BS ISO 14739-1:2014

BSI Standards Publication

Document management — 3D use of Product Representation Compact (PRC) format

Part 1: PRC 10001

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS ISO 14739-1:2014

National foreword

This British Standard is the UK implementation of ISO 14739-1:2014.

The UK participation in its preparation was entrusted to Technical Committee IDT/1, Document Management Applications.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2015.
Published by BSI Standards Limited 2015

ISBN 978 0 580 68764 8

ICS 35.240.30

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2015.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO 14739-1:2014

INTERNATIONAL STANDARD

ISO 14739-1

First edition
2014-12-15

Document management — 3D use of Product Representation Compact (PRC) format —

Part 1: PRC 10001

Gestion de documents — Utilisation en 3D du format compact de représentation de produit (PRC) —

Partie 1: PRC 10001

Reference number
ISO 14739-1:2014(E)

Documents
Document Sharing Hub

© ISO 2014

Full document details: BS ISO 14739-1:2014

COPYRIGHT PROTECTED DOCUMENT

© ISO 2014

All rights reserved. Unless otherwise specified, 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
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO 14739-1:2014

Contents

Page

Contents iii
Foreword v
1 Scope 1
2 Normative references 1
3 Terms and definitions 2
4 Document syntax conventions 2
4.1 Conventions 2
4.2 Example Structure 2
5 PRC file concepts 3
5.1 The PRC file 3
5.2 Versioning 5
5.3 Unique identifiers 6
5.4 Current data values 7
5.5 Userdata 7
5.6 Units 8
5.7 Tolerances 8
5.8 Compressed file sections 9
5.9 Compressed geometry 9
5.10 Compressed tessellation 9
6 PRC file contents 9
6.1 Fileheader 9
6.2 Filestructure 11
6.3 PRC Schema 13
7 PRC basic types 13
7.1 General 13
7.2 Uncompressed types 14
7.3 Compressed types 15
8 Base entities 21
8.1 General 21
8.2 Abstract root types 21
8.3 Structure and assembly 25
8.4 Miscellaneous Data 45
8.5 Graphics 56
8.6 Representation items 72
8.7 Markup 77
8.8 Tessellation 83
8.9 Topology 114
8.10 Curve 150
8.11 Surface 182
8.12 Mathematical Operator 209
9 Schema Definition 213
9.1 General 213
9.2 Enumeration Of Schema Tokens 214

Documents
Document Sharing Hub

© ISO 2014 – All rights reserved

Full document details: BS ISO 14739-1:2014

9.3 Schema Processing ... 216
9.4 Schema Requirements and Examples ... 222

10 I/O Algorithms ... 225
10.1 Getnumberofbitsusedtostoreunsignedinteger ... 225
10.2 Makeportable32bitsunsigned ... 225
10.3 Writebits ... 225
10.4 Writestring ... 226
10.5 Writefloatasbytes ... 226
10.6 Writecharacterarray ... 227
10.7 Writeshortarray ... 228
10.8 Writecompressedintegerarray ... 229
10.9 Writecompressedindicearray ... 229
10.10 Writeunsignedinteger ... 230
10.11 Writeinteger ... 230
10.12 Writeintegerwithvariablebitnumber ... 230
10.13 Writeunsignedintegerwithvariablebitnumber ... 231
10.14 Writedoublewithvariablebitnumber ... 231
10.15 Writenumberofbitsthenunsignedinteger ... 232
10.16 Writecompressedentitytype ... 232
10.17 Writedouble ... 233
10.18 Procedure For Writedouble ... 270

11 Tessellation Compression Support ... 274
11.1 General ... 274
11.2 Huffman Algorithm ... 275
11.3 Basis Pseudocode ... 277

Annex A (informative) **Example: Triangle** ... 281

Annex B (informative) **List of figures and tables** ... 283

Bibliography ... 284

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Foreword

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

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

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

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/TC 171, *Document management applications*, Subcommittee SC 2, *Application issues*.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Introduction

The data representations in PRC allows 3D design data, typically created in CAD and PLM systems, to be viewed and interrogated by visualization applications and to be integrated into complex documents.

This document specifies a wide range of data forms. The wide range is necessary to:

— Achieve a high fidelity, visually equivalent representation of 3D design data produced by an advanced CAD or PLM system without requiring the original application.

— Allow applications to compute high accuracy product shape measurements.

PRC is intended to complement native or open standard CAD and PLM formats as a compact, concise binary form for visualization and documentation. PRC is not intended as a data format for CAD interoperability or use in factory automation systems, e.g. automated manufacturing and inspection systems, which is addressed by the ISO 10303 standards.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

INTERNATIONAL STANDARD ISO 14739-1:2014

Document management — 3D use of Product Representation Compact (PRC) format —

Part 1:
PRC 10001

1 Scope

This International Standard describes PRC 10001 of a product representation compact (PRC) file format for three dimensional (3D) content data. This format is designed to be included in PDF (ISO 32000) and other similar document formats for the purpose of 3D visualization and exchange. It can be used for creating, viewing, and distributing 3D data in document exchange workflows. It is optimized to store, load, and display various kinds of 3D data, especially that coming from computer aided design (CAD) systems.

This International Standard does not apply to:

— Method of electronic distribution

— Converting CAD system generated datasets to the PRC format

— Specific technical design, user interface, implementation, or operational details of rendering

— Required computer hardware and/or operating systems

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 12651:1999, *Electronic imaging — Vocabulary*

ISO 24517-1:2008, *Document management — Engineering document format using PDF — Part 1: Use of PDF 1.6 (PDF/E-1)*

ISO 32000, *Document management — Portable document format*

IEEE 754, *Floating-Point Arithmetic*

The OpenGL Graphics System, A Specification, Version 4.1 (Core Profile), July 25, 2010[1]

[1] Available at http://www.opengl.org/registry/doc/glspec41.core.20100725.pdf

© ISO 2014 – All rights reserved

Documents Document Sharing Hub

Full document details: BS ISO 14739-1:2014

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 32000-1, ISO 24517-1 and ISO 12651 and the following apply.

3.1
PRC File Writer
software application which writes a particular PRC file

3.2
PRC File Reader
software application which reads a particular PRC file

3.3
byte
group of eight bits processed as a single unit of data

4 Document syntax conventions

4.1 Conventions

The following conventions are used within this document to describe data within a PRC File.

Terms highlighted in bold within this document signify field names in the description of entity types. Entity types are denoted in *italic*.

A table with three columns is used to describe the data within a contiguous portion of the file.

The first column indicates the name of the field. Field names are not unique and can be considered to have a scope limited to the data class.

The second column describes the type of the data. This might be

— A simple data type such as a Boolean, UnsignedInteger, or Double

— A simple class of data such as PRC_TYPE_TOPO_Body or PtrTopology where the name of the class is used to define the data stored for that class

— An Array<data class>[<size>] which indicates an array of data of the specified class. An array has <size> elements. Elements of an array are referenced beginning at 0.

The third column indicates if the field is required or optional. If the field is optional a condition is described when the field is present. The field may also be described.

4.2 Example Structure

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 1 — PRC file structure example

Name	Type	Value
flag1	*Boolean*	(Required) describe **flag1**
data_field1	*<data class1>*	(Optional; if **flag1** is TRUE) describe data_field1
data_field2	*<data class2>*	(Optional; if **flag1** is FALSE) describe data_field2
topological_body_data	*PRC_TYPE_TOPO_Body*	(Required) describe **topological_body_data**
data_type	*<enumerated type> or Integer*	(Required) describe **data_type**
data_of_type3	*<data class3>*	(Optional; if **data_type** is 1) describe **data_type3**
data_of_type4	*<data class4>*	(Optional; if **data_type** is 2) describe **data_type4**
Size	*UnsignedInteger*	(Required) describe **size**
array_of_type5	*Array<data class5>*[**size**]	(Required) describe the array of <data class5>

5 PRC file concepts

5.1 The PRC file

A PRC File is a sequential binary file, written in a way to make the file portable across machine architectures and operating systems.

PRC is optimized to store various kinds of 3D data, especially those coming from computer-aided design (CAD) systems. Most of the main constructs of CAD systems are supported within the PRC File Format:

— Assemblies and parts

— Trees of 3D entities (coordinate systems, wireframes, surfaces, and solids)

— Exact geometry representation for curves, surfaces

— Tessellated (triangulated) representation

— Markup data

PRC is meant to be multipurpose. There are two ways to store exact geometry and tessellation depending on the usage of the file and on the original information:

— Regular compression is used to directly represent CAD data without loss or transformation from the originating CAD system.

— High compression is used to store very small files, which have a specified physical tolerance from the originating shape. The tolerance is typically 0,001 mm for exact geometric data and 0,01 mm for tessellation data.

Each PRC File corresponds to a single model file (see 8.3.3) which defines the root product occurrences within the FileStructures of the PRC File. A PRC File is a collection of FileStructures which are independent from each other and can come from various authoring PRC File Writers. A FileStructure is the representation of an independent physical file denoting an independent 3D part, assembly, etc. within a PRC file. Hence, there is one header for each FileStructure with specific information and one global header for the model file which contains information about the PRC File Writer that assembled all these individual FileStructures into the final PRC File. Header sections gather primarily information on file version, FileStructure ids and offsets for reading / skipping sections.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Each FileStructure contains one or more product occurrences (see 8.3.10.2). The product occurrences denote the assembly hierarchy of the FileStructure. A product occurrence can have child nodes, which are also product occurrences. There is exactly one root product occurrence in each FileStructure. The root product occurrence is the only entry point to the FileStructure. (refer to Figure 1 below)

In parallel to the FileStructures, the model file also contains an array of root product occurrences that comprise the starting point for the entire assembly description. A product occurrence may refer to a corresponding part definition (see 8.3.11) which in turn contains:

— Geometrical data stored in a tree of representation items (see 8.6.3)

— An optional tree of part markups, grouped into annotation entities and views (see 8.7.2)

A product occurrence can contain:

— A tree of product markups

— Filters used to redefine the loading and presentation of data defined in the child product occurrences or in the part definition (see 8.3.12)

The representation items are defined through a combination of tessellation or topology and geometry data, which may be highly compressed. The markups are defined by tessellation data.

