%d].accountabilityId	"%d"	FCF[%d]	
FCF[%d].allAround	"%d"	FCF[%d]	
FCF[%d].allOver	"%d"	FCF[%d]	
FCF[%d].angleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees	FCF[%d]	
FCF[%d].attachmentType	String representing enumeration such that: "0"=angulardimension,	FCF[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
FCF[%d].blanked	"%d"	FCF[%d]	
FCF[%d].bold	"%d"	FCF[%d]	
FCF[%d].characteristic	String representing enumeration such that: "0"=profileofaline, "1"=circularrunout, "2"=perpendicularity, "3"=position, "4"=totalrunout, "5"=profileofasurface, "6"=cylindricity, "7"=symmetry, "8"=angularity, "9"=parallelism, "10"=concentricity, "11"=flatness, "12"=circularity, "13"=straightness, "14"=axisintersection	FCF[%d]	
FCF[%d].commaAsDecimal	"%d"	FCF[%d]	
FCF[%d].datumOnLeader	String representing enumeration such that: "0"=none, "1"=solid, "2"=filled	FCF[%d]	
FCF[%d].direction	"%.16g %.16g %.16g"	FCF[%d]	
FCF[%d].fcfTextUnderline	String representing enumeration such that: "0"=none, "1"=top, "2"=all	FCF[%d]	
FCF[%d].flag	"%d"	FCF[%d]	
FCF[%d].font	"%s"	FCF[%d]	
FCF[%d].group	"%s"	FCF[%d]	
FCF[%d].italic	"%d"	FCF[%d]	
FCF[%d].italicAngle	"%.16g"	FCF[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	FCF[%d]	
FCF[%d].language	"%s"	FCF[%d]	
FCF[%d].leadingZero	"%d"	FCF[%d]	
FCF[%d].lineFactor	"%.16g"	FCF[%d]	
FCF[%d].maxBonus	"%d"	FCF[%d]	
FCF[%d].maxBonusPrecision	"%d"	FCF[%d]	
FCF[%d].maxBonusValue	"%.16g"	FCF[%d]	
FCF[%d].nameComponents.displayName	"%s"	FCF[%d]	
FCF[%d].nameComponents.name	"%s"	FCF[%d]	
FCF[%d].originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright	FCF[%d]	
FCF[%d].panZoom	"%d"	FCF[%d]	
FCF[%d].profileType	String representing enumeration such that: "0"=bilateral, "1"=unilateralinside, "2"=unilateraloutside, "3"=bilateralunequal	FCF[%d]	
FCF[%d].profileValue	"%.16g"	FCF[%d]	
FCF[%d].profileValue2	"%.16g"	FCF[%d]	
FCF[%d].projectedBelow	"%d"	FCF[%d]	
FCF[%d].projectedPrecision	"%d"	FCF[%d]	
FCF[%d].spaceFactor	"%.16g"	FCF[%d]	
FCF[%d].stackFactor	"%.16g"	FCF[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009	FCF[%d]	
FCF[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	FCF[%d]	
FCF[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	FCF[%d]	
FCF[%d].symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d]	
FCF[%d].textAspect	"%.16g"	FCF[%d]	
FCF[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	FCF[%d]	
FCF[%d].textDirection	"%.16g %.16g %.16g"	FCF[%d]	
FCF[%d].textHeight	"%.16g"	FCF[%d]	
FCF[%d].textLineWidth.*measurementType*	"%.16g"	FCF[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
FCF[%d].textLineWidth.name	"%s"	FCF[%d]	
FCF[%d].textOrigin	"%.16g %.16g %.16g"	FCF[%d]	
FCF[%d].textRotationPoint	"%.16g %.16g %.16g"	FCF[%d]	
FCF[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	FCF[%d]	
FCF[%d].trailingZero	"%d"	FCF[%d]	
FCF[%d].underline	String representing enumeration such that: "0"=over, "1"=under	FCF[%d]	
FCF[%d].unit	"%s"	FCF[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
FCF[%d].uriRefs[%d]	"%s"	FCF[%d]	
FCF[%d].usage	"%s"	FCF[%d]	
FCF[%d].valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown	FCF[%d]	
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.36 PMI_CHAMFER_TYPE (0X030F)

Table 240 — PMI_CHAMFER_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DimensionText[%d]	NULL		DimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
DimensionText[%d].Text	NULL	DimensionText[%d]	
DimensionText[%d].Text.Item[%d]	NULL	DimensionText[%d].Text	
DimensionText[%d].Text.Item[%d].Outline	NULL	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].Outline.Side	NULL	DimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].Outlin	"%s"	DimensionText[%d].Text.Ite	*Side* must be replaced by one of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
e.Side.width.name		m[%d].Outline.Side	Top, Bottom, Left or Right.
DimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.colour replaces DimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.filled	"%d"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.style replaces DimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline,	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.type replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		DimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineType for compatiblity.
DimensionText[%d].Text.Item[%d].Outline.visible	"%d"	DimensionText[%d].Text.Item[%d].Outline	
DimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	DimensionText[%d].Text.Item[%d].Outline	DimensionText[%d].Text.Item[%d].Outline.width.name replaces DimensionText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.Item[%d].outlineName for compatiblity.
DimensionText[%d].Text.Item[%d].bold	"%d"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].font	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Item[%d].italic	"%d"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].language	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev,	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
DimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].string	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal,	DimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
DimensionText[%d].Text.Item[%d].textAspect	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textHeight	"%.16g"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	DimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Item[%d].textLineWidth.name	"%s"	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	DimensionText[%d].Text.Item[%d]	
DimensionText[%d].Text.Outline	NULL	DimensionText[%d].Text	
DimensionText[%d].Text.Outline.Side	NULL	DimensionText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Outline.Side.width.name	"%s"	DimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.colour replaces DimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineColour for compatiblity.
DimensionText[%d].Text.Outline.doubleOffset	"%.16g"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.filled	"%d"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.style replaces DimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of DimensionText[%d].Text.outline Style for compatiblity.
DimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.type replaces DimensionText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineType for compatiblity.
DimensionText[%d].Text.Outline.visible	"%d"	DimensionText[%d].Text.Outline	
DimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	DimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.Outline.width.name	"%s"	DimensionText[%d].Text.Outline	DimensionText[%d].Text.Outline.width.name replaces DimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DimensionText[%d].Text.outlineName for compatiblity.
DimensionText[%d].Text.bold	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.font	"%s"	DimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DimensionText[%d].Text.italic	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.italicAngle	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DimensionText[%d].Text	
DimensionText[%d].Text.language	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.lineFactor	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.maxExtend	"%d"	DimensionText[%d].Text	
DimensionText[%d].Text.name	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.spaceFactor	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DimensionText[%d].Text	
DimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DimensionText[%d].Text	
DimensionText[%d].Text.textAspect	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DimensionText[%d].Text	
DimensionText[%d].Text.textHeight	"%.16g"	DimensionText[%d].Text	
DimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	DimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DimensionText[%d].Text.textLineWidth.name	"%s"	DimensionText[%d].Text	
DimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	DimensionText[%d].Text	
DimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DimensionText[%d].Text	
DimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DimensionText[%d].Text	
DimensionText[%d].position	String representing enumeration such that: "0"=before,	DimensionText[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=above, "2"=below, "3"=after		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
DualDimensionText[%d]	NULL		DualDimensionText[%d] index expected to range from 0-3 inclusive.
			Each instance of this is expected to have a different position such that DimensionText[x].position != DimensionText[y].position for any combination of x and y
DualDimensionText[%d].Text	NULL	DualDimensionText[%d]	
DualDimensionText[%d].Text.Item[%d]	NULL	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.Item[%d].Outline	NULL	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].Outline.Side	NULL	DualDimensionText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].O	"0x00%02x%02x%02x" such that the values of blue, green	DualDimensionText[%d].Tex	*Side* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
utline.Side.colour	and red for the colour are encoded in the string	t.Item[%d].Outline.Side	Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	DualDimensionText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.colour replaces DualDimensionText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineColour for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.doubleOffset	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.filled	"%d"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed,	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.style replaces

