



DPC: 25/30494217 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 02 January 2025

Origin: International

Latest date for receipt of comments: 25 February 2025

Project No. 2024/01629

Responsible committee: AMT/4 Industrial data and manufacturing interfaces

Interested committees:

Title: Draft BS ISO 14306-4 Industrial automation systems and integration - JT file format specification for 3D visualization. Part 4: Version 3.

Please notify admin.start@bsigroup.com if you are aware of any keywords that might assist in clarifying or identifying the standard or of the content of this standard:

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD. THIS DRAFT IS NOT CURRENT BEYOND 25 February 2025

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <https://standardsdevelopment.bsigroup.com/>.

E-mail: admin.start@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/ Figure/Table/Note	Type of comment	Comment (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



DRAFT International Standard

ISO/DIS 14306-4

Industrial automation systems and integration — JT file format specification for 3D visualization —

Part 4: Version 3

ICS: 35.240.50

ISO/TC 184/SC 4

Secretariat: **ANSI**

Voting begins on:
2025-01-02

Voting terminates on:
2025-03-27

This document is circulated as received from the committee secretariat.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

Reference number
ISO/DIS 14306-4:2025(en)

Full document details: 25/30494217 DC

© ISO 2025  **Documents**
Document Sharing Hub



COPYRIGHT PROTECTED DOCUMENT

© ISO 2025

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	vii
Introduction	ix
1 Scope	1
2 Normative references	1
3 Terms and definitions, abbreviated terms	2
3.1 Terms and definitions	2
3.2 Abbreviated terms	2
4 Notational conventions	2
4.1 Diagrams and field descriptions	2
4.2 Data Types	6
4.3 Empty field	9
5 File Format	9
5.1 General	9
5.2 Object Types	9
5.3 File Structure	9
5.3.1 File Header	10
5.3.2 TOC Segment	11
5.3.3 Data Segment	12
5.4 Data Segments	18
6 LSG Segment	18
6.1 Segment Overview	18
6.2 Graph Elements	19
6.3 Node Elements	19
6.3.1 Base Node Element	19
6.3.2 Base Node Data	20
6.3.3 Partition Node Element	21
6.3.4 Group Node Element	23
6.3.5 Instance Node Element	24
6.3.6 Part Node Element	25
6.3.7 Meta Data Node Element	25
6.3.8 LOD Node Element	26
6.3.9 Range LOD Node Element	27
6.3.10 Switch Node Element	28
6.3.11 Base Shape Node Element	29
6.3.12 Vertex Shape Node Element	32
6.3.13 Tri-Strip Set Shape Node Element	33
6.3.14 Polyline Set Shape Node Element	33
6.3.15 Point Set Shape Node Element	34
6.3.16 Polygon Set Shape Node Element	35
6.3.17 NULL Shape Node Element	35
6.3.18 Primitive Set Shape Node Element	36
6.4 LSG Attribute Elements	38
6.4.1 Material Attribute Element	40
6.4.2 Texture Image Attribute Element	43
6.4.3 Draw Style Attribute Element	55
6.4.4 Light Set Attribute Element	57
6.4.5 Linestyle Attribute Element	58
6.4.6 Pointstyle Attribute Element	60
6.4.7 Geometric Transform Attribute Element	61
6.4.8 Palette Map Attribute Element	63
6.4.9 Infinite Light Attribute Element	64
6.4.10 Point Light Attribute Element	66
6.5 Property Atom Elements	69

6.5.1	Base Property Atom Element.....	69
6.5.2	String Property Atom Element.....	70
6.5.3	Integer Property Atom Element.....	71
6.5.4	Floating Point Property Atom Element.....	71
6.5.5	JT Object Reference Property Atom Element.....	72
6.5.6	Date Property Atom Element.....	73
6.5.7	Late Loaded Property Atom Element.....	74
6.5.8	Vector4f Property Atom Element.....	75
6.6	Property Table	75
7	Shape LOD Segment.....	77
7.1	Shape LOD Segment Overview.....	77
7.1.1	Tri-Strip Set Shape LOD Element	78
7.1.2	Polyline Set Shape LOD Element.....	78
7.1.3	Point Set Shape LOD Element.....	79
7.1.4	Polygon Set LOD Element.....	79
7.1.5	Null Shape LOD Element.....	92
7.1.6	Primitive Set Shape Element	92
8	Geometry Segments.....	99
8.1	Geometry Segments Overview.....	99
8.2	XT B-Rep Element	99
8.3	STEP B-Rep Segment.....	100
8.4	Wireframe Segment.....	100
9	Meta Data Segment.....	100
9.1	Meta Data Segment Overview.....	100
9.2	Property Proxy Meta Data Element.....	101
9.3	PMI Manager Meta Data Element.....	103
9.3.1	PMI Design Group Entities.....	105
9.3.2	PMI Associations	107
9.3.3	PMI User Attributes	109
9.3.4	PMI String Table	110
9.3.5	PMI Model Views	111
9.3.6	Generic PMI Entities.....	114
9.3.7	PMI CAD Tag Data.....	125
9.3.8	PMI Polygon Data.....	126
9.3.9	PMI Model View Sort Orders.....	129
9.3.10	PMI Association Properties	130
9.3.11	Generic PMI Additions	131
10	Info Segment.....	134
11	Data Compression and Encoding.....	135
11.1	Data Compression and Encoding Overview.....	135
11.2	Common Compression Data Collection Formats.....	136
11.2.1	Int32 Compressed Data Packet.....	136
11.2.2	Int64 Compressed Data Packet.....	141
11.2.3	Compressed Vertex Coordinate Array	144
11.2.4	Compressed Vertex Normal Array.....	145
11.2.5	Compressed Vertex Texture Coordinate Array.....	147
11.2.6	Compressed Vertex Colour Array.....	148
11.2.7	Compressed Vertex Flag Array.....	150
11.2.8	Compressed Auxiliary Fields Array	150
11.2.9	Point Quantizer Data.....	154
11.2.10	Texture Quantizer Data.....	154
11.2.11	Colour Quantizer Data.....	155
11.2.12	Uniform Quantizer Data.....	157
11.2.13	Compressed Entity List for Non-Trivial Knot Vector.....	157
11.2.14	Compressed Control Point Weights Data.....	160
11.2.15	Compressed Curve Data.....	161

11.2.16	Compressed CAD Tag Data	164
11.3	Encoding Algorithms	165
11.3.1	Uniform Data Quantization	165
11.3.2	Bitlength CODEC	165
11.3.3	Arithmetic CODEC	166
11.3.4	Deering Normal CODEC	171
11.4	LZMA compression	173
12	Conformance requirements	173
12.1	General Conformance	173
12.2	Conformance Classes	174
12.2.1	Approximated Geometry Visualization (CC01)	174
12.2.2	Precise Geometry Visualization (CC02)	174
13	Common Data Conventions and Constructs	174
13.1	Overview	174
13.2	Late-Loading Data	174
13.3	TOC Segment Location	175
13.4	Bit Fields	175
13.5	Empty Field	175
13.6	Hash Value	175
13.7	Scene graph construction	175
13.8	Metadata Conventions	176
13.8.1	Property Key Naming Conventions	176
13.8.2	PMI Properties	177
13.8.3	CAD Properties	178
13.8.4	Tessellation Properties	180
13.8.5	Miscellaneous Properties	181
13.8.6	The SUBNODE property and Reference Sets	182
13.8.7	Material and Surface Designation Properties	186
13.9	LSG Attribute Accumulation Semantics	186
13.10	LSG Part Structure	187
13.11	Range LOD Node Alternative Rep Selection	187
13.12	B-Rep Face Group Associations	188
13.13	Smart Topology Table (STT) Segment	188
13.14	Watermark Image	189
13.15	State Flags	189
Annex A	(normative) Information object registration	190
Annex B	(normative) Object Type Identifiers	191
Annex C	(informative) Coding Algorithms – An Implementation	193
Annex D	(informative) Hashing – An Implementation	216
Annex E	(informative) Polygon Mesh Topology Coder	219
Annex F	(normative) Per Face Group Attributes	237
Annex G	(normative) STEP B-Rep	241
Annex H	(normative) STEP schema	244
Annex I	(informative) XT B-Rep data segment	312
Annex J	(informative) Smart Topology Table (STT) Segment	414
Annex K	(normative) Wireframe Segment	441
Annex L	(informative) Mapping table 14306-4 XT B-Rep to STEP	444
Annex M	(informative) Procedural Geometry – Evaluation and Approximation	450
Annex N	(normative) Information Object Registration Scheme	485
Annex O	(informative) Change History	487

Bibliography	490
---------------------------	------------

Foreword

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

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

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO [had/had not] received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

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

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

This document was prepared by Technical Committee ISO/TC 184, Automation systems and integration, Subcommittee SC 4, Industrial data.