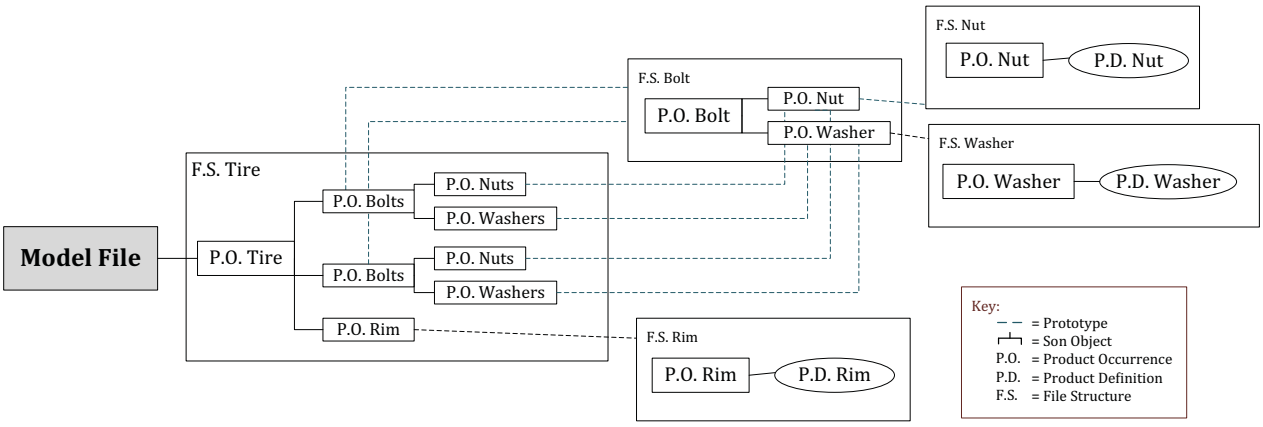

Figure 1 —Tessellation data

To optimize reading, a PRC File is arranged so that referenced entities are read before being referenced. Therefore, the FileStructures are ordered using parts, then subassemblies, and finally, the top (root) assembly.

A PRC File is composed of one header section, which starts with uncompressed data, one or more FileStructures, and one model file (*PRC_TYPE_ASM_ModelFile*) data section at the end, each individually compressed. Refer to 8.3.10 for more information about the PRC file structure.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 2 — PRC file structure

Section	Sub Sections	Compression	Description
FileHeader		Uncompressed	Defines originating author of the file; specifies start, and possibly end, location of other sections in file
File Structure 1	*FileStructureHeader*	Uncompressed	Identifiers of other File Structures referenced from within this FileStructure;
	Schema	Compressed	Description of changes between minimal version and authoring version of the FileStructure within the PRC File Format Specification
	Globals	Compressed	Referenced FileStructures and colors, line styles, and coordinate systems for each tree entity of the FileStructure
	Tree	Compressed	a description of the tree of items (product occurrences, part definitions, representation items, and markup)
	Tesselation	Compressed	All tessellated (triangulated) data in the leaf entities of the tree (representation items and markups).
	Geometry	Compressed	All exact geometry and topology data of the leaf entities of the tree(representation items)
	Extra Geometry	Compressed	Geometry summary data, which allow for partial loading of the FileStructure without loading the entire geometry
			Additional FileStructure sections in the PRC File
File Structure N		Compressed	Last FileStructure section in the PRC File
Schema	*ModelFile Schema*	Compressed	Description of changes between minimal version and authoring version of the ModelFile Section of the PRC File Format Specification
Model File Data	*PRC_TYPE_ASM_ModelFile*	Compressed	

5.2 Versioning

A file format version number is used to define the particular version of this international standard that a PRC File Reader or PRC File Writer conforms to.

Version numbers consist of the year modulo 2000 followed by three digits representing the day of the year. This international standard shall have the version 10001, corresponding to the 1st day of the year 2010.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

The **current_version** is the maximum version number that a PRC File Reader or PRC File Writer conforms to.

Each FileStructure and the FileHeader are independent and can be copied around or written by PRC File Writers of different versions. Each of them have their own **authoring_version** and **minimal_version_for_read** version numbers. The **authoring_version** represents the version of the international standard that the PRC File Writer conformed to at the time the FileHeader or the FileStructure was written. The **minimal_version_for_read** represents the version of the international standard that a PRC File Reader shall conform to to successfully read the FileHeader or the FileStructure. If the **current_version** of the PRC File Reader is less than the **minimal_version_for_read**, then the PRC File Reader shall not continue to process the file and report an error.

If **minimal_version_for_read** is lower than **authoring_version**, the PRC File Writer shall write a schema description in the PRC File providing the differences between the two versions of this international standard. This description will enable a PRC File Reader to read and skip new information, since these new data cannot be interpreted as they are from a newer version.

A PRC File schema contains a description of new fields or new entity types added between the **minimal_version_for_read** and the **authoring_version**. See 6.3 for a description of a schema.

A PRC File Reader uses the information in the schema to read and skip new information:

— After reading each entity type, the schema information is queried and new data are skipped accordingly, following the tokens.

— Each time an entity type is read, if the type is unknown, the schema is searched and its data is skipped.

5.3 Unique identifiers

5.3.1 General

A PRC File reader/writer shall generate (writer) or interprete (reader) unique identifiers for information within the PRC File. Each FileStructure within a PRC File has an identifier (UUID) which uniquely identifies this particular FileStructure among all of the FileStructures within the PRC File. Within each FileStructure, unique identifiers for referenceable entities are generated using the order that they are first encountered in the FileStructure. Thus, the first referenceable entity in the FileStructure has the number 1 as its identifier. Subsequent identifiers of referenceable entities are incremented by 1.

5.3.2 File structure

A PRC FileStructure is identified by an identifier (see *UncompressedUniqueId*) which is unique among all of the FileStructures within a single PRC File.

The method to calculate a unique identifier for a given FileStructure is not part of this international standard.

This approach offers an advantage, for example, such as when there is some intent to repurpose FileStructures inside other PRC files without entirely rewriting them.

5.3.3 Base entities

The PRC format provides support for referencing entities. See Entity Types in 8.2.1 for a list of entity types whose entities are referenceable.

The purpose of using references on entities is to enable an interpreter to handle the same entity several times without any duplication of data, either by any other program, or by another structure in the same or another PRC file.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

A referenceable entity is retrieved using the following:

— The UUID of the FileStructure.

— The entity's unique identifier (a non-zero unsigned 32-bit integer) within the FileStructure.

Just as the FileStructure UUID is unique among all of the FileStructures in the PRC File, the entity identifier is unique inside a FileStructure for all referenceable entities.

A PRC File Writer shall ensure that two referenceable entities inside the same FileStructure do not have the same identifier. The next available index within a FileStructure is the maximum value that has been assigned for identifiers to date, and is stored in the PRC File (see 8.3.4) so it is possible to safely add new entities to a FileStructure and assign them a unique identifier greater than this maximum value.

Data within the FileStructure tessellation and geometry sections are not accessible from outside the FileStructure using the approach for referenceable entities. These data are referenced only by data within the tree section (see 5.1 above) of the same FileStructure. However, it is possible (see 8.4.8) to reference topological data from outside the particular FileStructure which the topological entity lies in. This is restricted to faces in the current version of PRC but may be extended to other data in future versions of the format.

5.3.4 Other systems

In addition to the unique identifier mechanisms described above, this international standard provides for storage of identifiers from the originating system. The external identifiers and their persistence flag are stored as information in PRC strictly to support external workflows. These identifiers by themselves only exist to convey information and do not play any role in PRC.

5.4 Current data values

A PRC File reader and writer shall implement the concept of **CURRENT** for various data.

NOTE This enables smaller file sizes since duplicate data need not be written to the file. It also enables faster readers because some of the data is not being read. Default initial values are in parentheses.

— Current name (NULL)

— Current graphics

  — Index of layer (-1)

  — Index of line style (-1)

  — Behavior bit field (-1)

Current values shall be updated as they are encountered in the file. Values shall be reset when serialization is flushed at the end of every flate-compressed section (see 5.8 Compressed File Sections).

5.5 Userdata

The PRC File format provides a mechanism for a PRC File Writer to write private data within various sections of a PRC File. Such data shall consist of a bit stream of data containing the size of the bit stream followed by the specified number of bits.

NOTE Any PRC File Reader can read the bit stream and can even resave the private data, but it may not be able to interpret the data.

Each FileStructure within a PRC File may contain UserData. UserData are defined in conjunction with an application unique identifier (UUID) which allows for their interpretation. This application identifier

Documents
Document Sharing Hub

© ISO 2014 – All rights reserved

Full document details: BS ISO 14739-1:2014

shall be stored in the FileStructurerHeading. By default, any user data within the FileStructure shall be assumed to be written by this one writer. However, UserData may contain streams from different writers. Therefore, different application UUIDs shall be stored accordingly with each of these other data. UserData are meant to be interpreted only by software which is aware of its meaning according to a given writers UUID. A conforming PRC File Reader should either ignore those UserData, or interpret them according to the writers UUID. PRC also allows applications to define special attributes which behave similarly to UserData,as discussed in 7.3.3.8.

Unique application identifiers are assigned through Adobe Systems, initially, and probably ISO in the long term.. Each company should have it's own unique application identifier, and if they want to share data with another company, they should share application identifiers.

5.6 Units

A conforming writer shall define the units used in a PRC File. This should be done at both the model file level (see 8.3.3) and at the product occurrence level (see 8.3.10). The unit may come from an actual CAD file and therefore be considered reliable to represent physical values in the data. If a unit is valid/reliable, the flag unit_from_CAD_file shall be set to TRUE. However, some formats do not contain units. Regardless, it is mandatory to define one unit at both the model file level and at the product occurrence level .