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"4"=phantom, "5"=dashed, "6"=solid, "7"=centreline		DualDimensionText[%d].Text.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineStyle for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.type replaces DualDimensionText[%d].Text.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineType for compatiblity.
DualDimensionText[%d].Text.Item[%d].Outline.visible	"%d"	DualDimensionText[%d].Text.Item[%d].Outline	
DualDimensionText[%d].Text.Item[%d].Outline.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Item[%d].Outline.width.name	"%s"	DualDimensionText[%d].Text.Item[%d].Outline	DualDimensionText[%d].Text.Item[%d].Outline.width.name replaces DualDimensionText[%d].Text.Item[%d].outlineName due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.Item[%d].outlineName for compatiblity.
DualDimensionText[%d].Text.Item[%d].bold	"%d"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].font	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].italic	"%d"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].italicAngle	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].language	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineFactor	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer,	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
DualDimensionText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].spaceFactor	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].string	"%s"	DualDimensionText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textAspect	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textHeight	"%.16g"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			pixelWidth
DualDimensionText[%d].Text.Item[%d].textLineWidth.name	"%s"	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	DualDimensionText[%d].Text.Item[%d]	
DualDimensionText[%d].Text.Outline	NULL	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.Outline.Side	NULL	DualDimensionText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Outline.Side.width.name	"%s"	DualDimensionText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
DualDimensionText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.colour replaces DualDimensionText[%d].Text.outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of DualDimensionText[%d].Text.outlineColour for compatiblity.
DualDimensionText[%d].Text.Outline.doubleOffset	"%.16g"	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.filled	"%d"	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.style replaces DualDimensionText[%d].Text.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineStyle for compatiblity.
DualDimensionText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text.Outline	
DualDimensionText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.type replaces DualDimensionText[%d].Text.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineType for compatiblity.
DualDimensionText[%d].Text.Outline.visib	"%d"	DualDimensionText[%d].Tex	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
le		t.Outline	
DualDimensionText[%d].Text.Outline.width.*measurementType*	"%.16g"	DualDimensionText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.Outline.width.name	"%s"	DualDimensionText[%d].Text.Outline	DualDimensionText[%d].Text.Outline.width.name replaces DualDimensionText[%d].Text.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of DualDimensionText[%d].Text.outlineName for compatiblity.
DualDimensionText[%d].Text.bold	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.font	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.italic	"%d"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.italicAngle	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.language	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.lineFactor	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.maxExtend	"%d"	DualDimensionText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DualDimensionText[%d].Text.name	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.spaceFactor	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textAspect	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textHeight	"%.16g"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textLineWidth.*measurementType*	"%.16g"	DualDimensionText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
DualDimensionText[%d].Text.textLineWidth.name	"%s"	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	DualDimensionText[%d].Text	
DualDimensionText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	DualDimensionText[%d].Text	
DualDimensionText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	DualDimensionText[%d]	
FeatureIdentification	"%s"		
Format	NULL		*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.Item[%d]	NULL	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline	NULL	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.Side	NULL	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline.Si de	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline.Si de	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline.Si de	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.Side.width.*meas urementType*	"%.16g"	*Format*.Item[%d].Outline.Si de	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.Side.width.name	"%s"	*Format*.Item[%d].Outline.Side	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Format.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.colour replaces *Format*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineColour for compatiblity.
Format.Item[%d].Outline.doubleOffset	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.filled	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Format.Item[%d].Outline.style replaces *Format*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineStyle for compatiblity.
Format.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.type replaces *Format*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineType for compatiblity.
Format.Item[%d].Outline.visible	"%d"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].
Format.Item[%d].Outline.width.*measurementType*	"%.16g"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].Outline.width.name	"%s"	*Format*.Item[%d].Outline	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			Format.Item[%d].Outline.width.name replaces *Format*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Format*.Item[%d].outlineName for compatiblity.
Format.Item[%d].bold	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].font	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].italic	"%d"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].
Format.Item[%d].italicAngle	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].language	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot,	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
Format.Item[%d].spaceFactor	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].string	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout,	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
Format.Item[%d].textAspect	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textHeight	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.Item[%d].textLineWidth.name	"%s"	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			TextFormat[%d].
Format.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Format*.Item[%d]	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.bold	"%d"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.font	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italic	"%d"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.italicAngle	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.language	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.lineFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.spaceFactor	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Format.strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textAspect	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textHeight	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textLineWidth.*measurementType*	"%.16g"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Format.textLineWidth.name	"%s"	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
Format.underline	String representing enumeration such that: "0"=over, "1"=under	*Format*	*Format* must be replaced by one of ToleranceTextFormat[%d] or TextFormat[%d].
FreeState	NULL		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the		It is expected that you will also

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance,		have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
GeneralToleranceISO	NULL		
GeneralToleranceISO.toleranceClass	String representing enumeration such that: "0"=fine, "1"=medium, "2"=coarse, "3"=verycoarse,	GeneralToleranceISO	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright,	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
ReferenceText[%d]	NULL		ReferenceText[%d] index expected to range from 0-1 inclusive.
			Each instance of this is expected to have a different position such that ReferenceText[x].position != ReferenceText[y].position for any combination of x and y

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text	NULL	ReferenceText[%d]	
ReferenceText[%d].Text.Item[%d]	NULL	ReferenceText[%d].Text	
ReferenceText[%d].Text.Item[%d].Outline	NULL	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].Outline.Side	NULL	ReferenceText[%d].Text.Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].Outline.Side.width.name	"%s"	ReferenceText[%d].Text.Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.colour replaces ReferenceText[%d].Text.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineColour for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline	"%.16g"	ReferenceText[%d].Text.Ite	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
.doubleOffset		m[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline .fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Ite m[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline .filled	"%d"	ReferenceText[%d].Text.Ite m[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline .style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Ite m[%d].Outline	ReferenceText[%d].Text.Item[% d].Outline.style replaces ReferenceText[%d].Text.Item[% d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[% d].outlineStyle for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline .thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Ite m[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline .type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ReferenceText[%d].Text.Ite m[%d].Outline	ReferenceText[%d].Text.Item[% d].Outline.type replaces ReferenceText[%d].Text.Item[% d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[% d].outlineType for compatiblity.
ReferenceText[%d].Text.Item[%d].Outline .visible	"%d"	ReferenceText[%d].Text.Ite m[%d].Outline	
ReferenceText[%d].Text.Item[%d].Outline .width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Ite m[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].Outline.width.name	"%s"	ReferenceText[%d].Text.Item[%d].Outline	ReferenceText[%d].Text.Item[%d].Outline.width.name replaces ReferenceText[%d].Text.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.Item[%d].outlineName for compatiblity.
ReferenceText[%d].Text.Item[%d].bold	"%d"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].font	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].italic	"%d"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].italicAngle	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].language	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineFactor	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].lineWel	String representing enumeration such that: "0"=unset,	ReferenceText[%d].Text.Ite	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
dSupplementalSymbol	"1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	m[%d]	
ReferenceText[%d].Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].spaceFactor	"%.16g"	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].string	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textAspect	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
ReferenceText[%d].Text.Item[%d].textHeight	"%.16g"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	ReferenceText[%d].Text.Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Item[%d].textLineWidth.name	"%s"	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	ReferenceText[%d].Text.Item[%d]	
ReferenceText[%d].Text.Outline	NULL	ReferenceText[%d].Text	
ReferenceText[%d].Text.Outline.Side	NULL	ReferenceText[%d].Text.Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.Side.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Outline.Side.width.name	"%s"	ReferenceText[%d].Text.Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
ReferenceText[%d].Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline.colour replaces ReferenceText[%d].Text.outline Colour due to the increasing