The ISO 14306-4 Version 3 specification defines the requirements based upon the JT 10.5 file format requirements. The ISO 14306:2017 Edition 2 defines the requirements based upon the JT 9.5 file format requirements.

The main changes between the ISO 14306-4 Version 3 specification and the previous ISO 14306:2017 Edition 2 specification are as follows:

- The version string in the File Header has been updated to reflect ISO 14306-4 Version 3.
- The ISO 14306-4 Version 3 file compression format has changed from Z-Lib to the publicly available LZMA format.
- The technical description for MbStrings Composite Data Types has been updated to include UTF 16
- New segment types have been added for Smart Topology Table (STT) and Info Segment
- The Base Attribute Data description has been expanded to include the logical collection Base Attribute Data Fields V2. This logical collection is used to support per face group attributes.
- A new attribute element has been added to the LSG definition, 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, “Sabot Attribute Element”, has been added to the LSG definition. Sabot is used to insulate pre ISO 14306-4 Version 3 readers from attributes with non-fallback palette Index attributes in order to preserve forward compatibility.
- New PMI Association values have been included to support partial scene sectioning.
- The Table “Common Property Keys and Their Value Encoding formats” has been removed from the PMI Property descriptions in section 9.2.5 PMI Model Views.

- A complete description of all PMI String Property Atom Element values is made available as a Reference Data Library in electronic format.
- Generic PMI Entity Type values has a new value; 0x0309 Weld Note Type
- An Annex for 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)
- Two new PMI Associations Reason Code values have been added
256 -Association is to say cut the selected content that doesn't exist within the regular scene graph (such as a targeted PMI Display) within the given section
257 -Association is to identify that the PMI source content should be part of the MV (Model View) destination with its visibility toggled to off.
- The following changes have been made in the XTB-RepDataSegmentDescription section of the XTB-Rep Annex
Updates in the "Topology" description; addition of compound bodies, child bodies and standard bodies in the topology representation list and descriptions for the new body types.
Updates in the "World Topologies table" for the field name - "body" description
Updates to the "Body fields" table for the descriptions of the "Field names"; highest_node_id, attribute_groups, attribute chains, body type, region, edge, and vertex.
Updates to the "Region fields" table for the descriptions of "Field names"; "body"

The following minor editorial documentation changes, that do not impact the technical content of the standard, have also been included with this release.

- The Text Polyline Data Figure titled "Constructing Text Polylines from data arrays" has been updated to more clearly represent the description of this data.
- The Non-Text Polyline Data Figure titled "Constructing Non-Text Polylines from packed 2D data arrays" has been updated to more clearly represent the description of this data.

A list of all parts in the ISO 14306 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO 14306 format is an industry focused, high-performance, lightweight, flexible file format for capturing and repurposing 3D product definition data for visualization to enable collaboration and validation throughout the extended enterprise. The ISO 14306 format is streamable and contains compression for compact and efficient representation.

Some of the highlights of the ISO 14306 format include:

- built-in support for assemblies, sub-assemblies and part constructs;
- a flexible partitioning scheme, supporting single or multiple files;
- b-rep solid shape representations to provide precision to the light-weight viewing processes;
- product manufacturing information in support of paperless manufacturing initiatives;
- precise and imprecise wireframe shape representations;
- discrete purpose-built levels of detail;
- triangle sets, polygon sets, point sets, line sets and implicit primitive sets (such as cylinder, cone and sphere);
- a full array of visual attributes such as for materials, textures, lights;
- hierarchical bounding box and bounding spheres;
- data compression that allows producers of ISO 14306 files to fine tune the trade-off between compression ratio and fidelity of the data.

Beyond the data contents description of the ISO 14306 format, the overall physical structure/organization of the format is also designed to support operations such as:

- offline optimizations of the data contents, therefore file granularity and flexibility optimized to meet the needs of enterprise data translation solutions;
- asynchronous streaming of content, therefore viewing optimizations such as view frustum and
- occlusion culling and fixed-framerate display modes;
- layers, and layer filters.

NOTE This document is based on the JT Open version 10.5 specification, which defines a logical scene graph with geometry specific node and attribute support, extended to support ISO 10303.

[Annex A](#) contains an identifier that conforms to ISO/IEC 8824-1. The identifier unambiguously identifies this document in an open information system.

The International Organization for Standardization (ISO) draws attention to the fact that it is claimed that compliance with this document may involve the use of patents USA 20110199382, 8019788.

ISO takes no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO that he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO.

Information may be obtained from:

Siemens PLM Software Product Development
345 Woodcliff Drive

Fairport

NY 14450

USA

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO shall not be held responsible for identifying any or all such patent rights.

[Annex A](#) contains an identifier that conforms to ISO/IEC 8824-1. The identifier unambiguously identifies this document and the schema in an open information system. The ISO 14306 series applies the method as described in [Annex N](#).

Industrial automation systems and integration — JT file format specification for 3D visualization —

Part 4: Version 3

1 Scope

This document defines the syntax and semantics of a file format for the 3D visualization and interrogation of lightweight geometry and product manufacturing information derived from CAD systems, using visualization software tools that do not need the full capability of a CAD system.

This document has been adopted as a 3D visualization capability in addition to the ISO 10303 series.

The ISO 10303 series are the ISO standards adopted for the engineering data exchange, sharing and long-term archiving of product definition information throughout the product lifecycle.

In this document 3D visualization is defined as the visual presentation on a screen or another media of graphical and textual 3-dimensional representations of a set of data representing an object, information or results of a computational process in order to enable decision process by a human looking at the data visualized in a medium.

The ISO 14306 file format specification for 3D visualization includes data descriptions that can represent the following data.

- facet information (triangles), stored with geometry compression techniques;
- visual attributes such as lights, textures and materials;
- product manufacturing information (PMI);
- boundary representation (b-rep) solid model shape representation and associated metadata;
- configuration representations; and
- delivery methods such as asynchronous streaming of content.

The ISO 14306-4 Version 3 file format specification for 3D visualization does not specify the implementation of, or definition of a run-time architecture for viewing and/or processing ISO 14306 data.

2 Normative references

ISO 10303-21:2014, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure*

ISO 10303-42:2014, *Industrial automation systems and integration – Product data representation and exchange – Part 42: Integrated generic resource: geometric and topological representation*

ISO 14306-1:2023, *Industrial automation systems and integration - ISO 14306 Edition 3 file format specification for 3D product data visualization. Part 1: Overview and fundamental principles*

Reference Data Library - ISO14306_PMIPropertiesStrings.xlsx (electronic insert).

Reference Data Library – ISO14306_MaterialSurfaceProperties.xlsx (electronic insert).

3 Terms and definitions, abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 14306-2 apply.

3.2 Abbreviated terms

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

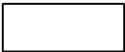
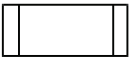
Abs	Absolute Value
Bbox	Bounding Box
B-Rep	Boundary Representation
CAD	Computer Aided Design
CODEC	Coder-Decoder
GD&T	Geometric Dimensioning and Tolerancing
GUID	Globally Unique Identifier
HSV	Hue, Saturation, Value
LsbFirst	Least Significant Byte First
Max	Maximum
Min	Minimum
MsbFirst	Most Significant 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

4 Notational conventions

4.1 Diagrams and field descriptions

Symbolic diagrams are used to describe the structure of the ISO 14306 file. The symbols used in these diagrams are shown in [Table 1](#) and 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” (therefore 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 shown in [Figure 1](#) the Data_Type is “I32” (a signed 32 bit integer) and Field_Name is “Count.”

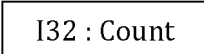


Figure 1 — rectangle box diagram

Diagram “folder” (therefore logical data collections) symbols are simply titled with a collection name. In the example shown in [Figure 2](#) the collection name is “Graph Elements.”

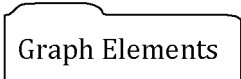


Figure 2 — folder diagram

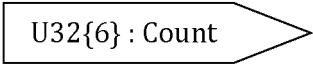
Diagram “rectangle box with lines at left and right sides” are simply titled with a logic step name. In the example shown in [Figure 3](#) the logic step name is “Recover First Shell Indices”.