The unit which shall be used is the first valid unit in the ModelFile / ProductOccurrence chain. If a valid unit is defined at the ModelFile level, it will apply to all product occurrences. Once a valid unit is found, the remainder of the data shall be interpreted with respect of that unit, even for occurrences higher in the product occurrence hierarchy.

NOTE In other words, if a product occurrence having no valid unit has a child with a valid unit, it is assumed that the entire hierarchy of model file and product occurrence are to be interpreted and used according to this unit.

If no valid unit is found at either the ModelFile level or any ProductOccurrence level (i.e. unit_from_CAD_file is always set to FALSE), a conforming PRC File Reader shall clearly indicate that the unit defined is not valid for measurement purposes.

5.7 Tolerances

PRC distinguishes between several notions of tolerances:

— A first notion of tolerance represents the maximum deviation between compressed and original data, as introduced by the lossy compression of geometry or topology. When provided, this tolerance shall be a user-defined value which represents a physical length given with a unit.

— A second notion of tolerance is introduced by numerical uncertainty inherent in every 3D modelling system. This form of tolerance is non-dimensional. Its purpose is to perform consistent numerical operations. This tolerance value corresponds to coincidence (e.g. of two 3D vertices) and is generally defined in conjunction with a minimal value representing zero and a maximal value representing infinity. For instance, a system might define coincidence at 1e-3, zero as any value less than 1e-12 and infinity as any value greater than 1e6. Then, additional logic outside the modelling system should define the unit so that the numerical values can be interpreted by the computer as physical values (i.e. in a particular unit).

In PRC, the 3D modelling system corresponds to entities in tessellation section (8.8) and topology section (7.9). Hence the various tolerances stored within these sections are always numerical values with no unit. A conforming writer shall never store a tolerance which can be directly interpreted as a physical value.

NOTE For instance, brep_data_compressed_tolerance in 8.9.19 which represents the deviation introduced between original data and compressed one is stored without unit, even if it might be derived from a user interface

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

which takes those units into account. Then, outside this 3D modelling system, unit is defined in ProductOccurrences and ModelFile as discussed in previous chapter. The physical interpretation of those tolerances should then be done from both indications. For instance, if a tolerance in a topology section is stored as 0,001 and if the unit of the ProductOccurrence it belongs to is 1000 meters, then the physical tolerance on data is actually 1 meter.

5.8 Compressed file sections

Within a PRC File, all sections, except header sections, are individually compressed with a Flate method.

NOTE This form of compression is considered to be "lossless". It occurs systematically whatever the actual content of the PRC file, and even if it contains compressed geometry or tessellation.

5.9 Compressed geometry

Compression of geometry results in very small files. This compression is "lossy" in that the geometry is an approximation to geometry to within a specified tolerance (typically 0,001mm). See 8.9.21 and 8.9.20 for entities representing compressed geometry. Note that the methodology to determine an approximation of the original geometry (e.g. analytic recognition) is not part of PRC standard. Only the resulting entities and the method to store them are described in PRC.

5.10 Compressed tessellation

Compression of tessellation data results in very small files. This compression is "lossy" in that the tessellation data is an approximation to tessellation data to within a specified tolerance (typically 0,01mm). See 8.8.9 for an entity representing compressed tessellation. Tessellation data is not necessarily generated from or considered as an "approximation" of geometry; it is an alternative way to convey data. Note that the methodology to determine an approximation of the original tessellation (e.g. polygon decimation) is not part of PRC standard. Only the resulting entities and the method to store them are described in PRC.

6 PRC file contents

6.1 Fileheader

6.1.1 General

The Header contains the file version for the authoring version (PRC Format Specification) that the PRC File Writer is based on and the minimal version for read (PRC Format Specification) that a conforming PRC File Reader is based on that write/read the data outside of the individual FileStructures in the PRC File.

Each FileStructure has a identifier which is unique among all of the FileStructures within this PRC File.

Each application has a unique identifier which enables the interpretation of UserData. This identifier shall be set to 0000 or to a valid application identifier. 0000 indicates that the authoring application is not "registered" but (as discussed above) there can still be user data from another application in the file. Valid identifiers are distributed by Adobe Systems, Inc. upon request.

A valid PRC File shall contain at least one FileStructure.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 3 — Fileheader

Name	Data Type	Data Description
	3 bytes: P, R, C	(Required) Characters "PRC"
minimal_version_for_read	*UncompressedUnsignedInteger*	(Required) Minimal version for read (see 5.2)
authoring_version	*UncompressedUnsignedInteger*	(Required) Authoring version (see 5.2)
unique_id_file	*UncompressedUniqueId*	(Required) Unique ID for file structure; The first and last sections of PRC File (before and after the FileStructures themselves) are a kind of special file structure which bear an ID as well.
unique_id_application	*UncompressedUniqueId*	(Required) Unique ID for application
filestructure_count	*UncompressedUnsignedInteger*	(Required) Number of FileStructures in PRC file
file_info	Array <*FileStructureDescription*>[**filestructure_count**]	(Required) Information describing each FileStructure in PRC File; the ordering of this array reflects the ordering of the FileStructure within the file.
start_offset	*UncompressedUnsignedInteger*	(Required) Start offset of ModelFileData section from beginning of PRC File (in bytes)
end_offset	*UncompressedUnsignedInteger*	(Required) End offset of ModelFileData section from beginning of PRC File (in bytes)
file_count	*UncompressedUnsignedInteger*	(Required) Number of uncompressed files that are saved in the PRC File
files:	Array <*UncompressedFiles*>[**file_count**]	(Optional; if **file_count** is greater than 0) Array of uncompressed files stored in the PRC File

6.1.2 *FileStructureDescription*

The *FileStructureDescription* contains information defining a particular FileStructure within the PRC File:

— Each FileStructure has an identifier which is unique among the FileStructures within a PRC File.

— The starting offset (in bytes from the beginning of the PRC File) of each of the following sections within a FileStructure: header, globals, tree, tessellation, geometry, and extra geometry.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 4 — *FileStructureDescription*

Name	Data Type	Data Description
unique_id	*UncompressedUniqueId*	(Required) Unique id for this file structure
	UncompressedUnsignedInteger	(Required) Reserved; shall be 0
section_count	*UncompressedUnsignedInteger*	(Required) Number of sections in this FileStructure (6 for header, globals, tree, tesselation, geometry, and extra geometry)
section_offsets	Array <*UncompressedUnsignedInteger*>[**section_count**]	(Required) Start offset (in bytes) of each section from the beginning of PRC File

6.1.3 *UncompressedFiles*

Directly embeds private data inside a PRC File. This may be referenced by objects in the same PRC File using the index of the uncompressed file within this array, and interpreted accordingly. Up to this version of the PRC File Format Specification, only picture objects make reference to these files (see 8.5.5)

Table 5 — *UncompressedFiles*

Name	Data Type	Data Description
count	*UncompressedUnsignedInteger*	(Required) Number of uncompressed files
array_of_files	Array<*UncompressedBlock*>[count]	(Required) Arbitrary data to embed within a PRC File

6.2 Filestructure

6.2.1 General

Table 6 — *Filestructure*

Name	Data Type	Data Description
header	*FileStructureHeader*	(Required)
schema	*FileStructureSchema*	(Required) Define the **schema** for the entities in this FileStructure which have changed between the **minimal_version_for_read** and the **authoring_version**
globals	*PRC_TYPE_ASM_FileStructureGlobals*	(Required) Referenced FileStructures and colors, line styles, and coordinate systems for each tree entity of the file structure
tree	*PRC_TYPE_ASM_FileStructureTree*	(Required) a description of the tree of items (product occurrences, part definitions, representation items, and markup)
tessellation	*PRC_TYPE_ASM_FileStructureTessellation*	(Requireed) All tessellated (triangulated) data in the leaf entities of the tree (representation items and markups).

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 6 *(continued)*

Name	Data Type	Data Description
Geometry	*PRC_TYPE_ASM_FileStructureGeometry*	(Required) All exact geometry and topology data of the leaf entities of the tree (representation items)
extra_geometry	*PRC_TYPE_ASM_FileStructureExtraGeometry*	(Required) Geometry summary data, which allow for partial loading of the file structure without loading the entire geometry

6.2.2 *FileStructureHeader*

A *FileStructureHeader* defines various properties of a FileStructure:

— The minimal file version of the a conforming PRC File Reader;

— The authoring version of a conforming PRC File Writer that wrote this FileStucture;

— The unique ID of this FileStructure; this ID shall be the same as the identifier in the PRC File Header section;

— The unique ID of the conforming PRC File Writer creating this file structure;

— A variable number of private uncompressed data files.

Table 7 — *FileStructureHeader*

Name	Data Type	Data Description
	3 bytes: P, R, C	(Required) Characters "PRC"
minimal_version_for_read	*UncompressedUnsignedInteger*	(Required) Minimal version for read (see 5.2)
authoring_version	*UncompressedUnsignedInteger*	(Required) Authoring version (see 5.2)
unique_id_file	*UncompressedUniqueId*	(Required) Unique ID of this FileStructure
unique_id_application	*UncompressedUniqueId*	(Required) Unique ID for application
file_count	*UncompressedUnsignedInteger*	(Required) Number of uncompressed files
files	Array <*UncompressedFile*>[file_count]	(Required) Array of uncompressed data

6.2.3 *FileStructureSchema*

Each FileStructure in a PRC File may represent a different version of the PRC Format Specification written by a different application (PRC File Writer). Each FileStructure shall define the schema for changes (new entities and new data fields to existing entities) for the entities that are stored within it. See 6.3 for a description of the facilities for describing a schema.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

6.3 PRC Schema

6.3.1 General

A schema describes changes between versions of PRC File Format Specification. See 5.2 for a basic description of versioning in PRC. This mechanism allows information to be added for a given entity type (**schema_type**) and still be readable by previous versions of the software.