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outline Colour for compatiblity.
ReferenceText[%d].Text.Outline.doubleOf fset	"%.16g"	ReferenceText[%d].Text.Ou tline	
ReferenceText[%d].Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text.Ou tline	
ReferenceText[%d].Text.Outline.filled	"%d"	ReferenceText[%d].Text.Ou tline	
ReferenceText[%d].Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	ReferenceText[%d].Text.Ou tline	ReferenceText[%d].Text.Outline .style replaces ReferenceText[%d].Text.outline Style due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outline Style for compatiblity.
ReferenceText[%d].Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text.Ou tline	
ReferenceText[%d].Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	ReferenceText[%d].Text.Ou tline	ReferenceText[%d].Text.Outline .type replaces ReferenceText[%d].Text.outline Type due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ReferenceText[%d].Text.outline Type for compatiblity.
ReferenceText[%d].Text.Outline.visible	"%d"	ReferenceText[%d].Text.Outline	
ReferenceText[%d].Text.Outline.width.*measurementType*	"%.16g"	ReferenceText[%d].Text.Outline	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.Outline.width.name	"%s"	ReferenceText[%d].Text.Outline	ReferenceText[%d].Text.Outline .width.name replaces ReferenceText[%d].Text.outline Name due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of ReferenceText[%d].Text.outline Name for compatiblity.
ReferenceText[%d].Text.bold	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.font	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.italic	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.italicAngle	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	ReferenceText[%d].Text	
ReferenceText[%d].Text.language	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.lineFactor	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.maxExtend	"%d"	ReferenceText[%d].Text	
ReferenceText[%d].Text.name	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.spaceFactor	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.strikethrough	String representing enumeration such that: "0"=none,	ReferenceText[%d].Text	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=single, "2"=double		
ReferenceText[%d].Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	ReferenceText[%d].Text	
ReferenceText[%d].Text.textAspect	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	ReferenceText[%d].Text	
ReferenceText[%d].Text.textHeight	"%.16g"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textLineWidth.*measurementType*	"%.16g"	ReferenceText[%d].Text	*measurementType* must be replaced with meterWidth or pixelWidth
ReferenceText[%d].Text.textLineWidth.name	"%s"	ReferenceText[%d].Text	
ReferenceText[%d].Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	ReferenceText[%d].Text	
ReferenceText[%d].Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	ReferenceText[%d].Text	
ReferenceText[%d].Text.underline	String representing enumeration such that: "0"=over, "1"=under	ReferenceText[%d].Text	
ReferenceText[%d].position	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after	ReferenceText[%d]	Only the before and after values are expected.
RevisionModifier	"%s"		
SafetyClassification	"%s"		
Statistical	NULL		
Text	NULL		*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d]	NULL	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline	NULL	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.Side	NULL	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.Side.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.Side.width.name	"%s"	*Text*.Item[%d].Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.colour replaces *Text*.Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			read/write of *Text*.Item[%d].outlineColour for compatiblity.
Text.Item[%d].Outline.doubleOffset	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.filled	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.style replaces *Text*.Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.type replaces *Text*.Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineType for compatiblity.
Text.Item[%d].Outline.visible	"%d"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].Outline.width.*measurementType*	"%.16g"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			replaced with meterWidth or pixelWidth
Text.Item[%d].Outline.width.name	"%s"	*Text*.Item[%d].Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Item[%d].Outline.width.name replaces *Text*.Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.Item[%d].outlineName for compatiblity.
Text.Item[%d].bold	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].font	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].italic	"%d"	*Text*.Item[%d]	*Text* must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].italicAngle	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].language	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].lineFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr,	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
Text.Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].spaceFactor	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.Item[%d].string	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross,	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"62"=orientationconstraint, "63"=datumtranslation		
Text.Item[%d].textAspect	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textHeight	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textLineWidth.*measurementType*	"%.16g"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth
Text.Item[%d].textLineWidth.name	"%s"	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText,

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	*Text*.Item[%d]	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline	NULL	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.Side	NULL	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Top, Bottom, Left or Right.
Text.Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.Side.width.*measurementType*	"%.16g"	*Text*.Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Text.Outline.Side.width.name	"%s"	*Text*.Outline.Side	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Side must be replaced by one of Top, Bottom, Left or Right.
Text.Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.colour replaces *Text*.outlineColour due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineColour for compatiblity.
Text.Outline.doubleOffset	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.filled	"%d"	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.style replaces *Text*.outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineStyle for compatiblity.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.type replaces *Text*.outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineType for compatiblity.
Text.Outline.visible	"%d"	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.Outline.width.*measurementType*	"%.16g"	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.Outline.width.name	"%s"	*Text*.Outline	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			Text.Outline.width.name replaces *Text*.outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of *Text*.outlineName for compatiblity.
Text.bold	"%d"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.font	"%s"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.italic	"%d"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.italicAngle	"%.16g"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			UnitSymbol[%d].
Text.justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.language	"%s"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.lineFactor	"%.16g"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.maxExtend	"%d"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.name	"%s"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.spaceFactor	"%.16g"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.strikethrough	String representing enumeration such that: "0"=none,	*Text*	*Text* must be replaced by one of

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=single, "2"=double		ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.subscript	String representing enumeration such that: "0"=sub, "1"=super	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textAspect	"%.16g"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textHeight	"%.16g"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textLineWidth.*measurementType*	"%.16g"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
			measurementType must be replaced with meterWidth or pixelWidth

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Text.textLineWidth.name	"%s"	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
Text.underline	String representing enumeration such that: "0"=over, "1"=under	*Text*	*Text* must be replaced by one of ChamferSymbol, ValueText, DualValueText, StyleText, DualUnitSymbol[%d] or UnitSymbol[%d].
ValueToCustomer	"%s"		
_____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	_____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
alignment	String representing enumeration such that: "0"=middle, "1"=bottom		
allAround	"%d"		
allOver	"%d"		
angleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
angleNumerator	"%.16g"		
appendedTextSpaceFactor	"%.16g"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
basic	"%d"		
blanked	"%d"		
bold	"%d"		
causality	String representing enumeration such that: "0"=key, "1"=functional, "2"=reference, "3"=associated, "4"=pmi		
chamferDisplayType	String representing enumeration such that: "0"=symbol, "1"=size, "2"=sizeangle, "3"=anglesize		
chamferLeaderType	String representing enumeration such that: "0"=perpendicular, "1"=parallel, "2"=linear		
chamferSeparatorCapital	"%d"		
chamferSpacing	"%.16g"		
chamferSymbolType	String representing enumeration such that: "0"=none, "1"=prefix, "2"=suffix		
commaAsDecimal	"%d"		
deviation	"%s"		
diameterPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dimensionLeadingZero	"%d"		
dimensionLineBetweenArrows	"%d"		
dimensionLineTrim	"%d"		
dimensionTrailingZero	"%d"		
direction	"%.16g %.16g %.16g"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
documentation	"%d"		
dualDimensionLineCenter	"%d"		
dualLowerDeltaDenominator	"%d"		
dualPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
dualPrecision	"%d"		
dualTolerancePrecision	"%d"		
dualType	String representing enumeration such that: "0"=positional, "1"=bracket, "2"=lineseparated		
dualUnit	"%s"		
dualUnitText	"%d"		
dualUpperDeltaDenominator	"%d"		
dualValueDenominator	"%d"		
featureOfSize	"%d"		
fitGrade	"%d"		
flag	"%d"		
foldLocation	"%.16g %.16g %.16g"		
font	"%s"		
fraction	"%d"		
fractionSize	String representing enumeration such that: "0"=full, "1"=twothirds, "2"=threequarters		
grade	"%d"		
group	"%s"		
inspection	"%d"		
inspectionDisplay	String representing enumeration such that: "0"=unset, "1"=none, "2"=before, "3"=after, "4"=beforeafter, "5"=all		
isReference	"%d"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
limitDisplay	String representing enumeration such that: "0"=none, "1"=limitfit		
limitFitOrder	String representing enumeration such that: "0"=valuelimitfittolerance, "1"=tolerancevaluelimitfit, "2"=valuetolerancelimitfit		
limitFitParenthesis	String representing enumeration such that: "0"=none, "1"=valuelimitfit, "2"=limitfit, "3"=tolerance, "4"=value, "5"=valuetolerance		
lineFactor	"%.16g"		
linearPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
lowerDelta	"%.16g"		
lowerDeltaDenominator	"%d"		
majorAngle	"%d"		
manual	"%d"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
narrowLeaderAngle	"%.16g"		
narrowOffset	"%.16g"		
notToScale	"%d"		
ordinateBaselineZero	"%d"		
origin	String representing enumeration such that: "0"=first, "1"=second		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
panZoom	"%d"		
patternCount	"%d"		
pitchDiaDeviation	"%s"		
pitchDiaGrade	"%d"		
precision	"%d"		
projected	"%d"		
radialPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
referenceContent	String representing enumeration such that: "0"=value, "1"=prefix, "2"=tolerance		
referenceDisplay	String representing enumeration such that: "0"=parenthesis, "1"=reference, "2"=matched		
singleSideFirst	"%d"		
singleSideLength	"%.16g"		
singleSided	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
standard	String representing enumeration such that: "0"=asme_y145m_1994, "1"=jis, "2"=iso, "3"=bs, "4"=ansi_y145m_1982, "5"=asme_y1441m_2003, "6"=din, "7"=gm_addendum_1994, "8"=asme_y145_2009		
statisticalPlacement	String representing enumeration such that: "0"=before, "1"=above, "2"=below, "3"=after		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
style	String representing enumeration such that: "0"=lineardiametral, "1"=radial, "2"=controlledradial, "3"=diametral, "4"=limits, "5"=ordinate, "6"=sphericalradial, "7"=sphericaldiametral, "8"=narrow, "9"=none		
subscript	String representing enumeration such that: "0"=sub, "1"=super		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLeaderPosition	String representing enumeration such that: "0"=none, "1"=aboveleader, "2"=afterleader, "3"=abovestub, "4"=afterstub		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
threadClass	"%d"		
toleranceAngleFormat	String representing enumeration such that: "0"=degrees, "1"=seconds, "2"=minutes, "3"=wholedegrees		
toleranceDisplay	String representing enumeration such that: "0"=minuslimit1, "1"=bilateral, "2"=pluslimit1, "3"=pluslimit2, "4"=equalbilateral, "5"=minuslimit2, "6"=upperunilateral, "7"=lowerunilateral		
toleranceLeadingZero	"%d"		
tolerancePrecision	"%d"		
toleranceTextSpaceFactor	"%.16g"		
toleranceTrailingZero	"%d"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
type	String representing enumeration such that: "0"=curvelength, "1"=linear, "2"=angular, "3"=radial		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
unitText	"%d"		
upperDelta	"%.16g"		
upperDeltaDenominator	"%d"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
value	"%.16g"		
valueDenominator	"%d"		
zeroToleranceDisplay	String representing enumeration such that: "0"=displayzero, "1"=blank, "2"=suppresstrailingzero, "3"=omit		
zeroToleranceSign	String representing enumeration such that: "0"=none, "1"=plus, "2"=minus		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.37 PMI_TABLE_TYPE (0x0311)