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

Diagram “rectangle box with clipped right side corners” (therefore 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 (for example 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 shown in [Figure 4](#) the Data Type is “U32”, “6” bits are used to store the value, and Field Name is “Count”



U32{6} : Count

Figure 4 — rectangle box with clipped right side corners

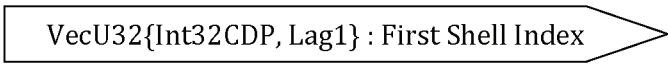
Data Type followed by open brace "{", compressed data packet type, ",", Predictor Type, closed brace "}", and a colon ":"; followed by the field name. This format for titling the diagram indicates that a vector of "Data Type" data (therefore *primal* values) is ran through "Predictor Type" algorithm 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 (therefore 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 ISO 14306 Edition 2 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 shown in [Figure 5](#) the Data Type is "VecU32", Int32 Compressed Data Packet type is used, Lag1 Predictor Type is used, and Field Name is "First Shell Index."



VecU32{Int32CDP, Lag1} : 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 (therefore difference from the predicted value), and this resulting output array of *residual* values is the data which is actually encoded/compressed. The ISO 14306-4 Version 3 format supports several Predictor Type algorithms and each use of Int32CDP specifies, using the notation format described above, what Predictor Type algorithm is being used on the data. The supported Predictor Type algorithms are as shown in [Table 2](#) (note that a sample implementation of decoding the predictor *residual* values back into the primal values can be found in the Coding Algorithms Annex.

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 shown in [Figure 6](#) the Data_Type is "VecU32" and Field_Name is "CodeText."

VecU32 : CodeText

Figure 6 — data type : field name diagram

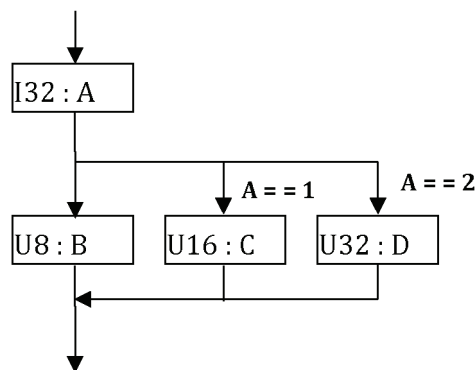
Note that for some file Segment Types there is LZMA compression also applied to all bytes of element data stored in the segment. This LZMA compression applied to all the segment's data is not indicated in the diagrams through the use of "rectangle box with clipped right side corners". Instead, one shall 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 14306-4 Version 3 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 shall be set to "0".

For folders (therefore 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, shown in [Figure 7](#), shows data field A followed by (depending on value of A) either data field B, C, or D.

**Figure 7 — data field 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 (for example 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 shown in [Figure 8](#):

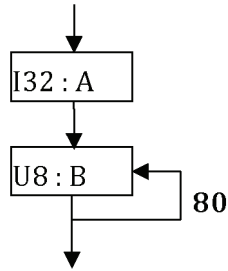


Figure 8 — loop construct example

The second form of this loop construct is used when the number of iterations is based on data (for example count) previously read from the file. In this case it is valid for there to be zero data iterations (zero count). This second form of the loop construct looks as shown in [Figure 9](#) (data field D is repeated C value times).

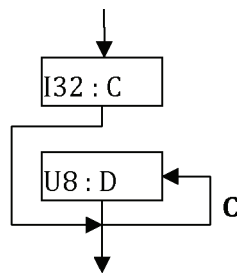


Figure 9 — loop construct with iterations example

4.2 Data Types

The data types that can occur in the ISO 14306-4 Version 3 binary files are listed in the following two tables.

The Basic Data Types table, shown in [Table 3](#), lists the basic/standard data types which can occur.

Table 3 — Basic Data Types

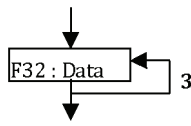
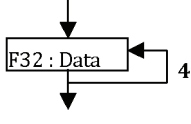
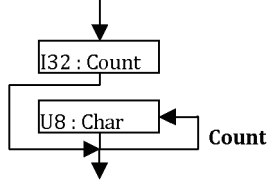
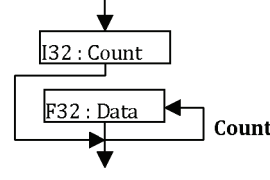
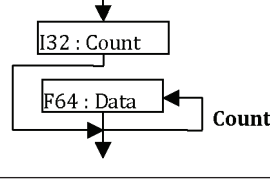
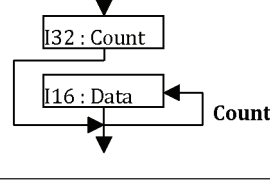
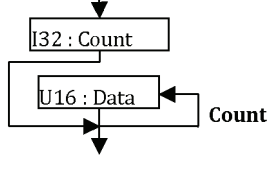
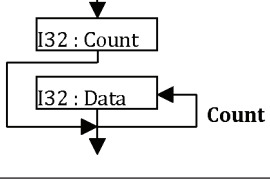
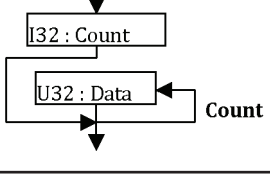
Type	Description
UChar	An unsigned 8-bit byte.
U8	An unsigned 8-bit integer value.
U16	An unsigned 16-bit integer value.
U32	An unsigned 32-bit integer value.
U64	An unsigned 64-bit integer value.
I16	A signed two's complement 16-bit integer value.
I32	A signed two's complement 32-bit integer value.
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 Types, as shown in [Table 4](#), lists some composite data types which are used to represent some frequently occurring groupings of the basic data types (for example Vector, RGBA colour). The composite data types are defined in this reference simply for convenience/brevity in describing the ISO 14306-4 Version 3 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 ISO 14306 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 ISO 14306 format GUIDs are used as unique identifiers (for example 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" (therefore 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 ($a + bi + cj + dk$) 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.	

Table 4 (continued)

Type	Description	Symbolic Diagram
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" (therefore 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.	

4.3 Empty field

When writing a ISO 14306 file whose data did not originate from reading a previous ISO 14306 file, an empty field shall be set to a value of “0”.

When writing a ISO 14306 file whose data originated from reading a previous ISO 14306 file (therefore rewriting a ISO 14306 file), empty fields should be written with the same value that was read from the originating ISO 14306 file.

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

5 File Format

5.1 General

This document shall be unambiguously identified in an open information system by the code defined in [Annex A](#).

5.2 Object Types

All objects represented in the format are assigned an “object identifier” (for example see Base Node Data, or Base Attribute Data Base Attribute Data) and all references from one object to another object are represented in the format using the referenced object’s “object identifier”. It is the responsibility of format readers/writers to maintain the integrity of these object references by doing appropriate pointer unswizzling/swizzling as 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 (therefore replacing references based on memory pointers with object identifier references).

All objects stored in an ISO 14306 Version 3 file are classified by type and thus include an object type identifier as part of their persisted data.

Object Type Identifiers specifications are defined in Object Type Identifiers [Annex B](#) of this document.

5.3 File Structure

An ISO 14306 file is structured as a sequence of blocks/segments, as shown in [Figure 10](#). 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 ISO 14306 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.

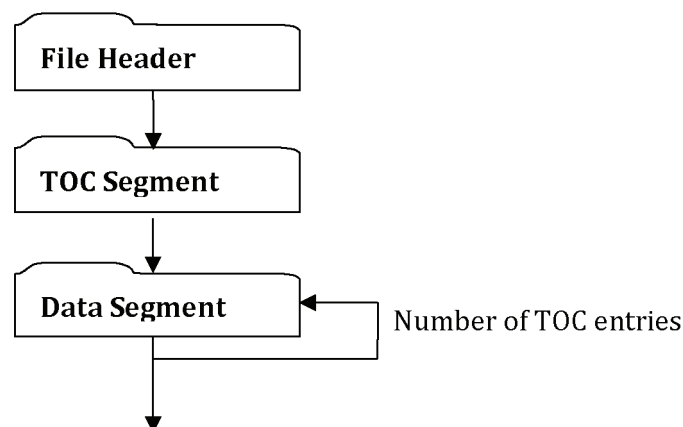


Figure 10 — ISO 14306 File Structure

5.3.1 File Header

The File Header is always the first block of data in an ISO 14306 file. The File Header contains information about the ISO 14306 file version and TOC location, which Loaders use to determine how to read the file. The exact contents of the File Header are as shown in [Figure 11](#):