Only those types which have changed between the **minimal_version_for_read** and **authoring_version** require a schema description, and only if they are present in the file.

Table 8 — PRC Schema

Name	Data Type	Data Description
schema_count	*UnsignedInteger*	(Required) Number of entity types that have changed between minimal version for read and authoring version
schemas	Array <*Entity_schema_definition*>[schema_count]	(Required) Schema definition of each type that has changed

6.3.2 *Entity_schema_definition*

This provides the schema definition for an entity type in a PRC File. The entity is described by an array of schema tokens which in turn should be viewed as a list of versioned blocks which describe versions of the entity. See 9.3.20 for a description.

Table 9 — *Entity_schema_definition*

Name	Data Type	Data Description
entity_type	*UnsignedInteger*	(Required) entity_type represents the entity type, such as 8.10.1, that is being described by the array of schema tokens.
token_count	*UnsignedInteger*	(Required) Number of tokens describing this entity type
schema_tokens	Array <*UnsignedInteger*>[token_count]	(Required) Array of schema tokens describing this entity type

7 PRC basic types

7.1 General

All data within a physical PRC File exists in a section which is a contiguous stream of bytes which starts and ends on a byte boundary. A section may be either uncompressed (header sections) or compressed (all other sections).

Data within an uncompressed section is written so that individual variables occupy a specific number of bytes.

Data within a compressed section has been written in a bit-by-bit manner with no restrictions on individual variables crossing byte boundaries. At the end of the section, the last byte is padded with zero bits, the entire section is compressed using flate (see Bibliography) and the compressed section is written to the file. At this point, the current name and current graphics are reset.

Reading one section is independent from reading other sections and one can skip directly to a section since the offset (in bytes) of the section from the beginning of the file is known as it is also stored in the PRC File.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

7.2 Uncompressed types

7.2.1 *UncompressedUnsignedInteger*

This type represents writing an unsigned integer. An unsigned integer is converted into a 4 byte array of unsigned characters which is independent of machine byte ordering using the algorithm MakePortable32BitsUnsigned.

Table 10 — *UncompressedUnsignedInteger*

Name	Data Type	Data Description
	4 bytes	(Required) 4 bytes representing unsigned integer

7.2.2 *UncompressedBlock*

The purpose of this function is to write the number of bytes being written followed by the specified number of bytes without the need to further interpret the content of the byte stream.

Table 11 — *UncompressedBlock*

Name	Data Type	Data Description
block_size	*UncompressedUnsignedInteger*	(Required) size (bytes) of uncompressed block
block	*Byte stream of the specified size*	(Required) Block of specified size

7.2.3 *UncompressedFiles*

This type represents writing the data for multiple uncompressed file data.

Table 12 — *UncompressedFiles*

Name	Data Type	Data Description
file_count	*UncompressedUnsignedInteger*	(Required) Number of uncompressed Files
files	Array <*UncompressedBlock*>[file_count]	(Required) Array of uncompressed file data

7.2.4 *UncompressedUniqueId*

This saves information on uniqueid in uncompressed mode.

Table 13 — *UncompressedUniqueId*

Name	Data Type	Data Description
unique_id0	*UncompressedUnsignedInteger*	(Required)
unique_id1	*UncompressedUnsignedInteger*	(Required)
unique_id2	*UncompressedUnsignedInteger*	(Required)
unique_id3	*UncompressedUnsignedInteger*	(Required)

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

7.3 Compressed types

7.3.1 General

The following types are used to define data contained within compressed sections of the PRC File. Each type may start or end at any bit position within a byte in the compressed section.

7.3.2 Atomic Types

7.3.2.1 *Bits(n)*

Writes n-number of bits from left to right.

7.3.2.2 *Boolean*

Boolean values shall be written as a single bit

Table 14 — *Boolean*

Name	Data Type	Data Description
	Bits(1)	(Required) Single bit in a bit stream

7.3.2.3 *Character*

Characters will be written as a single 8 bits.

Table 15 — *Character*

Name	Data Type	Data Description
	Bits(8)	(Required) Single character in a bit stream

7.3.2.4 *Integer*

This requires a special algorithm. See 10.11.

7.3.2.5 *IntegerWithVariableBitNumber*

This requires a special algorithm. See 10.12.

7.3.2.6 *UnsignedInteger*

This requires a special algorithm. See 10.10.

7.3.2.7 *UnsignedIntegerWithVariableBitNumber*

This requires a special algorithm. See 10.13.

7.3.2.8 *NumberOfBitsThenUnsignedInteger*

This requies a special algorithm. See 10.15.

7.3.2.9 *Float*

This requires a special algorithm. See 10.5.

7.3.2.10 *Double*

This requires a special algorithm. See 10.17.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

7.3.2.11 *DoubleWithVariableBitNumber*

This requires a special algorithm. See 10.14.

7.3.3 Compound Types

7.3.3.1 *CompressedEntityType*

This requires a special algorithm. See 10.16.

7.3.3.2 *CompressedUniqueId*

This saves information on uniqueid in compressed mode. See 5.3

Table 16 — *CompressedUniqueId*

Name	Data Type	Data Description
unique_id0	*UnsignedInteger*	(Required)
unique_id1	*UnsignedInteger*	(Required)
unique_id2	*UnsignedInteger*	(Required)
unique_id3	*UnsignedInteger*	(Required)

7.3.3.3 *String*

This is a UTF8-encoded string without a terminating NULL character. The number of characters does not include a terminating null character.

Table 17 — *String*

Name	Data Type	Data Description
null_flag	*Boolean*	(Required) TRUE if the string is not NULL; else FALSE
size	*UnsignedInteger*	(Optional; if **null_flag** is TRUE) Size of character array
string	Array <*Character*>[**size**]	(Optional; if **null_flag** is TRUE) Array of characters in String

7.3.3.4 *CharacterArray*

This requires a special algorithm. See 10.6.

7.3.3.5 *ShortArray*

This requires a special algorithm. See 10.7.

7.3.3.6 *CompressedIntegerArray*

This requires a special algorithm. See 10.8.

7.3.3.7 *CompressedIndiceArray*

This requires a special algorithm. See 10.9.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

7.3.3.8 *UserData*

Applications that write a PRC File may store an arbitrary bit stream of private data for the following data types :

— Subtypes of PRC_TYPE_ASM

— Subtypes of PRC_TYPE_RI

— Subtypes of PRC_TYPE_MKP

— Subtypes of PRC_TYPE_MISC

All those entities have UserData at the end of their data definition.

Table 18 — *UserData*

Name	Data Type	Data Description
stream_size	*UnsignedInteger*	(Required) Number of bits in the *UserDataStream*
stream	*UserDataStream*	(Optional; if **stream_size** > 0) Arbitrary bit stream of **stream_size** length

7.3.3.8.1 *UserDataStream*

Table 19 — *UserDataStream*

Name	Data Type	Data Description
section_count	*UnsignedInteger*	(Required) Number of UserData sub sections in the bit stream
sub_sections	Array <*UserDataSubSection*>[section_count]	(Required) Array of sub-sections

7.3.3.8.2 *UserDataSubSection*

Name	Data Type	Data Description
same_flag	*Boolean*	(Required) Same Application UUID as FileStructure
unique_id_application	*CompressedUniqueId*	(Optional; if **same_flag** is FALSE) Application UUID
stream_size	*UnsignedInteger*	(Required) Number of bits in the bit stream for this user data subsection
stream	*Bits(n)*	(Optional; if stream_size is > 0) Arbitrary bit stream of the stream_size length

7.3.4 Parameter Range Types

7.3.4.1 *Infinite_param*

The constant **infinite_param** shall have the double precision floating point value 12345. It is used to represent infinity, and as **–infinite_param** also minus infinity, for parameter values in **interval**.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

7.3.4.2 *Interval*

An Interval is a subset of R^1. It is represented by two double precision numbers defining the minimum and maximum values of the interval. An interval is defined to be all values between the minimum and maximum

minimum ≤ t ≤ maximum

The minimum shall not be smaller than **–infinite_param** and the maximum shall not be larger than **infinite_param**. Infinite or semi-infinite intervals may be specified by using infinite_param to represent infinity. The following are examples of intervals:

— [0.0, 1.0]

— [-infinite_param, infinite_param]

— [0.0, infinite_param]

The primary use of an interval is to define the domain of a curve. For a non-periodic curve, whether an open or closed curve, the interval defines the domain of legal parameter values for the curve. For a periodic closed curve, such as a circle or periodic closed NURBS curve, the interval defines the period (i.e. the length of the interval) as well as the primary domain of the curve. Any value is a legal parameter when evaluated modulo the period. For a periodic open curve, such as a proper subset of a circle, the interval defines the valid subset of R^1 that is the domain of legal parameter values.

Table 20 — *Interval*

Name	Data Type	Data Description
min_value	*Double*	(Required) Minimum value
max_value	*Double*	(Required) MaximumValue

7.3.4.3 *Parameterization*

Parameterization data provide a way to reparameterize a curve and to define the domain of the curve. The domain of the curve define legal parameter values.

Two doubles, **coeff_a** and **coeff_b**, are used to reparameterize a curve towards its implicit parameterization.

The evaluation formula to calculate the implicit parameter from a parameter is:

implicit_parameter ← . **coeff_a** * parameter + **coeff_b**

If there is no reparameterization of the curve, **coeff_a** shall be set to 1.0 and **coeff_b** shall be set to 0.0. In this case implicit_parameter equals the specified parameter.

Parameterization data also contain an interval used to define the domain of the curve. This interval restricts the curve before applying the reparameterization formula. The reparameterization formula can be used to calculate the implicit_interval from the given interval in the same way that it is used to calculate the implicit_parameter from a given parameter.