Table 241 — PMI_TABLE_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Column[%d]	NULL		%d index cannot exceed/equal columnCount.
Column[%d].ColumnValue[%d]	NULL	Column[%d]	
Column[%d].ColumnValue[%d].List.Value[%d]	NULL	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].List.Value[%d].description	"%s"	Column[%d].ColumnValue[%d].List.Value[%d]	
Column[%d].ColumnValue[%d].List.Value[%d].value	"%s"	Column[%d].ColumnValue[%d].List.Value[%d]	
Column[%d].ColumnValue[%d].List.type	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].editable	"%d"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].format	"%s"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].maxExclusive	"%.16g"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].maxInclusive	"%.16g"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].maxLength	"%d"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].minExclusive	"%.16g"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].minInclusive	"%.16g"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].minLength	"%d"	Column[%d].ColumnValue[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Column[%d].ColumnValue[%d].referenceRef	"%s"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].stepValue	"%.16g"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].title	"%s"	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].type	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	Column[%d].ColumnValue[%d]	
Column[%d].ColumnValue[%d].value	"%s"	Column[%d].ColumnValue[%d]	
Column[%d].Outline	NULL	Column[%d]	
Column[%d].Outline.Side	NULL	Column[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Column[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Column[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Column[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Column[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Column[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Column[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Column[%d].Outline.Side.width.*measurementType*	"%.16g"	Column[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Column[%d].Outline.Side.width.name	"%s"	Column[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
Column[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Column[%d].Outline	Column[%d].Outline.colour replaces Column[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Column[%d].outlineColour for compatiblity.
Column[%d].Outline.doubleOffset	"%.16g"	Column[%d].Outline	
Column[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Column[%d].Outline	
Column[%d].Outline.filled	"%d"	Column[%d].Outline	
Column[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Column[%d].Outline	Column[%d].Outline.style replaces Column[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Column[%d].outlineStyle for compatiblity.
Column[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Column[%d].Outline	
Column[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart,	Column[%d].Outline	Column[%d].Outline.type replaces Column[%d].outlineType due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		semantics. We recommend continuing to support read/write of Column[%d].outlineType for compatiblity.
Column[%d].Outline.visible	"%d"	Column[%d].Outline	
Column[%d].Outline.width.*measurementType*	"%.16g"	Column[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Column[%d].Outline.width.name	"%s"	Column[%d].Outline	Column[%d].Outline.width. name replaces Column[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Column[%d].outlineName for compatiblity.
Column[%d].blanked	"%d"	Column[%d]	
Column[%d].bold	"%d"	Column[%d]	
Column[%d].commaAsDecimal	"%d"	Column[%d]	
Column[%d].font	"%s"	Column[%d]	
Column[%d].italic	"%d"	Column[%d]	
Column[%d].italicAngle	"%.16g"	Column[%d]	
Column[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	Column[%d]	
Column[%d].language	"%s"	Column[%d]	
Column[%d].lineFactor	"%.16g"	Column[%d]	
Column[%d].number	"%d"	Column[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Column[%d].spaceFactor	"%.16g"	Column[%d]	
Column[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	Column[%d]	
Column[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	Column[%d]	
Column[%d].symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Column[%d]	
Column[%d].textAspect	"%.16g"	Column[%d]	
Column[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Column[%d]	
Column[%d].textDirection	"%.16g %.16g %.16g"	Column[%d]	
Column[%d].textHeight	"%.16g"	Column[%d]	
Column[%d].textLineWidth.*measurementType*	"%.16g"	Column[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Column[%d].textLineWidth.name	"%s"	Column[%d]	
Column[%d].textOrigin	"%.16g %.16g %.16g"	Column[%d]	
Column[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Column[%d]	
Column[%d].underline	String representing enumeration such that: "0"=over, "1"=under	Column[%d]	
Column[%d].width	"%.16g"	Column[%d]	
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Leader[%d].perpendicularToTerminato r	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"5"=dashed, "6"=solid, "7"=centreline		Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
RevisionModifier	"%s"		
Row[%d]	NULL		%d index cannot exceed/equal rowCount.
Row[%d].Cell[%d]	NULL	Row[%d]	
Row[%d].Cell[%d].CellValue[%d]	NULL	Row[%d].Cell[%d]	
Row[%d].Cell[%d].CellValue[%d].List.Value[%d]	NULL	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].List.Value[%d].description	"%s"	Row[%d].Cell[%d].CellValue[%d].List.Value[%d]	
Row[%d].Cell[%d].CellValue[%d].List.Value[%d].value	"%s"	Row[%d].Cell[%d].CellValue[%d].List.Value[%d]	
Row[%d].Cell[%d].CellValue[%d].List.type	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].editable	"%d"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].format	"%s"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].maxExclusive	"%.16g"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].maxInclusive	"%.16g"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].maxLength	"%d"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].minExclusive	"%.16g"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].minInclusive	"%.16g"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].minLength	"%d"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].refere	"%s"	Row[%d].Cell[%d].CellValue[	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
nceRef		%d]	
Row[%d].Cell[%d].CellValue[%d].stepValue	"%.16g"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].title	"%s"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].type	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].CellValue[%d].value	"%s"	Row[%d].Cell[%d].CellValue[%d]	
Row[%d].Cell[%d].Outline	NULL	Row[%d].Cell[%d]	
Row[%d].Cell[%d].Outline.Side	NULL	Row[%d].Cell[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Cell[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Cell[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Cell[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Row[%d].Cell[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Cell[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Row[%d].Cell[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Cell[%d].Outline.Side.width.*measurementType*	"%.16g"	Row[%d].Cell[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Row[%d].Cell[%d].Outline.Side.width.name	"%s"	Row[%d].Cell[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Row[%d].Cell[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Cell[%d].Outline	Row[%d].Cell[%d].Outline.colour replaces Row[%d].Cell[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Row[%d].Cell[%d].outlineColour for compatiblity.
Row[%d].Cell[%d].Outline.doubleOffset	"%.16g"	Row[%d].Cell[%d].Outline	
Row[%d].Cell[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Cell[%d].Outline	
Row[%d].Cell[%d].Outline.filled	"%d"	Row[%d].Cell[%d].Outline	
Row[%d].Cell[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Row[%d].Cell[%d].Outline	Row[%d].Cell[%d].Outline.style replaces Row[%d].Cell[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Row[%d].Cell[%d].outlineStyle for compatiblity.
Row[%d].Cell[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Row[%d].Cell[%d].Outline	
Row[%d].Cell[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart,	Row[%d].Cell[%d].Outline	Row[%d].Cell[%d].Outline.type replaces Row[%d].Cell[%d].outlineType due to the increasing complexity of Outline