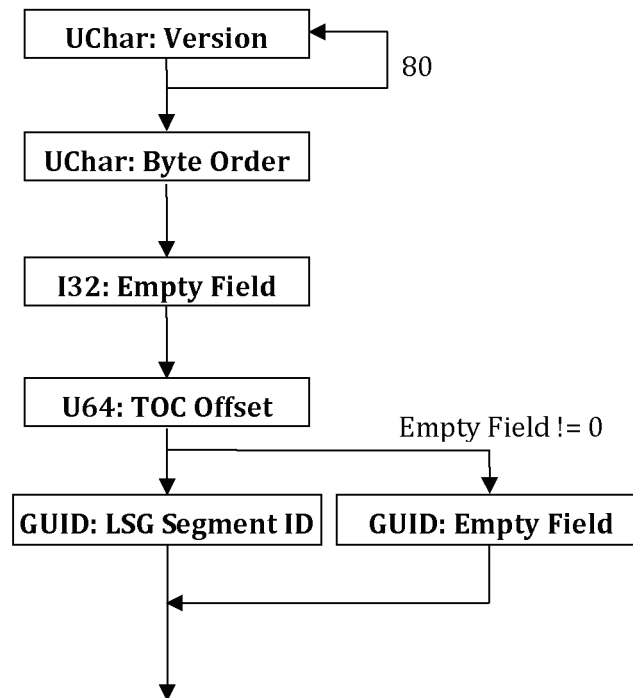


Figure 11 — File Header data collection

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 10.5 Optional Comment ISO14306-4 Optional Comment

The version description for ISO 14306-4 Version 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 ISO 14306 file readers to validate that the ISO 14306 files has not been corrupted by ASCII mode FTP transfers.

The initial 12 characters of the header will be **Version 10.5** followed by a whitespace character. The string **ISO14306-4** will be written anywhere in the remaining header comment. The remaining 50 characters are available for additional comments to be written. These could be either before or after the **ISO14306-4** string.

As an example, for an ISO 14306-4 edition three file the header string will appear as follows as a minimum:

“Version 10.5 ISO14306-4 \n\r\n “

Alternatively, for example any the following header strings would also be valid:

“Version 10.5 foo ISO14306-4 bar \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.

5.3.2 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 ISO 14306 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, as shown in [Figure 12](#).

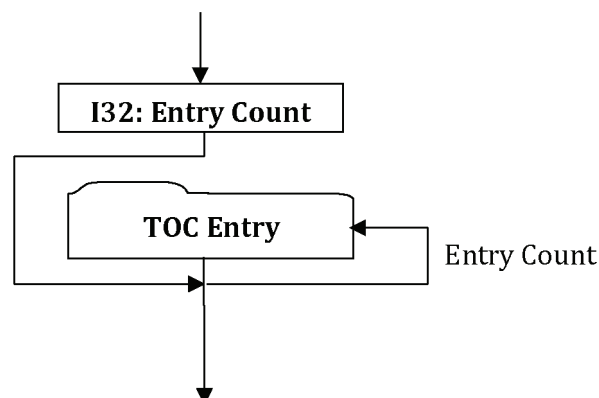


Figure 12 — TOC Segment data collection

I32: Entry Count

Entry Count is the number of entries in the TOC.

TOC Entry

Each TOC Entry represents a Data Segment within the ISO 14306 file. The essential function of a TOC Entry is to map a Segment ID to an absolute byte offset within the file, as shown in [Figure 13](#).

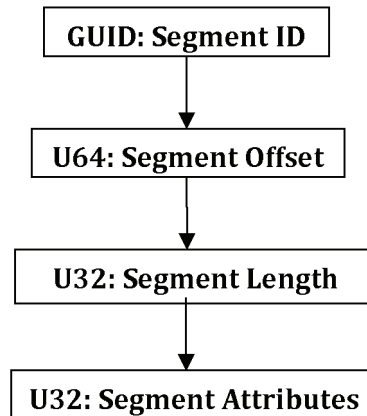


Figure 13 — TOC Entry data collection

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, as shown in [Table 5](#).

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 the Segment Types table of this document.

5.3.3 Data Segment

All data stored in an ISO 14306-4 Version 3 file shall be defined within a Data Segment. Data Segments are “typed” based on the general classification of data they contain. See the 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, the Segment Types table has a column indicating whether the Segment Type may have compression applied to its Data bytes.

All Data Segments have the same basic structure, as shown in [Figure 14](#).

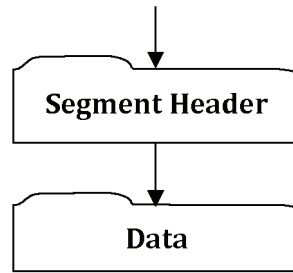


Figure 14 — Data Segment data collection

Segment Header

Segment Header contains information that determines how the remainder of the Segment is interpreted by the loader, as shown in [Figure 15](#).

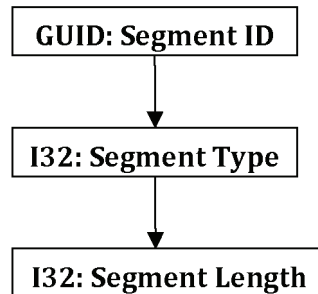


Figure 15 — Segment Header data collection

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 shown in [Table 6](#):

Table 6 — Segment Types

Type	Data Contents	Compression
1	Logical Scene Graph	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
12	Shape LOD5	No

Table 6 (continued)

Type	Data Contents	Compression
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
23	STT	Yes
30	MultiXT B-Rep	Yes
31	InfoSegment	Yes
32	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 (therefore 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 (therefore 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 (therefore 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 (therefore it is the size of the complete segment) and should be equal to the length value stored with this segment's TOC Entry.

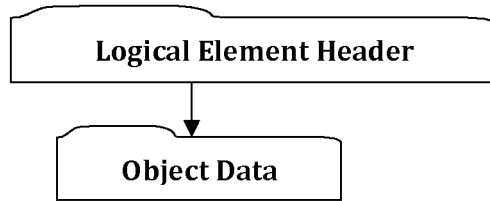
Data

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 an ISO 14306 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), shown in [Figure 16](#), 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 (therefore 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 is indicated Segment Types table. 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 ISO 14306 file compression/encoding techniques can be found in this document.

For Segment Types that do NOT support compression on all Segment Data.



For Segment Types that DO support compression on all Segment Data.

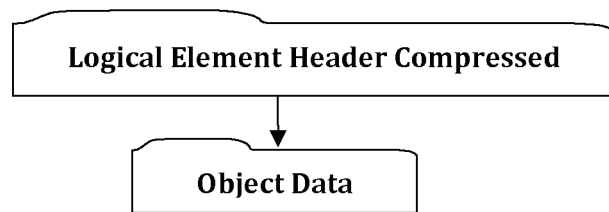


Figure 16 — Data collection

Logical Element Header

Logical Element Header contains data defining the length in bytes of the Element along with the Element Header, shown in [Figure 17](#).

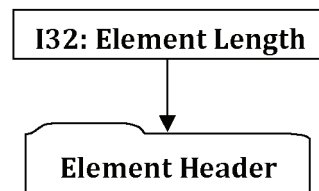


Figure 17 — Logical Element Header data collection

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, shown in [Figure 18](#).

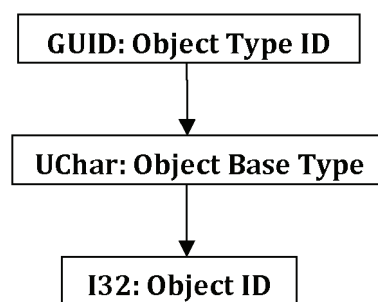


Figure 18 — Element Header data collection

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 an ISO 14306-4 Version 3 file can be found in [Annex B](#). If the GUID is not found in [Annex B](#), 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, shown in [Table 7](#), 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 the Logical Element Header description in this document).

Logical Element Header Compressed

Logical Element Header Compressed data collection, shown in [Figure 19](#), 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.

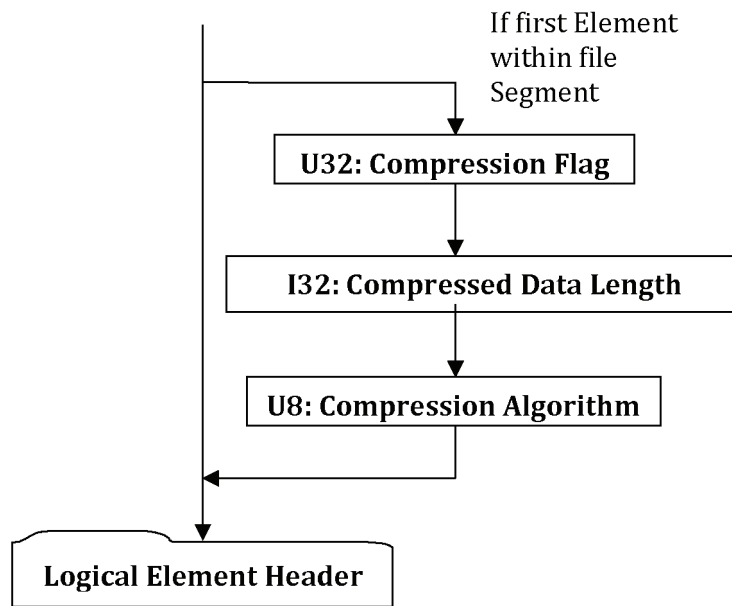


Figure 19 — Logical Element Header Compressed data collection

U32: Compression Flag

Compression Flag is a flag indicating whether compression is ON/OFF for all data elements in the file Segment. Valid values, shown in [Table 8](#), 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, shown in [Table 9](#), include the following:

Table 9 — Compression algorithm values

= 1	No compression
= 3	LZMA compression

Logical Element Header

See previous Logical Element Header description. 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.