All curves shall have an interval to define the legal parameter values of the curve. In the case of base curves (curves not defined by reference to other curves), this interval defines the domain of definition for the curve. For curves which are defined in terms of other curves, the interval represents a subset of the curve which may be the entire curve or a portion of it. The interpretation of the interval depends upon the curve being periodic or non-periodic (see the definition of Interval).

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

For instance, a circle has the implicit domain of [0.0, 2π] and a line has the implicit domain of [-infinite_param, infinite_param]. A portion of the circle might be limited to [0.0, π].

Table 21 — *Parameterization*

Name	Data Type	Data Description
trim_interval	*Interval*	(Required) trim interval
coeff_a	*Double*	(Required) Coeff_a
coeff_b	*Double*	(Required) Coeff_b

As an example, consider a circle. It has an implicit parameterization on the interval [0.0, 2π]. For this case:

— Coeff_a = 1.0

— Coeff_b = 0.0

— Interval = [0.0, 2π]

To reparameterize the circle so the parameter values are in the interval [0.0, 1.0] would give

— Coeff_a = 2π

— Coeff_b = 0.0

— Interval = [0.0, 1.0]

To define a semi circle

— Coeff_a = 1.0

— Coeff_b = 0.0

— Interval = [-π/2.0, π/2.0]

7.3.4.4 *Domain*

A Domain is a subset of R^2 that is used to define the domain of definition of a surface. It is represented by a 2D vector defining the minimum UV parameter values and a 2D vector defining the maximum UV parameters.

The domain of the surface is

minimum U ≤ u ≤ maximum U

minimum V ≤ v ≤ maximum V

The interval in U (or V) when used with a specific surface definition will define a surface that is open, closed, or periodic in that parameter. For a non-periodic surface, whether an open or closed surface, the interval defines the domain of legal parameter values for the surface. For a periodic closed surface, such as a cone or periodic closed NURBS surface, the interval defines the period (i.e. the length of the interval) as well as the primary domain of the surface. Any value is a legal parameter when evaluated modulo the period. For a periodic open surface, such as a proper subset of a cone, the interval defines the valid subset of R^2 that is the domain of legal parameter values.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 22 — *Domain*

Name	Data Type	Data Description
min_uv	*Vector2d*	(Required) Minimum U and V
max_uv	*Vector2d*	(Required) Maximum U and V

7.3.4.5 *UVParameterization*

This describes a reparameterization of a surface towards implicit parameterization. Coeff_a shall be set to 1.0 and Coeff_b shall be set to 0.0 if there is no reparameterization.

The evaluation formula is:

If (swap_uv is FALSE) {

implicit_param.u ← **u_param_coeff_a** * param.u + **u_param_coeff_b**

implicit_param.v ← **v_param_coeff_a** * param.v + **v_param_coeff_b**

} else {

implicit_param.u ← **v_param_coeff_a** * param.v + **v_param_coeff_b**

implicit_param.v ← **u_param_coeff_a** * param.u + **u_param_coeff_b**

}

Table 23 — *UVParameterization*

Name	Data Type	Data Description
swap_uv	*Boolean*	(Required) **swap_uv** TRUE implies swap the uv param; FALSE implies do not swap
suface_domain	*Domain*	(Required) The domain of the surface is specified in numbers before the previous reparameterization formula is aplied.
u_param_coeff_a	*Double*	(Required)
v_param_coeff_a	*Double*	(Required)
u_param_coeff_b	*Double*	(Required)
v_param_coeff_b	*Double*	(Required)

7.3.5 Basic Geometry Types

7.3.5.1 *Vector2d*

Representation of a 2D vector. This type may be used to represent either 2D positions or vectors. The context shall be used to determine if a position or vector is meant.

Table 24 — *Vector2d*

Name	Data Type	Data Description
x_value	*Double*	(Required)
y_value	*Double*	(Required)

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

7.3.5.2 *Vector3d*

Representation of a 3D vector. This type may be used to represent either 3D positions or vectors. The context shall be used to determine if a position or vector is meant.

Table 25 — *Vector3d*

Name	Data Type	Data Description
x_value	*Double*	(Required)
y_value	*Double*	(Required)
z_value	*Double*	(Required)

7.3.5.3 *BoundingBox*

Define a bounding box with sides parallel to the XYZ coordinate planes using two diagonal corners of the box. The minimum and maximum shall satisfy

— Xmin < Xmax

— Ymin < Ymax

— Zmin < Zmax

If the above is not satisfied the bounding box is considered invalid.

Table 26 — *BoundingBox*

Name	Data Type	Data Description
minimum_corner	*Vector3d*	(Required)
maximum_corner	*Vector3d*	(Required)

8 Base entities

8.1 General

BASE ENTITIES represent high level concepts such as curves, surfaces, topology, parts, assemblies, markups, or tessellation data which are stored in a PRC File. Entities are defined by a type name used for descriptive purposes, a type value which is stored in the PRC File to indicate that the data defining the entity follows, and an indication if the entity is referenceable. An entity is referenceable if it may be referenced using a unique identifier.

8.2 Abstract root types

8.2.1 Entity types

Table 27 — Abstract root types entity types

Type Name	Type Value
PRC_TYPE_ROOT	0
PRC_TYPE_ROOTBase	*PRC_TYPE_ROOT*+ 1
PRC_TYPE_ROOT_PRCBaseWithGraphics	*PRC_TYPE_ROOT*+ 2
PRC_TYPE_ROOT_PRCBaseNoReference	*PRC_TYPE_ROOT* + 3

Documents
Document Sharing Hub

© ISO 2014 – All rights reserved

Full document details: BS ISO 14739-1:2014

8.2.2 *PRC_TYPE_ROOT*

An entity of this type in a PRC File is to be interpreted as a NULL pointer or entity depending on the specific circumstances.

8.2.3 *PRC_TYPE_ROOTBase*

8.2.3.1 General

This is the abstract root type for any PRC entity.

8.2.3.2 *ContentPRCBase*

This represents common data for all PRC base entities that are not eligble to be referenced.. All PRC base entities have attribute and name information.

Table 28 — *ContentPRCBase*

Name	Data Type	Data Description
attribute_data	*AttributeData*	(Required) Attribute data associated with the entity
entity_name	*Name*	(Required) Entity Name

8.2.3.3 *ContentPRCRefBase*

This represents common data for all PRC base entities that are eligible to be referenced. All PRC base entities have attribute, name information, a PRC FileStructure unique identifier and a persistent and non-persistent identifier from the originating CAD file.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 29 — *ContentPRCRefBase*

Name	Data Type	Data Description
attribute_data	*AttributeData*	(Required) Attribute data associated with the entity
entity_name	*Name*	(Required) Entity Name
unique_id_cad	*UnsignedInteger*	(Required) persistent CAD identifier from originating CAD file
unique_id	*UnsignedInteger*	(Required) PRC FileStructure unique identifier

8.2.3.3.1 *AttributeData*

A base entity may have zero or more associated attributes.

Table 30 — *AttributeData*

Name	Data Type	Data Description
attribute_count	*UnsignedInteger*	(Required) Number of attributes
attributes	Array <*PRC_TYPE_MISC_Attribute*>[attribute_count]	(Required) An array of attributes

8.2.3.3.2 *Name*

PRC employs the concept of a current_name which retains the name of the last entity being read or written. If the name of the subsequent entity being read or written is the same as the current name, no name will be in the file. Otherwise, a name for the entity is read from the file and the current name is updated to this new name. This is done to optimize on space within the file.

Table 31 — *Name*

Name	Data Type	Data Description
same_name	*Boolean*	(Required) TRUE implies the name of this entity is the same as the current name; FALSE implies that a new name is in the file
name	*String*	(Optional; if **same_name** is FALSE) Name of the entity and the current name is set to this name

8.2.4 *PRC_TYPE_ROOT_PRCBaseWithGraphics*

8.2.4.1 General

Information for any base PRC entity which can be referenced and which contains graphics.

PRC employs the concept of current graphics content which retains the graphics content of the last entity being read or written. If the graphics content of the subsequent entity being read or written is the same as that of the current graphics content, no graphics content will be in the file. Otherwise, a graphics content for the entity is read from the file and the current graphics content is updated to this new graphics content. This is done to optimize on space within the file.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 32 — *PRC_TYPE_ROOT_PRCBaseWithGraphics*

Name	Data Type	Data Description
base	*ContentPRCRefBase*	(Required) Base information associated with the entity
same_graphics	*Boolean*	(Required) SameGraphicsAsCurrent
graphic_content	*GraphicsContent*	(Optional; if **same_graphics** is TRUE) Graphical data associated with the entity and the current graphics content is set to this graphics content

8.2.4.2 *GraphicsContent*

index_of_line_style represents the index into the array of styles, which is stored in FileStructureInternalGlobalData (See section 8.3.5.2below). The array of styles is a set of PRC_TYPE_GRAPH_Style entities (See 8.5.3) and should have a value less than 65535.

behaviour_bit_field is an unsigned short integer (2 bytes). This bit field controls visibility and removal of an entity as well as how values of entities (visibility, color, transparency, layer, line pattern, and line width) are inherited in the tree of entities.

Each value that is subject to inheritance has two bit flags in the **behavior_bit_field**: one flag is called Childherit. If ChildHerit is set, the actual value of ParentHerit is ignored and assumed to be FALSE.

The actual value that the entity uses depends on the path of entites from the root of the tree of entites to this entity and the **behavior_bit_field** settings along this path.

If there are ChildHerit flags in the tree, it is the lowest node in the tree which has this flag which defines the value. Else if there are ParentHerit flags in the tree, it is the highest node in the tree which has this flag which defines the value. Else if there is no flag, the current value is set, if any.