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram		semantics. We recommend continuing to support read/write of Row[%d].Cell[%d].outlineType for compatiblity.
Row[%d].Cell[%d].Outline.visible	"%d"	Row[%d].Cell[%d].Outline	
Row[%d].Cell[%d].Outline.width.*measurementType*	"%.16g"	Row[%d].Cell[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Row[%d].Cell[%d].Outline.width.name	"%s"	Row[%d].Cell[%d].Outline	Row[%d].Cell[%d].Outline.width.name replaces Row[%d].Cell[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Row[%d].Cell[%d].outlineName for compatiblity.
Row[%d].Cell[%d].blanked	"%d"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].bold	"%d"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].commaAsDecimal	"%d"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].font	"%s"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].italic	"%d"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].italicAngle	"%.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	Row[%d].Cell[%d]	
Row[%d].Cell[%d].language	"%s"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].lineFactor	"%.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].mergeAcross	"%d"	Row[%d].Cell[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Row[%d].Cell[%d].mergeDown	"%d"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].number	"%d"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].spaceFactor	"%.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	Row[%d].Cell[%d]	
Row[%d].Cell[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	Row[%d].Cell[%d]	
Row[%d].Cell[%d].symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textAspect	"%.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textDirection	"%.16g %.16g %.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textHeight	"%.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textLineWidth.*measurementType*	"%.16g"	Row[%d].Cell[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Row[%d].Cell[%d].textLineWidth.name	"%s"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textOrigin	"%.16g %.16g %.16g"	Row[%d].Cell[%d]	
Row[%d].Cell[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Row[%d].Cell[%d]	
Row[%d].Cell[%d].underline	String representing enumeration such that: "0"=over, "1"=under	Row[%d].Cell[%d]	
Row[%d].Outline	NULL	Row[%d]	
Row[%d].Outline.Side	NULL	Row[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Row[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Row[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Row[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Outline.Side.width.*measurementType*	"%.16g"	Row[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Row[%d].Outline.Side.width.name	"%s"	Row[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Row[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Outline	Row[%d].Outline.colour replaces Row[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Row[%d].outlineColour for compatiblity.
Row[%d].Outline.doubleOffset	"%.16g"	Row[%d].Outline	
Row[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d].Outline	
Row[%d].Outline.filled	"%d"	Row[%d].Outline	
Row[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Row[%d].Outline	Row[%d].Outline.style replaces Row[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Row[%d].outlineStyle for compatiblity.
Row[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Row[%d].Outline	
Row[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Row[%d].Outline	Row[%d].Outline.type replaces Row[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Row[%d].outlineType for compatiblity.
Row[%d].Outline.visible	"%d"	Row[%d].Outline	
Row[%d].Outline.width.*measurementType*	"%.16g"	Row[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Row[%d].Outline.width.name	"%s"	Row[%d].Outline	Row[%d].Outline.width.name replaces Row[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of Row[%d].outlineName for compatiblity.
Row[%d].RowValue[%d]	NULL	Row[%d]	
Row[%d].RowValue[%d].List.Value[%d]	NULL	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].List.Value[%d].description	"%s"	Row[%d].RowValue[%d].List.Value[%d]	
Row[%d].RowValue[%d].List.Value[%d].value	"%s"	Row[%d].RowValue[%d].List.Value[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Row[%d].RowValue[%d].List.type	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].editable	"%d"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].format	"%s"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].maxExclusive	"%.16g"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].maxInclusive	"%.16g"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].maxLength	"%d"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].minExclusive	"%.16g"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].minInclusive	"%.16g"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].minLength	"%d"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].referenceRef	"%s"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].stepValue	"%.16g"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].title	"%s"	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].type	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	Row[%d].RowValue[%d]	
Row[%d].RowValue[%d].value	"%s"	Row[%d].RowValue[%d]	
Row[%d].blanked	"%d"	Row[%d]	
Row[%d].bold	"%d"	Row[%d]	
Row[%d].commaAsDecimal	"%d"	Row[%d]	
Row[%d].font	"%s"	Row[%d]	
Row[%d].header	"%d"	Row[%d]	
Row[%d].height	"%.16g"	Row[%d]	
Row[%d].italic	"%d"	Row[%d]	
Row[%d].italicAngle	"%.16g"	Row[%d]	
Row[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	Row[%d]	
Row[%d].language	"%s"	Row[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Row[%d].lineFactor	"%.16g"	Row[%d]	
Row[%d].number	"%d"	Row[%d]	
Row[%d].spaceFactor	"%.16g"	Row[%d]	
Row[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	Row[%d]	
Row[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	Row[%d]	
Row[%d].symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d]	
Row[%d].textAspect	"%.16g"	Row[%d]	
Row[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Row[%d]	
Row[%d].textDirection	"%.16g %.16g %.16g"	Row[%d]	
Row[%d].textHeight	"%.16g"	Row[%d]	
Row[%d].textLineWidth.*measurementType*	"%.16g"	Row[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Row[%d].textLineWidth.name	"%s"	Row[%d]	
Row[%d].textOrigin	"%.16g %.16g %.16g"	Row[%d]	
Row[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Row[%d]	
Row[%d].underline	String representing enumeration such that: "0"=over, "1"=under	Row[%d]	
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			Right.
TextFormat[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fille	"%d"	TextFormat[%d].Item[%d].O	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
d		utline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove,	TextFormat[%d].Item[%d]	

"Key" Property Atom Value String	"Value" Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane,	TextFormat[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single,	TextFormat[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=double		
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	
UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			utlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough,	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single,	UnitSymbol[%d]	

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"2"=double		
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
			references are tied together as an intersection reference.
____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which reference is an origin reference.
accountabilityId	"%d"		
allAround	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft,		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
columnCount	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
rowCount	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.38 PMI_ORGANISATION_TYPE (0x0245)

Table 242 — PMI_ORGANISATION_TYPE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Description	"%s"		
DisplayPlane.origin	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.xaxis' and 'DisplayPlane.zaxis'
DisplayPlane.xaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.zaxis'
DisplayPlane.zaxis	"%.16g %.16g %.16g"		This is property records the original defintion of the PMI display plane Should also have 'DisplayPlane.origin' and 'DisplayPlane.xaxis'
FeatureIdentification	"%s"		
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthof unit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be

	editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy, "17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle,		hidden.

	"73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
Keyword[%d]	"%s"		
LAYER	"%d"		
Leader[%d]	NULL		
Leader[%d].Jog[%d]	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].Reference	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d]	
Leader[%d].Reference[1]	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path. This property formating is used to semantically tie a singlar semantic aspect with an entity.	Leader[%d].Reference	This extension is for use when the Leader is referencing the intersection of two targets. This property contains the second target with the Leader[%d].Reference property containing the first
Leader[%d].allAroundSymbolSize	"%.16g"	Leader[%d]	
Leader[%d].arrowAngle	"%.16g"	Leader[%d]	
Leader[%d].arrowColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].arrowLength	"%.16g"	Leader[%d]	
Leader[%d].arrowLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].arrowLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	

Leader[%d].arrowOutSideLength	"%.16g"	Leader[%d]	
Leader[%d].arrowPlacement	String representing enumeration such that: "0"=out, "1"=in	Leader[%d]	
Leader[%d].arrowType	String representing enumeration such that: "0"=closed, "1"=open, "2"=filled, "3"=none, "4"=origin, "5"=cross, "6"=dot, "7"=filleddot, "8"=wedge, "9"=plus, "10"=x, "11"=closedsolid, "12"=closeddouble, "13"=closeddoublesolid, "14"=openddouble, "15"=filleddouble, "16"=integral, "17"=box, "18"=filledbox, "19"=datum, "20"=filleddatum, "21"=solidorigin, "22"=filledorigin, "23"=xorigin, "24"=solidbox, "25"=roundbackdart, "26"=solidroundbackdart, "27"=filledroundbackdart	Leader[%d]	
Leader[%d].arrowWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].arrowWidth.name	"%s"	Leader[%d]	
Leader[%d].colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].direction	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].dotDiameter	"%.16g"	Leader[%d]	
Leader[%d].extensionJogAngle	"%.16g"	Leader[%d]	
Leader[%d].extensionJogDefintion	String representing enumeration such that: "0"=startend, "1"=startangle, "2"=endangle	Leader[%d]	
Leader[%d].extensionJogEnd	"%.16g"	Leader[%d]	
Leader[%d].extensionJogOut	"%d"	Leader[%d]	
Leader[%d].extensionJogStart	"%.16g"	Leader[%d]	
Leader[%d].extensionJogging	String representing enumeration such that: "0"=on, "1"=off, "2"=application	Leader[%d]	
Leader[%d].extensionLineColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Leader[%d]	
Leader[%d].extensionLineExtension	"%.16g"	Leader[%d]	
Leader[%d].extensionLineGap	"%.16g"	Leader[%d]	
Leader[%d].extensionLineThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	

Leader[%d].extensionLineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].extensionWidth.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].extensionWidth.name	"%s"	Leader[%d]	
Leader[%d].halfArrow	String representing enumeration such that: "0"=left, "1"=right, "2"=full	Leader[%d]	
Leader[%d].leaderAttachment	String representing enumeration such that: "0"=top, "1"=middle, "2"=bottom	Leader[%d]	
Leader[%d].lineTextGap	"%.16g"	Leader[%d]	
Leader[%d].lineType	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Leader[%d]	
Leader[%d].offsetCentre	"%d"	Leader[%d]	
Leader[%d].perpendicularToStub	"%d"	Leader[%d]	
Leader[%d].perpendicularToTerminator	"%d"	Leader[%d]	
Leader[%d].radiusToCentre	"%d"	Leader[%d]	
Leader[%d].stubAttachment	String representing enumeration such that: "0"=side, "1"=underline	Leader[%d]	
Leader[%d].stubDirection	String representing enumeration such that: "0"=left, "1"=right, "2"=inferred	Leader[%d]	
Leader[%d].stubLength	"%.16g"	Leader[%d]	
Leader[%d].suppressed	"%d"	Leader[%d]	
Leader[%d].tParm	"%.16g"	Leader[%d]	
Leader[%d].terminator	"%.16g %.16g %.16g"	Leader[%d]	
Leader[%d].textOverLeaderFactor	"%.16g"	Leader[%d]	
Leader[%d].textOverStubFactor	"%.16g"	Leader[%d]	
Leader[%d].thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Leader[%d]	
Leader[%d].uvParm.U	"%.16g"	Leader[%d]	Expect to also have