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).

5.4 Data Segments

An ISO 14306 file consists of the following mandatory segments of data:

- LSG segment contains a collection of objects (therefore elements) connected through directed references to form a directed acyclic graph structure (therefore 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 (for example names, semantic roles) of those components, and a hierarchical structure expressing the component relationships.
- Shape LOD segment contains an element that defines the geometric shape definition data (for example vertices, polygons, normals, etc.) for a particular shape level of detail or alternative representation.

In addition to these mandatory segments a set of optional segments for geometry representation, PMI and meta data are available:

For geometry representation:

- STEP B-Rep segment contains an element that defines the precise geometric boundary representation data for a particular part in normative STEP Part 42 (ISO 10303-42) boundary representation format as a part 21 compressed file (ISO 10303-21).
- XT B-Rep Segment contains an element that defines the recommended precise geometric boundary representation data for a particular part in boundary representation format.
- 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.
- Wireframe Segment contains an element that defines the precise 3D wireframe data for a particular part.

For PMI and metadata:

- Info Segment: contains text strings with authoring information for file it exists in.
- Meta Data Segment is used to store large collections of meta-data in separate addressable segments of the ISO 14306 file, including PMI. Storing meta-data in a separate addressable segment allows references (from within the ISO 14306 file) to these segments to be constructed such that the meta-data can be late-loaded.
- STT Segment Smart Topology Table (hereafter referred to as STT) Segment contains an Element that defines the lightweight topology description for a particular Part.

For complete information on all the segments of an ISO 14306 file see the segment descriptions for each segment in their specific sections in this document.

6 LSG Segment

6.1 Segment Overview

LSG Segment contains a collection of objects (therefore Elements) connected through directed references to form a directed acyclic graph structure (therefore 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 (for example 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 (therefore 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, shown in [Figure 20](#).

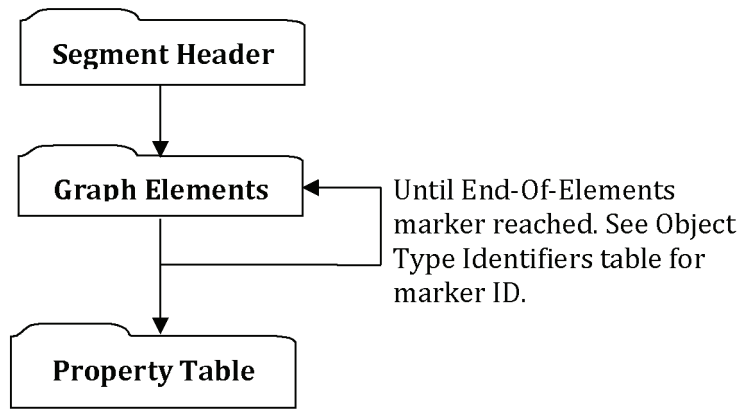


Figure 20 — LSG Segment data collection

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

6.2 Graph Elements

Graph Elements form the backbone of the LSG directed acyclic graph structure and in doing so serve as the ISO 14306 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 (for example Attribute Elements) that is inherited down the LSG to all descendant nodes.

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

Each of these general Graph Element classifications (therefore 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.

6.3 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.

6.3.1 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, as shown in [Figure 21](#), has no implied LSG semantic behaviour nor can it contain any children nodes.

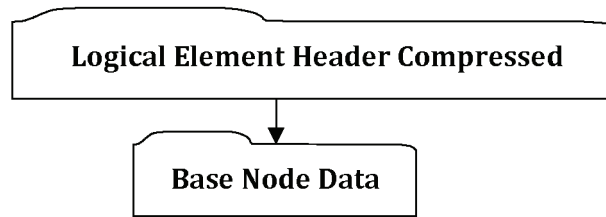


Figure 21 — Base Node Element data collection

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

6.3.2 Base Node Data

An example diagrammatic representation of Base Node Data is shown in [Figure 22](#).

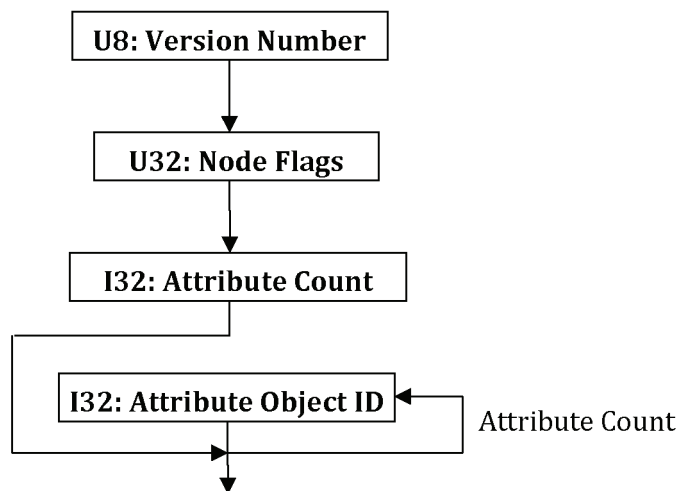


Figure 22 — Base Node Data collection

U8: Version Number

Version Number is the version identifier for this node. The value is set to 1.

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 shall be set to “0”, as shown in [Table 10](#).

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.

6.3.3 Partition Node Element

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

A partition in a ISO 14306 file shall always be either the root or leaf node. A leaf partition node represents an external ISO 14306 file reference and provides a means to partition a model into multiple physical ISO 14306 files (for example separate ISO 14306 file per part in an LSG assembly). An example diagrammatic representation of the Partition Node Element data collection is shown in [Figure 23](#).