Potential values and meanings of this bit field are:

Table 33 — behavior_bit_field

Name	Value	Meaning
PRC_GRAPHICS_Show	0x0001	The entity is shown.
PRC_GRAPHICS_ChildHeritShow	0x0002	Shown entity child inheritance.
PRC_GRAPHICS_FatherHeritShow	0x0004	Shown entity parent inheritance.
PRC_GRAPHICS_ChildHeritColor	0x0008	Color/material child inheritance.
PRC_GRAPHICS_ParentHeritColor	0x0010	Color/material parent inheritance
PRC_GRAPHICS_ChildHeritLayer	0x0020	Layer child inheritance.
PRC_GRAPHICS_ParentHeritLayer	0x0040	Layer parent inheritance.
PRC_GRAPHICS_ChildHeritTransparency	0x0080	Transparency child inheritance.
PRC_GRAPHICS_ParentHeritTransparency	0x0100	Transparency parent inheritance
PRC_GRAPHICS_ChildHeritLinePattern	0x0200	Line pattern child inheritance.
PRC_GRAPHICS_ParentHeritLinePattern	0x0400	Line pattern parent inheritance.
PRC_GRAPHICS_ChildHeritLineWidth	0x0800	Line width child inheritance
PRC_GRAPHICS_ParentHeritLineWidth	0x1000	Line width parent inheritance
PRC_GRAPHICS_Removed	0x2000	The entity has been removed and no longer appears in the tree

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 34 — *GraphicsContent*

Name	Data Type	Data Description
biased_layer_index	*UnsignedInteger*	**(Required) layer_index** + 1 where **layer_index** represents the layer the entity lies on. It should have a value less than 65535.
biased_index_of_line_style	*UnsignedInteger*	**(Required) index_of_line_style** + 1 where **index_of_line_style** represents the index into the array of styles, which is stored in *FileStructureInternalGlobalData* (See section 8.3.5.2 below). The array of styles is a set of *PRC_TYPE_GRAPH_Style* entities (See 8.5.3) and should have a value less than 65535.
behavior_bit_field1	*UnsignedCharacter*	**(Required) behavior_bit_field1 is behavior_bit_field & 0xff**
behavior_bit_field2	*UnsignedCharacter*	**(Required) behavior_bit_field2 is behavior_bit_field >> 8**

8.2.5 *PRC_TYPE_ROOT_PRCBaseNoReference*

This is the abstract root type for any PRC entity that can not be referenced.

This type is useful for schema descriptions. For example, *EPRCSchema_Parent_Type* can be used to define a new type which has one of three different ancestor types. It can be a child of either *PRC_TYPE_ROOTBase*, *PRC_TYPE_ROOT_PRCBaseWithGraphics*, or *PRC_TYPE_ROOT_PRCBaseNoReference*.

PRC_TYPE_ROOTBase: ancestor type would indicate that the new type is a referencable type

PRC_TYPE_ROOT_PRCBaseNoReference: ancestor type would indicate that the new type is not referencable.

PRC_TYPE_ROOT_PRCBaseWithGraphics.: ancestor type would indicate that the new type is a referencable type which bear graphics

Examples :

PRC_TYPE_GRAPH_LinePattern has PRC_TYPE_ROOTBase in its ancestor chain

PRC_TYPE_ASM_FileStructure has *PRC_TYPE_ROOT_PRCBaseNoReference* in its ancestor chain

PRC_TYPE_RI_Curve has *PRC_TYPE_ROOT_PRCBaseWithGraphics* in its ancestors chain

8.3 Structure and assembly

8.3.1 Entity types

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 35 — Structure and assembly entity types

Type Name	Type Value	Referenceable
PRC_TYPE_ASM	*PRC_TYPE_ROOT* + 300	no
PRC_TYPE_ASM_ModelFile	*PRC_TYPE_ASM*+ 1	no
PRC_TYPE_ASM_FileStructure	*PRC_TYPE_ASM*+ 2	no
PRC_TYPE_ASM_FileStructureGlobals	*PRC_TYPE_ASM*+ 3	no
PRC_TYPE_ASM_FileStructureTree	*PRC_TYPE_ASM*+ 4	no
PRC_TYPE_ASM_FileStructureTessellation	*PRC_TYPE_ASM*+ 5	no
PRC_TYPE_ASM_FileStructureGeometry	*PRC_TYPE_ASM*+ 6	no
PRC_TYPE_ASM_FileStructureExtraGeometry	*PRC_TYPE_ASM*+ 7	no
PRC_TYPE_ASM_ProductOccurrence	*PRC_TYPE_ASM*+ 8	yes
PRC_TYPE_ASM_PartDefinition	*PRC_TYPE_ASM*+ 9	yes
PRC_TYPE_ASM_Filter	*PRC_TYPE_ASM*+ 10	yes

8.3.2 *PRC_TYPE_ASM*

This is the abstract type for the top level PRC structure.

8.3.3 *PRC_TYPE_ASM_ModelFile*

8.3.3.1 General

A model file (*PRC_TYPE_ASM_ModelFile*) is typically created by importing data from a CAD file. There is only a single *PRC_TYPE_ASM_ModelFile* entity in a PRC File. The model file contains product occurrences, which are split into different FileStructures.

units_from_CAD_file is to be interpreted as discussed in section *Units* (See 5.6)

number_of_root_product_occurrences is the number of root product occurrences in the model file.

product_occurrences represents the root product occurrences in the model file.

file_structure_index_in_model_file indicates the index at which the FileStructure should be stored in memory within a ModelFile (*PRC_TYPE_ASM_ModelFile*). A conforming PRC File Reader should reconstitute / maintain FileStructures in memory in the order that they appear in the ModelFile, regardless of the order in the physical PRC file. In the physical PRC file, the order of file structures shall be in accordance with their dependencies with other file structures as follows: if a given file structure A depends on another file structure B (in the sense that A references elements of B), B shall appear first in the PRC physical file. This restriction does not exist for the memory storage of the file structures.

number_of_file_structures is obtained directly from the header of the PRC File.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 36 — *PRC_TYPE_ASM_ModelFile*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_ModelFile*
Base	*ContentPRCBase*	(Required) Base information associated with the entity
units_from_CAD_file	*Boolean*	(Required) Units_from_CAD_file is TRUE if the units come from a CAD system and are thus suitable for measurement; else FALSE
	Double	(Optional; if **units_from_CAD_file** is TRUE) Units in multiple of mm
number_of_root_product_occurrences	*UnsignedInteger*	(Required) **number_of_root_product_occurrences**
product_occurences	Array <*ProductOccurrenceReference*> [**number_of_root_product_occurrences**]	(Required) References to the root **product occurrences** in the model file
file_structure_index_in_model_file	Array <*UnsignedInteger*> [**number_of_root_product_occurrences**]	(Required) **file_structure_index_in_model_file**: Indicies to to FileStructure within the model file; the size of the array is obtained directly from the header of the PRC file
user_data	*UserData*	(Required) User defined data

8.3.3.2 *ProductOccurrenceReference*

This defines the unique identifier of the *PRC_TYPE_ASM_FileStructure* and the index of the root product occurence in the array of product occurrences within the *PRC_TYPE_ASM_FileStructureTree*, as contained within the *PRC_TYPE_ASM_FileStructure*.

product_occurrence_is_active is reserved for future use, and is currently unused. Its default value should be TRUE. Its use will be to control storing different versions/configurations of a product in the same PRC File. Currently, only one can be active at a time but you can switch from one to the other.

Table 37 — *ProductOccurrenceReference*

Name	Data Type	Data Description
unique_id	*CompressedUniqueId*	(Required) Unique identifier of the FileStructure that contains this root product occurrence.
root_index	*UnsignedInteger*	(Required) Index of the root product occurrence within the FileStructure
product_occurence_is_active	*Boolean*	(Required) TRUE if the product occurrence is active; else FALSE

8.3.4 *PRC_TYPE_ASM_FileStructure*

This type gathers internal data of a file structure as described in *PRC_TYPE_ASM_FileStructureTree*

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 38 — *PRC_TYPE_ASM_FileStructure*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_FileStructure*
Base	*ContentPRCBase*	(Required) Base information associated with the entity
next_available_index	*UnsignedInteger*	(Required) **next_available_index** is used when re-opening the FileStructure, to be able to safely add entities with a unique id without overlap to pre-existing entities.
index_product_occurrence	*UnsignedInteger*	(Required) **index_product_occurrence** is used to denote which product occurrence inside the FileStructure is the unique root.

8.3.5 *PRC_TYPE_ASM_FileStructureGlobals*

8.3.5.1 General

This type gathers global data of a file structure as described in Section 6.2.

Table 39 — *PRC_TYPE_ASM_FileStructureGlobals*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_FileStructureGlobals*
Base	*ContentPRCBase*	(Required) Base information associated with the entity
file_count	*UnsignedInteger*	(Required) Number of referenced FileStructures
unique_ids	Array <*CompressedUniqueId*> [**file_count**]	(Required) Unique ids for FileStructures within the PRC File which are referenced by entities in this FileStructure
global_data	*FileStructureInternalGlobalData*	(Required)
user_data	*UserData*	(Required) User defined data

8.3.5.2 *FileStructureInternalGlobalData*

8.3.5.2.1 General

This internal structure is used in ***PRC_TYPE_Asm_FileStructureGlobals***.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 40 — *FileStructureInternalGlobalData*