			Leader[%d].uvParam.V.
Leader[%d].uvParm.V	"%.16g"	Leader[%d]	Expect to also have Leader[%d].uvParam.U.
Leader[%d].width.*measurementType*	"%.16g"	Leader[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
Leader[%d].width.name	"%s"	Leader[%d]	
Outline	NULL		
Outline.Side	NULL	Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.Side.width.*measurementType*	"%.16g"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
Outline.Side.width.name	"%s"	Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	Outline.colour replaces outlineColour due to the increasing complexity of Outline semantics. We

			recommend continuing to support read/write of outlineColour for compatiblity.
Outline.doubleOffset	"%.16g"	Outline	
Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	Outline	
Outline.filled	"%d"	Outline	
Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	Outline	Outline.style replaces outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineStyle for compatiblity.
Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	Outline	
Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	Outline	Outline.type replaces outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of outlineType for compatiblity.
Outline.visible	"%d"	Outline	
Outline.width.*measurementType*	"%.16g"	Outline	*measurementType* must be replaced with meterWidth or pixelWidth
Outline.width.name	"%s"	Outline	Outline.width.name replaces outlineName due to the increasing complexity

			of Outline semantics. We recommend continuing to support read/write of outlineName for compatiblity.
PMIGeometryColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextBackgroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextFlatToScreenOpacity	"%s"		replaces PMITextBackgroundOpacity
PMITextForegroundColor	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
PMITextInPlaneOpacity	"%s"		
RevisionModifier	"%s"		
SafetyClassification	"%s"		
TextFormat[%d]	NULL		
TextFormat[%d].Item[%d]	NULL	TextFormat[%d]	
TextFormat[%d].Item[%d].Outline	NULL	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].Outline.Side	NULL	TextFormat[%d].Item[%d]. Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side. thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]. Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.Side.	"%.16g"	TextFormat[%d].Item[%d].	*Side* must be replaced by

width.*measurementType*		Outline.Side	one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.Side.width.name	"%s"	TextFormat[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
TextFormat[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.colour replaces TextFormat[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineColour for compatiblity.
TextFormat[%d].Item[%d].Outline.doubleOffset	"%.16g"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.filled	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.style replaces TextFormat[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of

			TextFormat[%d].Item[%d].outlineStyle for compatiblity.
TextFormat[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.type replaces TextFormat[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineType for compatiblity.
TextFormat[%d].Item[%d].Outline.visible	"%d"	TextFormat[%d].Item[%d].Outline	
TextFormat[%d].Item[%d].Outline.width.*measurementType*	"%.16g"	TextFormat[%d].Item[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].Outline.width.name	"%s"	TextFormat[%d].Item[%d].Outline	TextFormat[%d].Item[%d].Outline.width.name replaces TextFormat[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of TextFormat[%d].Item[%d].outlineName for compatiblity.
TextFormat[%d].Item[%d].bold	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].font	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints,	TextFormat[%d].Item[%d]	

	"2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none		
TextFormat[%d].Item[%d].italic	"%d"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].italicAngle	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].language	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr, "18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot,	TextFormat[%d].Item[%d]	

	"49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge		
TextFormat[%d].Item[%d].spaceFactor	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].string	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline, "24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textAspect	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textHeight	"%.16g"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textLineWid	"%.16g"	TextFormat[%d].Item[%d]	*measurementType* must be

th.*measurementType*			replaced with meterWidth or pixelWidth
TextFormat[%d].Item[%d].textLineWidth.name	"%s"	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d].Item[%d]	
TextFormat[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d].Item[%d]	
TextFormat[%d].bold	"%d"	TextFormat[%d]	
TextFormat[%d].font	"%s"	TextFormat[%d]	
TextFormat[%d].italic	"%d"	TextFormat[%d]	
TextFormat[%d].italicAngle	"%.16g"	TextFormat[%d]	
TextFormat[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	TextFormat[%d]	
TextFormat[%d].language	"%s"	TextFormat[%d]	
TextFormat[%d].lineFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].spaceFactor	"%.16g"	TextFormat[%d]	
TextFormat[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	TextFormat[%d]	
TextFormat[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	TextFormat[%d]	
TextFormat[%d].textAspect	"%.16g"	TextFormat[%d]	
TextFormat[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	TextFormat[%d]	
TextFormat[%d].textHeight	"%.16g"	TextFormat[%d]	
TextFormat[%d].textLineWidth.*measurementType*	"%.16g"	TextFormat[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
TextFormat[%d].textLineWidth.name	"%s"	TextFormat[%d]	
TextFormat[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	TextFormat[%d]	
TextFormat[%d].underline	String representing enumeration such that: "0"=over, "1"=under	TextFormat[%d]	

UnitSymbol[%d]	NULL		
UnitSymbol[%d].Item[%d]	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Item[%d].Outline	NULL	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].Outline.Side	NULL	UnitSymbol[%d].Item[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Item[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.colour replaces UnitSymbol[%d].Item[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].o

			utlineColour for compatiblity.
UnitSymbol[%d].Item[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.filled	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.style replaces UnitSymbol[%d].Item[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Item[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.type replaces UnitSymbol[%d].Item[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineType for compatiblity.
UnitSymbol[%d].Item[%d].Outline.visible	"%d"	UnitSymbol[%d].Item[%d].Outline	
UnitSymbol[%d].Item[%d].Outline.widt	"%.16g"	UnitSymbol[%d].Item[%d].	*measurementType* must be

h.*measurementType*		Outline	replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].Outline.width.name	"%s"	UnitSymbol[%d].Item[%d].Outline	UnitSymbol[%d].Item[%d].Outline.width.name replaces UnitSymbol[%d].Item[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].Item[%d].outlineName for compatiblity.
UnitSymbol[%d].Item[%d].bold	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].font	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].format	String representing enumeration such that: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime, "11"=none	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italic	"%d"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].italicAngle	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].language	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].lineWeldSupplementalSymbol	String representing enumeration such that: "0"=unset, "1"=convex, "2"=toesblended, "3"=concave, "4"=flush, "5"=permanentbackingstrip, "6"=removablebackingstrip, "7"=stud, "8"=intermittent, "9"=backing, "10"=backingremoveafterweld, "11"=spacer, "12"=consumableinsert, "13"=flusheskd, "14"=machininggradedjunction, "15"=surfacefinishbasic, "16"=surfacefinishbasiccomplete, "17"=surfacefinishmrr,	UnitSymbol[%d].Item[%d]	

	"18"=surfacefinishmrrcomplete, "19"=surfacefinishmrp, "20"=surfacefinishmrpcomplete, "21"=meltthrough		
UnitSymbol[%d].Item[%d].lineWeldSymbol	String representing enumeration such that: "0"=unset, "1"=jisseam, "2"=singlebevelgroove, "3"=singleugroove, "4"=edgeflange, "5"=surface, "6"=steepflankedsinglebevel, "7"=flaresinglevgroove, "8"=surfacejoint, "9"=singlevgroove, "10"=inclinedjoint, "11"=plug, "12"=fillet, "13"=squaregroove, "14"=spot, "15"=singlej, "16"=square, "17"=overlay, "18"=singleu, "19"=singlejgroove, "20"=doubleflange, "21"=singleflange, "22"=spotprojected, "23"=plugandslot, "24"=singlebevelbroadrootface, "25"=edge, "26"=jisstaggeredfillet1, "27"=singlev, "28"=flaresinglebevelgroove, "29"=singlevbroadrootface, "30"=steepflankedsinglev, "31"=removablebackingstrip, "32"=permanentbackingstrip, "33"=jisfillet, "34"=singlebevel, "35"=backing, "36"=foldjoint, "37"=jisstaggeredfillet2, "38"=seam, "39"=bead, "40"=stud, "41"=intermittent, "42"=backingplate, "43"=meltthrough, "44"=solder, "45"=kgroove, "46"=stake, "47"=keyhole, "48"=jispot, "49"=jisspotflatelectrode, "50"=eskdheat, "51"=eskdsmooth, "52"=eskdstaggerchain, "53"=eskdstaggercheck, "54"=eskdnotallaround, "55"=eskddiameter, "56"=eskdnumber, "57"=isoedge	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].spaceFactor	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].string	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].symbol	String representing enumeration such that: "0"=centreline, "1"=partingline, "2"=depth, "3"=countersink, "4"=conicaltaper, "5"=slope, "6"=counterbore, "7"=square, "8"=phi, "9"=plusminus, "10"=degrees, "11"=between, "12"=arclength, "13"=leftparenthesis, "14"=rightparenthesis, "15"=projectedtolerance, "16"=mmc, "17"=lmc, "18"=freestate, "19"=ohm, "20"=circularrunout, "21"=totalrunout, "22"=profileofasurface, "23"=profileofaline,	UnitSymbol[%d].Item[%d]	