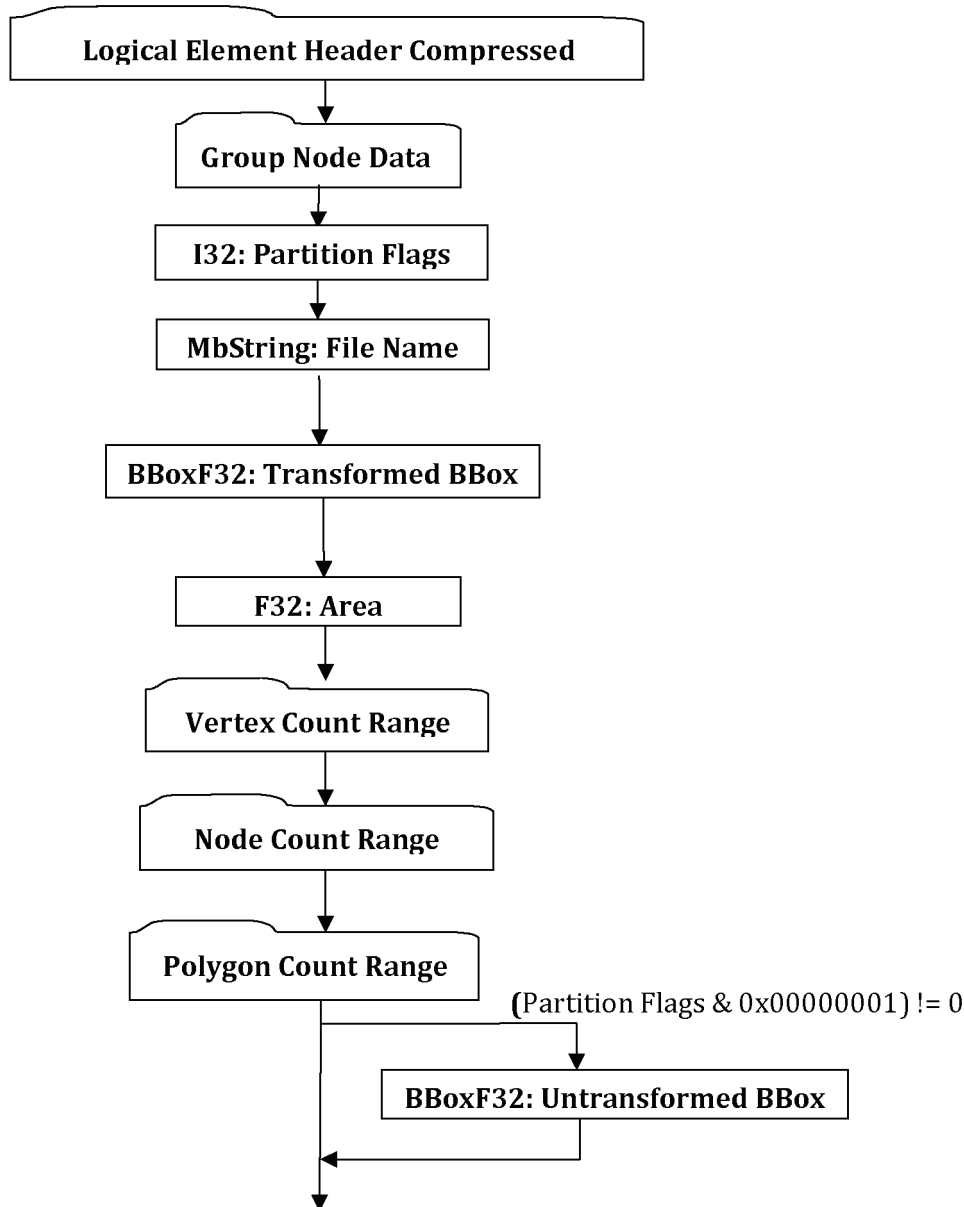


Figure 23 — Partition Node Element data collection

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, as shown in [Table 11](#), such as indicating the presence of optional data. All bits fields that are not defined as in use shall 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 ISO 14306 file relative to the location of the referencing ISO 14306 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 ISO 14306 data to determine whether to load the data contained within the Partition node (therefore 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 (therefore values scaled by MCS scaling).

Vertex Count Range

Vertex Count Range, as shown in [Figure 24](#), 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.

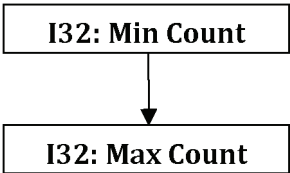


Figure 24 — Vertex Count Range data collection

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 ISO 14306 data to determine whether to load the data contained within the Partition node (therefore is any part of the bounding box within the view frustum).

6.3.4 Group Node Element

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

Group Nodes, as shown in [Figure 25](#), 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.

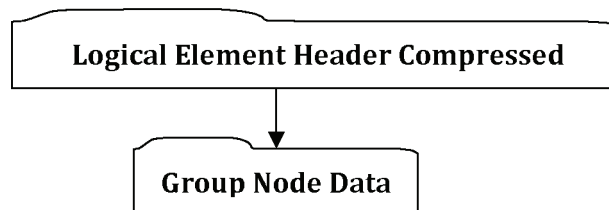


Figure 25 — Group Node Element data collection

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

An example diagrammatic representation of Group Node Data is shown in [Figure 26](#).

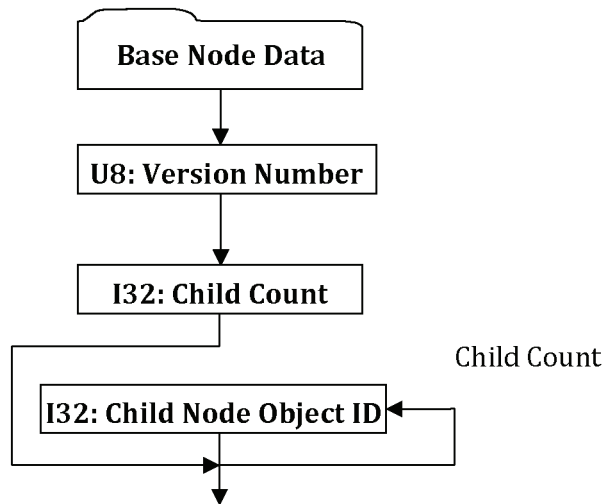


Figure 26 — Group Node Data collection

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. The value is set to 1.

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.

6.3.5 Instance Node Element

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

An Instance Node, shown in [Figure 27](#), 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.

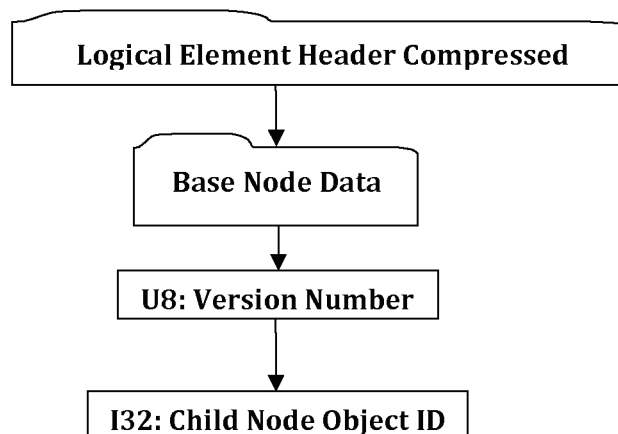


Figure 27 — Instance Node Element data collection

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. The value is set to 1.

I32: Child Node Object ID

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

6.3.6 Part Node Element

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

A Part Node Element, shown in [Figure 28](#), 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 (for example B-Rep data, PMI data).

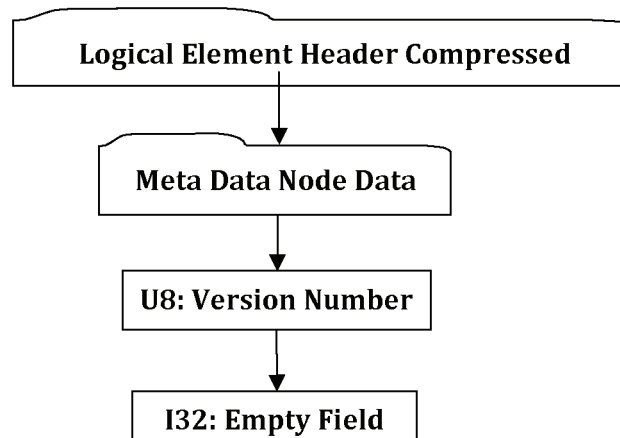


Figure 28 — Part Node Element data collection

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. The value is set to 1.

I32: Empty Field

Refer to Common Data Conventions and Constructs Empty Field description.

6.3.7 Meta Data Node Element

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

The Meta Data Node Element, shown in [Figure 29](#), is a node type used for storing references to specific “late loaded” meta-data (for example properties, PMI). The referenced meta-data is stored in a separate addressable segment of the ISO 14306 file (see Meta Data Segment) and thus the use of this Meta Data Node

Element is in support of the ISO 14306 file loader/reader “best practice” of late loading data (therefore storing the referenced meta-data in separate addressable segment of the ISO 14306 file allows a ISO 14306 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).

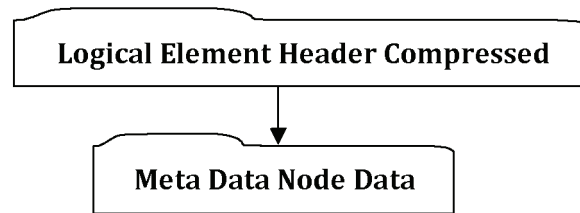


Figure 29 — Meta Data Node Element data collection

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

A diagrammatic representation of Meta Data Node Data is shown in [Figure 30](#).

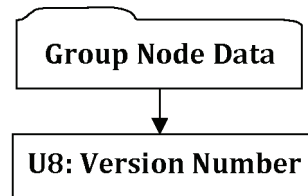


Figure 30 — Meta Data Node Data collection

Complete description for Group Node Data, as shown in [Figure 30](#), can be found in Group Node Data.

U8: Version Number

Version Number is the version identifier for this data. The value is set to 1.

6.3.8 LOD Node Element

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

A 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. A diagrammatic representation of LOD Node Element data is shown in [Figure 31](#).

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

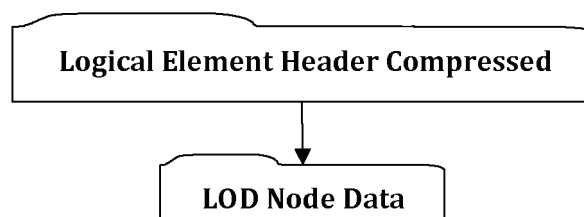


Figure 31 — LOD Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 31](#), can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

LOD Node Data

A diagrammatic representation of LOD Node Data is shown in [Figure 32](#).