Name	Data Type	Data Description
tess_chord	*Double*	(Required) Tessellation chord height
tess_angle	*Double*	(Required) Tessellation angle (degrees)
serialize_help	*MarkupSerializationHelper*	(Required) See below
color_count	*UnsignedInteger*	(Required) Number of colors
Colors	Array <*RgbColor*>[**color_count**]	(Required) Array of color definitions
picture_count	*UnsignedInteger*	(Required) Number of pictures
Pictures	Array <*PRC_TYPE_GRAPH_Picture*>[**picture_count**]	(Required) Array of pictures
texture_count	*UnsignedInteger*	(Required) Number of textures definitions
Textures	Array <*PRC_TYPE_GRAPH_TextureDefinition*> [**texture_count**]	(Required) Array of texture definitions
material_count	*UnsignedInteger*	(Required) Number of materials
Materials	Array <*PRC_TYPE_GRAPH_Material*> [**material_count**]	(Required) Array of materials
line_pattern_count	*UnsignedInteger*	(Required) Number of line patterns
line_patterns	Array <*PRC_TYPE_GRAPH_LinePattern*> [**line_pattern_count**]	(Required) Array of line patterns
style_count	*UnsignedInteger*	(Required) Number of styles
Styles	Array <*PRC_TYPE_GRAPH_Style*>[**style_count**]	(Required) Array of category 1 line styles
fill_count	*UnsignedInteger*	(Required) Number of fill patterns
Fills	Array <*PRC_TYPE_GRAPH_FillPattern*>[**fill_count**]	(Required) Array of fill patterns
ref_coord_count	*UnsignedInteger*	(Required) Number of reference coordinate systems
ref_coords	Array <*PRC_TYPE_RI_CoordinateSystem*> [**ref_coord_count**]	(Required) Array of reference coordinate systems

8.3.5.2.2 *MarkupSerializationHelper*

8.3.5.2.2.1 General

This Global data is composed of font information for markup. The following example shows how the global data is serialized for markup font information.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 41 — *MarkupSerializationHelper*

Name	Data Type	Data Description
default_font_family_name	*String*	(Required) **default_font_family_name** defines the case-sensitive default font family name used for text if a given font family is not available on the computer. If this default font family itself is not available on the computer, the font will be MyriadPro from Adobe Systems, Inc.
font_keys_count	*UnsignedInteger*	(Required) Number of fonts
font_keys_of_font	Array <*FontKeysSameFont*> [**font_keys_count**]	(Required) **font_keys_of_font** represents several font keys sharing the same base font name. Indices for font keys used in *ContentMarkupTess* are calculated from this array. For example, if there are two font names representing 4 and 5 font keys respectively, index #7 would be represented by font_keys[1][3].

8.3.5.2.3 *FontKeysSameFont*

This type describes a list of usages of the same font (referred to by its name) with different metrics and attributes.

— **attributes** represents the font attributes, and is a combination of the values in the table below.

Table 42 — *FontKeySameFont*

Name	Data Type	Data Description
font_name	*String*	(Required) Font Name
character_set	*UnsignedInteger*	(Required) Character Set
key_count	*UnsignedInteger*	(Required) Number-of-font-keys
font_key_list	Array<*FontKey*> [**key_count**]	(Required)

Table 43 — *FontKey*

Name	Data Type	Data Description
font_size	*UnsignedInteger*	(Required) font size + 1
font_attributes	*Character*	(Required) Font Attributes

The following table contains the possible values for the **Character Set**.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 44 — Character Set

Value	Description
0	Roman
1	Japanese
2	Traditional Chinese
3	Korean
4	Arabic
5	Hebrew
6	Greek
7	Cyrillic
8	RightLeft
9	Devanagari
10	Gurmukhi
11	Gujarati
12	Oriya
13	Bengali
14	Tamil
15	Telugu
16	Kannada
17	Malayalam
18	Sinhalese
19	Burmese
20	Khmer
21	Thai
22	Laotian
23	Georgian
24	Armenian
25	Simplified Chinese
26	Tibetan
27	Mongolian
28	Geez
29	EastEuropeanRoman
30	Vietnamese
31	ExtendedArabic

font_attributes represents the font attributes, and is a combination of the following values.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 45 — Font attributes

Font attribute value	Description
2	Bold
4	Italic
8	Underlined
16	Strike-Out
32	Overlined
64	Stretch. In case the font to be used is not the original font, this attribute value indicates that the text shall be stretched to fit within its bounding box
128	Wire. This attribute value indicates that the original font is a wireframe font.

8.3.5.2.4 RgbColor

Color definition with 3 components.

Table 46 — *RgbColor*

Name	Data Type	Data Description
Red	*Double*	(Required) Red
Green	*Double*	(Required) Green
Blue	*Double*	(Required) Blue

8.3.6 *PRC_TYPE_ASM_FileStructureTree*

Each FileStructure within a PRC file has a FileStructureTree which defines

— the number of parts and part data

— the number of product occurrences and product occurence data

within this FileStructure.

By convention, the data within PRC Files are ordered so that data is defined before it is referenced. In the case of a FileStructureTree, part data is defined before product occurrence data which may refer to it, but within the array of part data, the order of the parts is immaterial. This is not the case for product occurrence data.

Product occurrence data represent an assembly, with a single product occurrence being the root. Each product occurrence (except the root) may be referenced by only one product occurrence. However, each product occurrence may refer to multiple product occurrences. The array of product occurence data should be ordered so that any product occurrence is defined before the product occurrence referring to it.

The FileStructureInternalData defines the index of the root product occurrence and the next available index available to use when assigning unique index (identifiers) to referenceable entities within the FileStructure.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 47 — *PRC_TYPE_ASM_FileStructureTree*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_FileStructureTree*
Base	*ContentPRCBase*	(Required) Base information associated with the entity
part_count	*UnsignedInteger*	(Required) Number of part definitions
Parts	Array <*PRC_TYPE_ASM_PartDefinition*>[**part_count**]	(Required) An array of part definitions
product_count	*UnsignedInteger*	(Required) Number of product occurrences
Products	Array <*PRC_TYPE_ASM_ProductOccurrence*>[**product_count**]	(Required) An array of product occurrences
internal_data	*PRC_TYPE_ASM_FileStructure*	(Required) FileStructureInternalData
user_data	*UserData*	(Required) User defined data

8.3.7 *PRC_TYPE_ASM_FileStructureTessellation*

This type gathers tessellation data of a file structure (See 6.2).

Table 48 — *PRC_TYPE_ASM_FileStructureTessellation*

Name	Data Type	Data Description
	UnsignedInteger	(Required) PRC_TYPE_ASM_FileStructureTesssellation
base	*ContentPRCBase*	(Required) Base information associated with the entity
tess_count	*UnsignedInteger*	(Required) Number_of_tessellations
tess	Array <*PRC_TYPE_TESS*>[**tess_count**]	(Required) Content of all tessellations
user_data	*UserData*	(Required) User defined data

8.3.8 *PRC_TYPE_ASM_FileStructureGeometry*

8.3.8.1 General

This type gathers geometry data of a file structure (See 6.2).

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 49 — *PRC_TYPE_ASM_FileStructureGeometry*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_FileStructureGeometry*
Base	*ContentPRCBase*	(Required) Base information associated with the entity
exact_geometry	*FileStructureExactGeometry*	(Required) Topological context entities and their associated brep bodies
user_data	*UserData*	(Required) User defined data

8.3.8.2 *FileStructureExactGeometry*

The *FileStructureExactGeometry* section consists of an array of "topological contexts". Each "topological context" contains an array of brep bodies contained within that topological context. Every geometrical and topological entity within a FileStructure may only belong to a single topological context.

A pair of indices (topological context, brep body) uniquely identifies a brep body within the topolological entities of a FileStructure.

Table 50 — *FileStructureExactGeometry*

Name	Data Type	Data Description
topo_context_count	*UnsignedInteger*	(Required) Number of topological contexts
topo_contexts	Array <*TopologicalContext*>[topo_context_count]	(Required) Array of topological contexts together with its associated brep bodies

8.3.9 *PRC_TYPE_ASM_FileStructureExtraGeometry*

8.3.9.1 General

The extra geometry data is summary data pertaining to the geometry which can be used to enable partial loading of the file structure without loading the entire geometry. This type gathers summary information of the exact geometry section, by topological context.

Table 51 — *PRC_TYPE_ASM_FileStructureExtraGeometry*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_FileStructureExtraGeometry*
Base	*ContentPRCBase*	(Required) Base information associated with the entity
extra_geom_count	*UnsignedInteger*	(Required) Number of extra geometry contexts
extra_geom	Array <*ExtraGeometry*>[**extra_geom_count**]	(Required)
user_data	*UserData*	(Required) User defined data

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

8.3.9.2 *ExactGeometry*

This is the summary of a topological context.

Table 52 — *ExactGeometry*

Name	Data Type	Data Description
Summary	*GeometrySummary*	(Required) Summary Data
context_data	*ContextGraphics*	(Required) Graphics Context Data

8.3.9.2.1 *GeometrySummary*

8.3.9.2.1.1 General

This describes the summary list of the bodies of the topological context.

The **number_of_bodies** shall be the same as the number of bodies in the context.

Table 53 — *GeometrySummary*

Name	Data Type	Data Description
number_of_bodies	*UnsignedInteger*	(Required) The **number_of_bodies** shall be the same as the number of bodies in the context.
Bodies	Array <*BodyInformation*>[**number_of_bodies**]	(Required) Graphics information specific to each body

8.3.9.2.1.2 *BodyInformation*

This describes the summary information of a body.

The **body_serial_type** shall be the type of the body that is in the topological context or it shall be set to *PRC_ROOT_TYPE* if the body has no geometry. See the section 8.9.15 for details.

Table 54 — *BodyInformation*

Name	Data Type	Data Description
body_serial_type	*UnsignedInteger*	**(Required)**
tolerance	*Double*	(Optional; if **body_serial_type** is *PRC_TYPE_TOPO_BrepDataCompress* or **body_serial_type** is *PRC_TYPE_TOPO_SingleWireBodyCompress* or **body_serial_type** is *PRC_TYPE_TESS_3D_Compress*) The **tolerance** which corresponds to the compression tolerance for the corresponding entity

8.3.9.2.2 *ContextGraphics*

8.3.9.2.2.1 General

This describes the summary list of the graphical attributes of entities within the topological context.