	"24"=flatness, "25"=straightness, "26"=symmetry, "27"=perpendicularity, "28"=parallelism, "29"=cylindricity, "30"=concentricity, "31"=circularity, "32"=angularity, "33"=micro, "34"=position, "35"=envelope, "36"=rfs, "37"=tangentplane, "38"=lessthanorequal, "39"=greaterthanorequal, "40"=threadprefix, "41"=triangle, "42"=statistical, "43"=radius, "44"=circledu, "45"=fitfunction, "46"=safetycompliance, "47"=quantity, "48"=independency, "49"=continuousfeature, "50"=spotface, "51"=sphi, "52"=sphere, "53"=lefttaper, "54"=righttaper, "55"=leftpitch, "56"=rightpitch, "57"=approximatedimension, "58"=axisintersection, "59"=eskdperpendicular, "60"=eskdparallel, "61"=eskdcross, "62"=orientationconstraint, "63"=datumtranslation		
UnitSymbol[%d].Item[%d].textAspect	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textHeight	"%.16g"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d].Item[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Item[%d].textLineWidth.name	"%s"	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Item[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d].Item[%d]	
UnitSymbol[%d].Outline	NULL	UnitSymbol[%d]	
UnitSymbol[%d].Outline.Side	NULL	UnitSymbol[%d].Outline	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom,	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or

	"5"=dashed, "6"=solid, "7"=centreline		Right.
UnitSymbol[%d].Outline.Side.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.Side.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
			measurementType must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.Side.width.name	"%s"	UnitSymbol[%d].Outline.Side	*Side* must be replaced by one of Top, Bottom, Left or Right.
UnitSymbol[%d].Outline.colour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.colour replaces UnitSymbol[%d].outlineColour due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineColour for compatiblity.
UnitSymbol[%d].Outline.doubleOffset	"%.16g"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.fillColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.filled	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.style	String representing enumeration such that: "0"=unset, "1"=longdashed, "2"=dotted, "3"=dotteddashed, "4"=phantom, "5"=dashed, "6"=solid, "7"=centreline	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.style replaces UnitSymbol[%d].outlineStyle due to the increasing complexity of Outline semantics. We recommend

			continuing to support read/write of UnitSymbol[%d].outlineStyle for compatiblity.
UnitSymbol[%d].Outline.thickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.type	String representing enumeration such that: "0"=box, "1"=triangle, "2"=circle, "3"=ellipse, "4"=underline, "5"=square, "6"=scoredcircle, "7"=diamond, "8"=flagright, "9"=flagleft, "10"=flagboth, "11"=oblong, "12"=oblongright, "13"=oblongleft, "14"=sticking, "15"=set, "16"=fixedsupport, "17"=nota, "18"=symmetricalpart, "19"=symmetricalset, "20"=scoredrectangle, "21"=parallelogram	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.type replaces UnitSymbol[%d].outlineType due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineType for compatiblity.
UnitSymbol[%d].Outline.visible	"%d"	UnitSymbol[%d].Outline	
UnitSymbol[%d].Outline.width.*measurementType*	"%.16g"	UnitSymbol[%d].Outline	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].Outline.width.name	"%s"	UnitSymbol[%d].Outline	UnitSymbol[%d].Outline.width.name replaces UnitSymbol[%d].outlineName due to the increasing complexity of Outline semantics. We recommend continuing to support read/write of UnitSymbol[%d].outlineName for compatiblity.
UnitSymbol[%d].bold	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].font	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].italic	"%d"	UnitSymbol[%d]	

UnitSymbol[%d].italicAngle	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre	UnitSymbol[%d]	
UnitSymbol[%d].language	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].lineFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].maxExtend	"%d"	UnitSymbol[%d]	
UnitSymbol[%d].name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].spaceFactor	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].strikethrough	String representing enumeration such that: "0"=none, "1"=single, "2"=double	UnitSymbol[%d]	
UnitSymbol[%d].subscript	String representing enumeration such that: "0"=sub, "1"=super	UnitSymbol[%d]	
UnitSymbol[%d].textAspect	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string	UnitSymbol[%d]	
UnitSymbol[%d].textHeight	"%.16g"	UnitSymbol[%d]	
UnitSymbol[%d].textLineWidth.*measurementType*	"%.16g"	UnitSymbol[%d]	*measurementType* must be replaced with meterWidth or pixelWidth
UnitSymbol[%d].textLineWidth.name	"%s"	UnitSymbol[%d]	
UnitSymbol[%d].textOrientation	String representing enumeration such that: "0"=horizontal, "1"=vertical	UnitSymbol[%d]	
UnitSymbol[%d].textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin	UnitSymbol[%d]	
UnitSymbol[%d].underline	String representing enumeration such that: "0"=over, "1"=under	UnitSymbol[%d]	
ValueToCustomer	"%s"		
____JtTkIntersectionReference___*reference0*	ORIGIN*reference1*	____JtTkIntersectionReference___*reference1*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d

			%d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkIntersectionReference___*reference1*	ORIGIN*reference0*	_____JtTkIntersectionReference___*reference0*	ORIGIN is optional and should only be added if the Intersection reference is also an Origin reference.
			reference0 and *reference1* should be in the format "%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.
			Should exist in pairs, with the opposite ordering of values also persited, to identify both way which references are tied together as an intersection reference.
_____JtTkOriginReference___*reference*	"%d %d %d %s" such that first in is the dst id, second the dst type, third the reason(optional) and the string the path.		*reference* is the property value encoded also into the key identifying which

			reference is an origin reference.
accountabilityId	"%d"		
attachmentType	String representing enumeration such that: "0"=angulardimension, "1"=ongeometry, "2"=lineardimension, "3"=leader, "4"=stacked, "5"=ordinatedimension, "6"=radialdimension, "7"=diametraldimension, "8"=noleader, "9"=stackedleft, "10"=stackedright, "11"=stackedabove		
blanked	"%d"		
bold	"%d"		
commaAsDecimal	"%d"		
flag	"%d"		
font	"%s"		
group	"%s"		
italic	"%d"		
italicAngle	"%.16g"		
justification	String representing enumeration such that: "0"=right, "1"=centre, "2"=left, "3"=topleft, "4"=topright, "5"=topcentre, "6"=bottomleft, "7"=bottomright, "8"=bottomcentre		
language	"%s"		
lineFactor	"%.16g"		
nameComponents.displayName	"%s"		
nameComponents.name	"%s"		
originAnchor	String representing enumeration such that: "0"=topleft, "1"=topcentre, "2"=topright, "3"=middleleft, "4"=middlecentre, "5"=middleright, "6"=bottomleft, "7"=bottomcentre, "8"=bottomright		
panZoom	"%d"		
spaceFactor	"%.16g"		
stackFactor	"%.16g"		
strikethrough	String representing enumeration such that: "0"=none, "1"=single,		

	"2"=double		
subscript	String representing enumeration such that: "0"=sub, "1"=super		
symbolColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textAspect	"%.16g"		
textColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
textDirection	"%.16g %.16g %.16g"		
textHeight	"%.16g"		
textLineWidth.*measurementType*	"%.16g"		*measurementType* must be replaced with meterWidth or pixelWidth
textLineWidth.name	"%s"		
textOrigin	"%.16g %.16g %.16g"		
textRotationPoint	"%.16g %.16g %.16g"		
textThickness	String representing enumeration such that: "0"=normal, "1"=thick, "2"=thin		
underline	String representing enumeration such that: "0"=over, "1"=under		
unit	"%s"		
uriRefs[%d]	"%s"		
usage	"%s"		
valid	String representing enumeration such that: "0"=true, "1"=false, "2"=unknown		
PMIGeometryWidth	%d		Unsigned decimal integer representing text size in units of pixels
PMITextShowBorder	Encoded enumeration such that “0” = False “1” = True		
PMITextSize	%d		Unsigned decimal integer representing text size in units of pixels
PMITextInPlane	Encoded enumeration such that “0” = False “1” = True		

N.39 PMIDataModelView

Table 243 — PMIDataModelView

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d]	%d;%d;%s;%d;%s;%d;%d;%d;%d;%s;%d;%s; Such that the formatted value encodes a general attribute by: 1;<lengthofname>;<name>;<length of value>;<value>;<DataType>;<MeasureEnum>;<editable?>;<lengthofunit>;<unit>;<lengthofcategory>;<category>; Where name, value, unit and category are strings editable idicates if the attribute is editable DataType is on of the following values: "0"=int, "1"=ints, "2"=real, "3"=reals, "4"=boolean, "5"=booleans, "6"=string, "7"=reference, "8"=enum, "9"=list, "10"=datetime and MeasureEnum is on of the following values: "-1"=none, "0"=length, "1"=area, "2"=volume, "3"=mass, "4"=massdensity, "5"=fatiguestrengthcoefficient, "6"=time, "7"=angle, "8"=velocity, "9"=acceleration, "10"=force, "11"=forceperunitlength, "12"=pressure, "13"=moment, "14"=stress, "15"=strain, "16"=strainenergy,		It is expected that you will also have an additional property: key = <name> and value = <value> which is not hidden to go with this property which should be hidden.