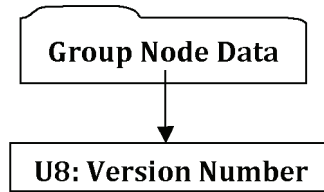


Figure 32 — LOD Node Data collection

Complete description for Group Node Data, as shown in [Figure 32](#), can be found in Group Node Data.

U8: Version Number

Version Number is the version identifier for this node. The value is set to 1.

6.3.9 Range LOD Node Element

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

Range LOD Nodes, as shown in [Figure 33](#), 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.

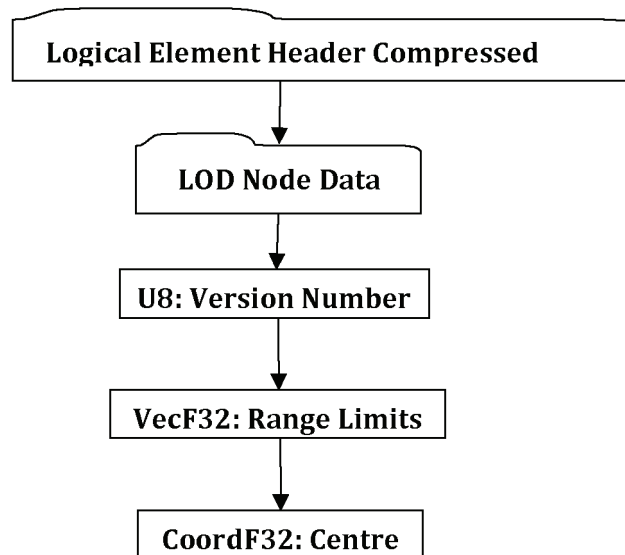


Figure 33 — Range LOD Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 33](#), 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. The value is set to 1.

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 ISO 14306 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 ISO 14306 Version 3 data are free to throw these values away and compute new values based on their desired LOD selection semantics.

6.3.10 Switch Node Element

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

The Switch Node, as shown in [Figure 34](#), 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.

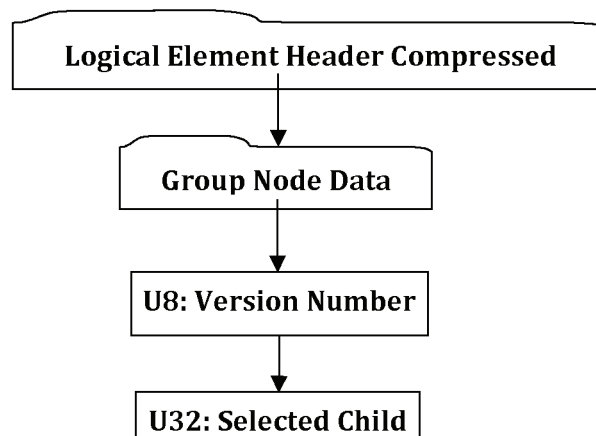


Figure 34 — Switch Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 34](#), 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. The value is set to 1.

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 (for example 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 ISO 14306 file, allows an ISO 14306 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 ISO 14306 format supports. The following sub-sections document the various Shape Node Element types.

6.3.11 Base Shape Node Element

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

Base Shape Node Element, shown in [Figure 35](#), represents the simplest form of a shape node that can exist within the LSG.

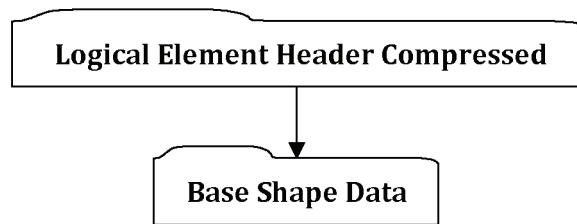


Figure 35 — Base Shape Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 35](#), can be found in the File Format section of this document under Data Segment in the logical collection describing Data .

Base Shape Data

A diagrammatic representation of Base shape data is shown in [Figure 36](#).

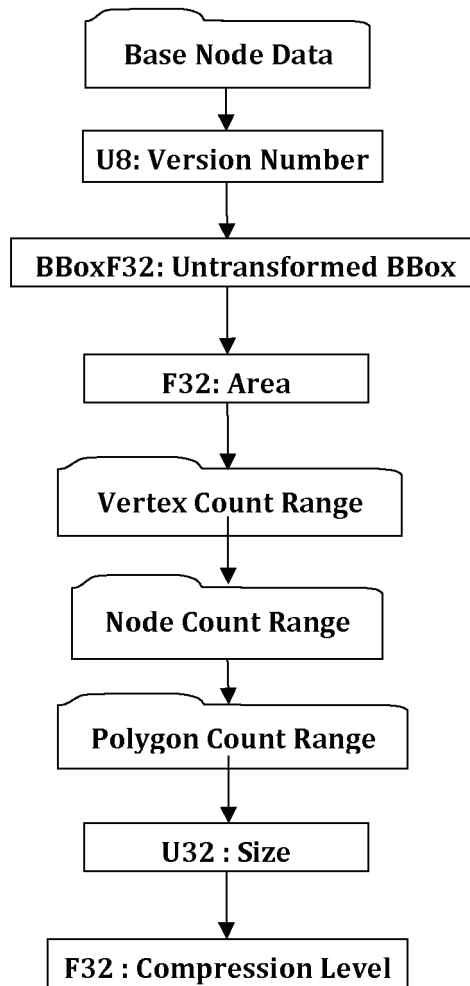


Figure 36 — Base Shape Data collection

A complete description of Base Node Data, as shown in [Figure 36](#), 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. The value is set to 1.

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 (therefore values scaled by MCS scaling).

Vertex Count Range

Vertex Count Range, as shown in [Figure 37](#), 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.

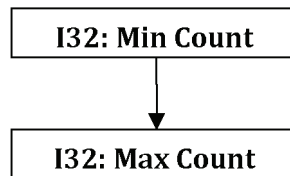


Figure 37 — Vertex Count Range data collection

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 (ISO 14306 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. ISO 14306 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, as shown in [Table 12](#), 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 ISO 14306 format by other data stored with both the Shape Node and the Shape LOD Element (for example 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.

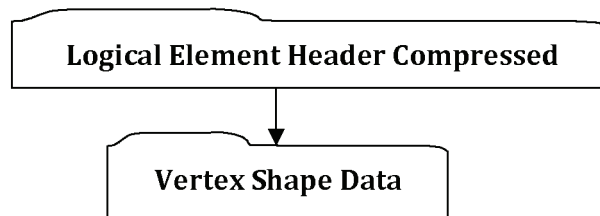
Table 12 — Compression level values

= 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.

6.3.12 Vertex Shape Node Element

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

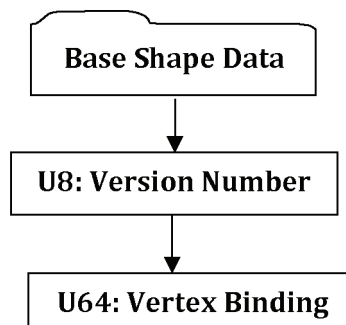
Vertex Shape Node Element represents shapes defined by collections of vertices, as shown in [Figure 38](#).

**Figure 38 — Vertex Shape Node Element data collection**

A complete description of Logical Element Header Compressed, as shown in [Figure 38](#), can be found in the File Format section of this document under Data Segment in the logical collection describing Data.

Vertex Shape Data

A diagrammatic representation of Vertex Shape Data is shown in [Figure 39](#).

**Figure 39 — Vertex Shape Data collection**

A complete description of Base Shape Data, as shown in [Figure 39](#), 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. The value is set to 1.

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 shall be set to “0”. For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

6.3.13 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, as shown in [Figure 40](#), 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.

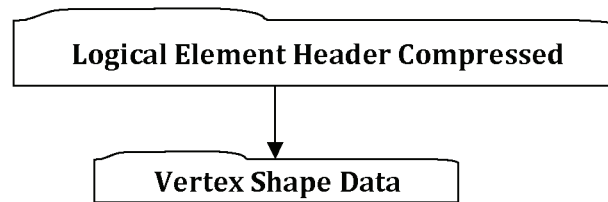


Figure 40 — Tri-Strip Shape Node Element data collection

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.

6.3.14 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, as shown in [Figure 41](#), 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.

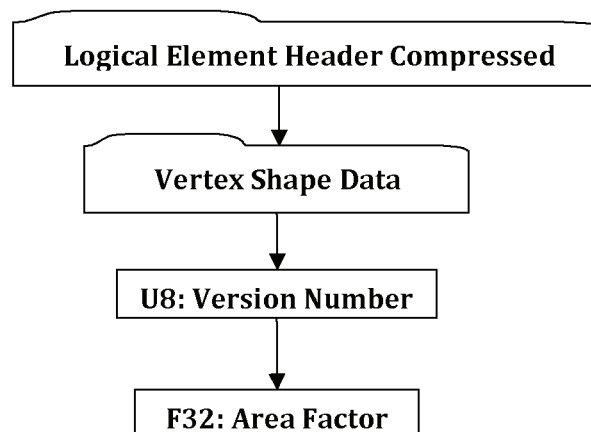


Figure 41 — Polyline Set Shape Node Element data collection

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. The value is set to 1.

F32: Area Factor

Area Factor specifies a multiplier factor applied to a Polyline Set computed surface area. In ISO 14306 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].

6.3.15 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, as shown in [Figure 42](#), defines a collection of independent and unconnected points. Each point constitutes one primitive of the set and is defined by one vertex coordinate.

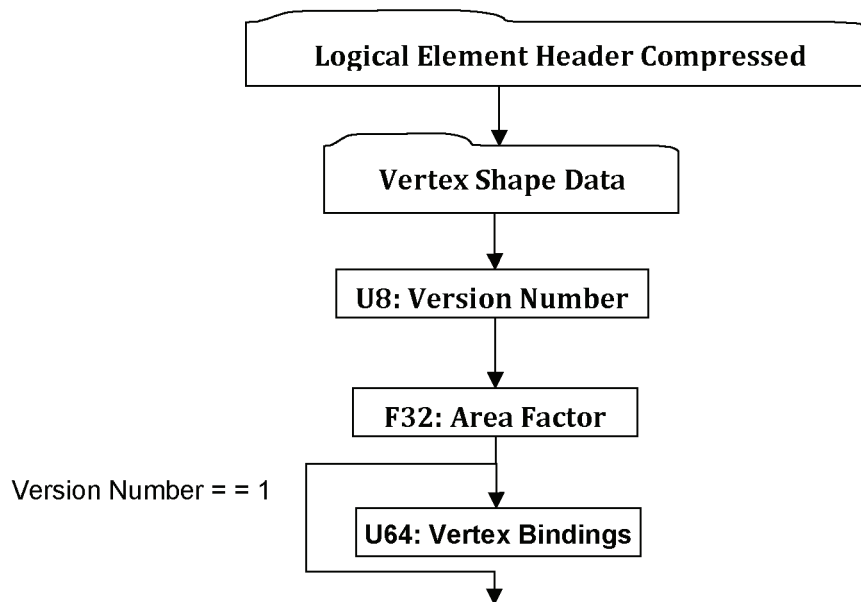


Figure 42 — Point Set Shape Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 42](#), 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. The value is set to 1.

F32: Area Factor

Area Factor specifies a multiplier factor applied to the Point Set computed surface area. In ISO 14306 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 (therefore 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 shall be set to “0”. For more information see Vertex Shape LOD Data U64: Vertex Bindings.

6.3.16 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, as shown in [Figure 43](#), 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.

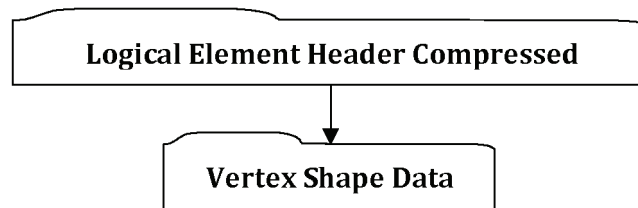


Figure 43 — Polygon Set Shape Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 43](#), 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.

6.3.17 NULL Shape Node Element

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

A NULL Shape Node Element, as shown in [Figure 44](#), defines a shape which has no direct geometric primitive representation (therefore 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 (therefore not persisted).

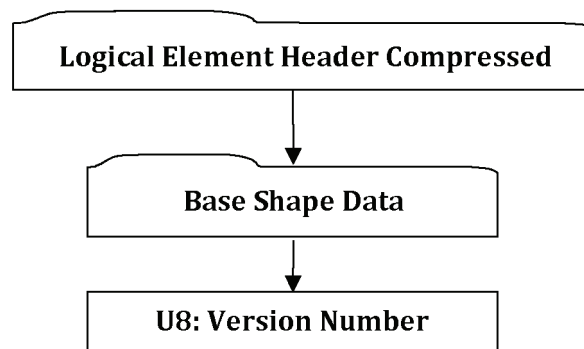


Figure 44 — NULL Shape Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 44](#), 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. The value is set to 1.

6.3.18 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 (for example box, cylinder, sphere, etc.) whose LODs can be procedurally generated. “Procedurally generate” means that the raw geometric shape definition data (for example vertices, polygons, normals, etc.) for LODs is not directly stored; instead some basic shape information is stored (for example sphere centre and radius) from which LODs can be generated.

Primitive Set Shape Node Elements, as shown in [Figure 45](#), 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 ISO 14306 file, allows a ISO 14306 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.

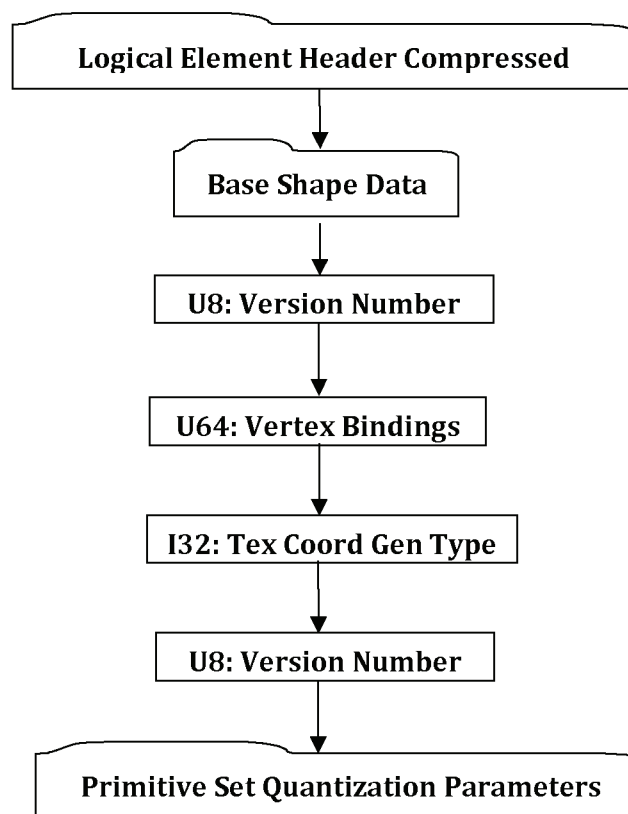


Figure 45 — Primitive Set Shape Node Element data collection

A complete description of Logical Element Header Compressed, as shown in [Figure 45](#), 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. The value is set to 1.

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 shall be set to "0". For more information see Vertex Shape LOD Data U64 : Vertex Bindings.

I32: Tex Coord Gen Type

Texture Coord Gen Type, as shown in [Table 13](#), 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 (therefore 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, as shown in [Table 14](#), is the version identifier for this element. The value of this Version Number indicates the format of data fields to follow. This information is related to the information stored in [Table 13](#).

Table 14 — Version Number values

= 0	Version 0 Format
= 1	Version 1 Format

Primitive Set Quantization Parameters

Primitive Set Quantization Parameters, as shown in [Figure 46](#), specifies for the two shape data type grouping (therefore 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 ISO 14306 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.

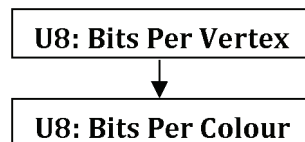


Figure 46 — Primitive Set Quantization Parameters data collection

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.

6.4 LSG Attribute Elements

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

For applications producing or consuming ISO 14306 format data, it is important that the ISO 14306 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 ISO 14306 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. ISO/CD 14306 Third Edition 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 (therefore 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

A diagrammatic representation of Base Attribute Data is shown in [Figure 47](#).

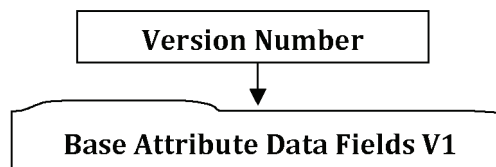


Figure 47 — Base Attribute Data collection

I8: Version Number

Version Number, as shown in [Figure 47](#), 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 (therefore. Material, Texture etc.) In all cases Base Attribute Data Fields V1 shall be read first.

Base Attribute Data Fields V1