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
	"17"=strainenergydensity, "18"=temperature, "19"=heatflux, "20"=convectioncoefficient, "21"=thermalconductivity, "22"=thermalexpansioncoefficient, "23"=specificheat, "24"=angularvelocity, "25"=angularacceleration, "26"=fatiguelife, "27"=heatflowrate, "28"=thermalenergy, "29"=massmomentofinertia, "30"=dynamicviscosity, "31"=heatgeneration, "32"=thermalconductance, "33"=conductanceperunitlength, "34"=thermalresistance, "35"=massflowrate, "36"=volumeflowrate, "37"=temperaturedifference, "38"=frequency, "39"=coefficientperunitlength, "40"=momentofinertiaarea, "41"=viscousdamping, "42"=energy, "43"=power, "44"=momentum, "45"=temperaturegradient, "46"=energyperunitmass, "47"=dissipationrateofenergyperunitmass, "48"=massflux, "49"=massperunitlength, "50"=massperunitarea, "51"=electriccurrent, "52"=electricalresistance, "53"=electricalresistivity, "54"=electricalconductance, "55"=voltage, "56"=voltagepertemperature, "57"=diffusivity, "58"=latentheatpermass, "59"=thermalenergyperarea, "60"=thermalpidgain, "61"=thermalpidintegraltermconstant, "62"=thermalpidderivativetermconstant, "63"=headlosscoefficient, "64"=tsaiwucoefficient, "65"=masslength, "66"=pervolume, "67"=warpingconstant, "68"=stresscompliance, "69"=lengthperunitpressure, "70"=pressureperunitlength, "71"=pressureperunitvelocity, "72"=momentperangle, "73"=coefficientperunittime, "74"=angularmomentumperunitangle, "75"=thermalcapacitance, "76"=inductance, "77"=voltageperangularvelocity, "78"=perarea, "79"=temperaturechangerate, "80"=jerk, "81"=angularjerk, "82"=magneticfieldstrength, "83"=magneticfluxdensity		

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
GeneralAttribute[%d].ListValue[%d]	"%s"	GeneralAttribute[%d]	
ModelViewBehavior	Encoded string that is a logical combination of the desired behaviours: ApplyCamera=0x0001, ApplyPartVisibility=0x0002, ApplyPMIVisibility=0x0004, HideOtherParts=0x0008, ShowOtherParts=0x0010, HideOtherPMI=0x0200		
ModelViewCameraType	String representing enumeration such that: "0"=perspective, "1"=orthographic		
bottomLeftBackgroundColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
bottomRightBackgroundColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
edgeColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
exploded	"%d"		
modelViewStyle	String representing enumeration such that: "0"=shaded, "1"=shadedwithedges, "2"=hidden, "3"=hiddenmarked, "4"=wireframe, "5"=transparent		
shadingStyle	String representing enumeration such that: "0"=flat, "1"=smooth		
topLeftBackgroundColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		
topRightBackgroundColour	"0x00%02x%02x%02x" such that the values of blue, green and red for the colour are encoded in the string		

N.40 PART TRANSFORM (0x030C)

Table 244 — PART TRANSFORM

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
TRANSFORMATION_MATRIX	“%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f”		Also supported for EntityType 0x0114 if Type Name is “Part Transform”

N.41 MODEL VIEW STYLE (0x0310)

Table 245 — MODEL VIEW STYLE

“Key” Property Atom Value String	“Value” Property Atom Value String Encoding Format	Prequist Key	AdditionalNote
Style	“Shaded” “ShadedWithEdges” “Hidden” “HiddenMarked “Wireframe” “Transparent”		

Bibliography

[18] OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. OpenGL Programming Guide : The official guide to learning OpenGL Version 2, Fifth Edition. Addison-Wesley 2005.

[19] Michael Deering. Geometry Compression, Computer Graphics, Proceedings SIGGRAPH '95. August 1995, pp. 13-20.

[20] GNU C Library. Floating-Point Conversions. Available from World Wide Web: http://www.gnu.org/software/hello/manual/libc/Floating_002dPoint-Conversions.html#Floating_002dPoint-Conversions

[21] Michael Deering, Craig Gotsman, Stefan Gumhold, Jarek Rossignac, and Gabriel Taubin. 3D Geometry Compression Course Notes for SIGGRAPH 2000, July 25, 2000.

[22] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[23] Les Piegl and Wayne Tiller, *The NURBS Book,* Springer-Verlag, 1997.

[24] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder*, Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes*, Graphical Models, Vol. 64, No. 3-4, Pages: 147 - 168, 2002.

[25] Greg Roelofs, Mark Adler, Jean-loup Gailly. zLib compression library. Available from World Wide Web: http://www.zlib.net/

[26] *JT Open Program* (http://www.jtopen.com) --- A program to help members leverage the benefits of open collaboration across the extended enterprise through the adoption of the JT format, a technology that makes it possible to view and share product information throughout the product lifecycle. Membership in the JT Open Program provides access to the JT Open Toolkit library, which among other things, provides read and write access to JT data and enforces certain JT conventions to ensure data compatibility with other JT-enabled applications.

[27] *JT2Go download* (http://www.jt2go.com) --- JT2Go is the no-charge 3D JT viewer from Siemens. JT2Go puts 3D data at your fingertips by allowing anyone to download the no-charge viewer. JT2Go also allows anyone to embed 3D JT data directly into Microsoft Office documents. JT2Go offers full 3D interactivity on parts, assemblies, and even 2D drawings (CGM & TIF).

[28] *Siemens: PLM Components: Parasolid: XT Pipeline* (http://www.ugs.com/products/open/parasolid/pipeline.shtml) --- This web page provides information on the Parasolid precise boundary representation format (XT) and how this XT format fits within the Siemens vision of seamless exchange of digital product models across enterprises, between different disciplines, using their PLM applications of choice.

[29] *OpenGL Programming Guide : The official guide to learning OpenGL Version 2*, Fifth Edition, by OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis (Addison-Wesley 2005) --- This book gives in-depth explanation of the OpenGL Specification and will provide further insight into the significance of some of the data (e.g. Materials, Textures) that can exist in a JT file. Information in this book may also serve as a guide for how one could process the data contained in a JT file to produce/render an image on the screen.

[30] Michael Deering, *Geometry Compression*, Computer Graphics, Proceedings SIGGRAPH '95, August 1995, pp. 13-20.

[31] Michael Deering, Craig Gotsman, Stefan Gumhold, Jarek Rossignac, and Gabriel Taubin, *3D Geometry Compression*, Course Notes for SIGGRAPH 2000, July 25, 2000.

[32] *OpenGL Shading Language Specification* (http://www.opengl.org/documentation/glsl/) --- OpenGL Shading Language (GLSL) as defined by the OpenGL Architectural Review Board, the governing body of OpenGL.

[33] K. Weiler. *Topological Structures for Geometric Modeling*, PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1986.

[34] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1989.

[35] Les Piegl and Wayne Tiller, *The NURBS Book*, Springer-Verlag, 1997.

[36] *Planetmath.org - Huffman Coding* (http://planetmath.org/encyclopedia/HuffmanCoding.html) --- This web page provides a technical overview of Huffman coding which is one form of data encoding used within the JT format.

[37] Michael Schindler, *Practical Huffman Coding* (http://www.compressconsult.com/huffman/#encoding) --- This web page provides some coding hints for implementing Huffman coding which is one form of data encoding used within the JT format.

[38] Glen G. Langdon Jr., *An Introduction to Arithmetic Coding*, IBM Journal of Research and Development, Volume 28, Number 2, March 1984, pp. 135-149.

[39] Paul G. Howard and Jeffrey Scott Vitter, *Practical Implementation of Arithmetic Coding. Image and Text Compression*, ed. J. A. Storer, Kluwer Academic Publishers, April 1992, pp. 85-112.

[40] zlib.net (http://www.zlib.net/) --- This web page provides (either directly or through links) complete detailed information on ZLIB compression including frequently asked questions, technical documentation, source code downloads, etc.

[41] Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder, *Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes*, Graphical Models, Vol. 64, No. 3-4, Pages: 147 - 168, 2002.