The following loop shows how to traverse a topological context to gather *GraphicInformation*. A *GraphicInformation* is gathered as soon as there is a graphic content provided for a particular entity :

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

```
For (i=0; i<number_of_body; i++) {
    If (body[i] is PRC_TYPE_TOPO_BrepData) {
        For (j=0; j<body[i].number_of_connex; j++) {
            For (k=0; k<body[i].connex[j].number_of_shell; k++) {
                For (l=0; l<body[i].connex[j].shell[k].number_of_face; l++) {
                    Add_to_output(body[i].connex[j].shell[k].face[l])
                }
            }
        }
    }
}
```

number_of_treat_type corresponds to the number of entity types for which *GraphicsInformation* is stored (currently, only *PRC_TYPE_TOPO_Face* is supported, so if there are graphics on some faces, **number_of_treat_type** is 1, else it is 0).

The current graphics as explained in 8.5 is reset prior to writing/reading the context graphics.

Table 55 — *ContextGraphics*

Name	Data Type	Data Description
number_of_treat_type	*UnsignedInteger*	(Required) number_of_treat_type
treat_types	Array <*GraphicsInformation*>[**number_of_treat_type**]	(Required) Graphic information for each type

8.3.9.2.2.2 GraphicsInformation

This describes the particular graphics for one particular type of the topological context.

Table 56 — *GraphicsInformation*

Name	Data Type	Data Description
element_type	*UnsignedInteger*	(Required)
number_of_element	*UnsignedInteger*	(Required) The **number_of_element** represents the number of elements collected during a recursive search on the Context data (including duplicated elements) as shown in 8.5.
element_information	Array <*ElementInformation*>[**number_of_element**]	(Required) Graphics Information for each Element

8.3.9.2.2.3 *ElementInformation*

This describes the particular graphics for one particular element of the topological context.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 57 — *ElementInformation*

Name	Data Type	Data Description
has_graphics	*Boolean*	(Required) **has_graphics** is TRUE if element[i] has graphics
graphic_behavior	*ElementGraphicsBehavior*	(Optional; if **has_graphics** is TRUE) If element[i] has Graphics

8.3.9.2.2.4 *ElementGraphicsBehavior*

This describes graphics for one particular element of the topological context.

Table 58 — *ElementGraphicsBehavior*

Name	Data Type	Data Description
use_context	*Boolean*	(Required) **use_context** is TRUE if use current graphic context
biased_layer_index	*UnsignedInteger*	(Optional; if use_context is FALSE) **layer_index** + 1 where **layer_index** represents the index in the array of layers stored in *FileStructureInternalGlobalData*.
biased_index_of_line_style	*UnsignedInteger*	(Optional; if use_context is FALSE) **index_of_line_style** + 1 where **index_of_line_style** represents the index in the array of styles stored in *FileStructureInternalGlobalData*.
behavior_bit_field	Array <*UnsignedCharacter*>[2]	(Optional; if use_context is FALSE) **behavior_bit_field** **behaviour_bit_field** is an unsigned short integer (2 bytes) . See Section *ContextGraphics* for details.

8.3.10 *PRC_TYPE_ASM_ProductOccurrence*

8.3.10.1 General

A product occurrence defines an assembly tree. In the case of a single part, the product occurrence points directly to a part definition (*PRC_TYPE_ASM_PartDefinition*). In the case of a more complex assembly, a product occurrence is comprised of a list of product occurences.

A product occurrence is comprised of the following data:

— Part definition: A pointer to the corresponding part definition. It can be null.

— Product prototype: A pointer to the corresponding product occurrence prototype. It can be null.

— External data: A pointer to the corresponding external product occurrence. It can be null.

— Children: An array of pointers to the child product occurrences.

The product prototype is the product occurrence of a subassembly or part to be used in the parent assembly. This prototype acts as a template for a given product occurrence, and lets you link to information inside the subpart or assembly, such as geometry. When building assemblies from

AVA Documents
Document Sharing Hub

© ISO 2014 – All rights reserved

Full document details: BS ISO 14739-1:2014

subassemblies, the tree of children of the product occurrence is duplicated from the prototype description. For external data, the tree is only described inside the external data product occurrence. (refer to Figure 1 in Section 5.1)

When the assembly is heterogeneous (originating from different CAD systems), the link is specified through the external data rather than the prototype.

A product occurrence has at most one parent.

Table 59 — *PRC_TYPE_ASM_ProductOccurrence*

Name	Data Type	Data Description
	UnsignedInteger	(Required) *PRC_TYPE_ASM_ProductOccurrence*
Base	*PRCBaseWithGraphics*	(Required)
references_product_occurence	*ReferencesOfProductOccurrence*	(Required)
product_behavior	*Character*	(Required) **product_behavior** represents the various flags for the product. In this version of PRC, only *PRC_PRODUCT_BEHAVIOUR_SUPPRESSED* == 0x01 is used. The other flags should be set to 0.
product_information	*ProductInformation*	(Required)
has_transform	*Boolean*	(Required) TRUE if there is a transformation; else FALSE
Location	*Transformation*	(Optional; if has_transform is TRUE) **location** transforms entities in the product occurrence to its parent space.
entity_ref_count	*UnsignedInteger*	(Required) Number of references
entity_reference	Array <*PRC_TYPE_MISC_EntityReference*>[**entity_ref_count**]	(Required) entity_reference is the referenced entities with possible modifiers towards their nominal definition, which may include location, color, and visibility.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 59 *(continued)*

Name	Data Type	Data Description
Markups	*MarkupData*	(Required) **markups** represents the markups of this product occurrence (as opposed to part definition markups); these are grouped into annotations.
number_of_views	*UnsignedInteger*	(Required) Number of **views**
Views	Array <PRC_TYPE_MKP_View>[number_of_views]	(Required) **views** represents the views which can contain annotations or specific display parameters.
has_filter	*Boolean*	(Required) TRUE if product occurrence has **entity_filter**; else FALSE
entity_filter	*PRC_TYPE_ASM_Filter*	(Optional; if **has_filter** is TRUE) **entity_filter** is a specific filter applied when loading data from product prototypes denoting sub-assemblies.
number_of_display_filters	*UnsignedInteger*	(Required) Number of **display_filters**
display_filters	Array <*PRC_TYPE_ASM_Filter*>[**number_of_display_filters**]	(Required) **display_filters** represents the filters to use for display. Several filters can be specified, but only one can be active (see 8.3.12).
number_of_scene_parameters	*UnsignedInteger*	(Required) Number of **scene_display_parameters**
scene_display_parameters	Array <*PRC_TYPE_GRAPH_SceneDisplayParameter*[**number_of_scene_param**]	(Required) **scene_display_parameters** is reserved for future use.
user_data	*UserData*	(Required) User defined data

8.3.10.2 *ReferencesOfProductOccurrence*

8.3.10.2.1 General

For a given file structure, the product occurrences should be ordered based on the following criteria:

— A product prototype in the same file structure should be stored before any occurrences that use it.

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

— External data in the same file structure should be stored before any occurrences that use it.

— A child occurrence should be stored before its parent.

— A part definition can be referenced several times in the same file structure.

(See to Figure 1 in Section 5.1.)

Table 60 — *ReferencesOfProductOccurrence*

Name	Data Type	Data Description
biased_index_part	*UnsignedInteger*	(Required) **index_part** + 1 where **index_part** represents the index of the part definition in the array of part definitions of the same FileStructure.
biased_index_prototype	*UnsignedInteger*	(Required) **index_prototype** + 1 where **index_prototype** represents the index of the product prototype in the FileStructure product occurrences array.
prototype_in_same_file_structure	*FileIdentifier*	(Optional; if **index_prototype** is NOT -1) **prototype_in_same_file_structure** indicates whether the prototype is in the same *FileStructure*.
biased_index_external_data	*UnsignedInteger*	(Required) **index_external_data** + 1 where **index_external_data** represents the index of the external data.
external_data_in_same_file_structure	*FileIdentifier*	(Optional; if **index_external_data** is NOT -1) **external_data_in_same_file_structure** indicates whether the external data is in the same *FileStructure*.
number_of_child_product_occurrences	*UnsignedInteger*	(Required) Number of child product occurrences
index_child_occurrence	Array <*UnsignedInteger*>[**number_of_child_product_occurrences**]	(Required) **index_child_occurrence**, which is mandatory in the same file structure, represents the index of the child product occurrence.

8.3.10.2.2 *FileIdentifier*

This is the identifier of the FileStructure that the prototype or external data if it is different from the FileStructure that the product occurrence lies in.

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

Table 61 —*FileIdentifier*

Name	Data Type	Data Description
Flag	*Boolean*	(Required) **flag** is TRUE if the entity exists in the same *FileStructure*;
unique_id	*CompressedUniqueId*	(Optional; if flag is FALSE) the identifier of the FileStructure that the entity lies in

8.3.10.3 *ProductInformation*

8.3.10.3.1 General

This is used to save general information associated with a product occurrence.

Table 62 — *ProductInformation*

Name	Data Type	Data Description
unit_from_CAD_file	*Boolean*	(Required) **unit_from_CAD_file** indicates whether the unit is read from the native CAD file.
Unit	*Double*	(Required) **unit** represents the units in mm.
product_information_flags	*Character*	(Required) **product_information_flags** is described in *PRCProductFlag*.
product_load_status	*Integer*	(Required) **product_load_status** is described in *EPRCProductLoadStatus*.

8.3.10.3.2 *PRCProductFlag*

These flags represent characteristics of product occurrences.

A product occurrence can be:

— Default: The product occurrence is the default container, configuration, or view. This means that it is loaded by default in the originating CAD system.

— Internal: when used as a prototype of another product occurrence, this product occurrence does not come from a different physical file. Hence it should belong to the same file structure.

— Container: The product occurrence acts as a repository of child occurrences that do not necessarily have relationships between them. This is useful for situations where a single CAD file can correspond to a whole database of parts and assemblies.

— Configuration: This is a specific arrangement of a product with respect to its whole hierarchy. Some parts may differ or are in a different position. For example, consider the case of an automobile where the steering wheel may be either on the left or right side.

— View: A product occurrence which is a view refers to another product occurrence (its prototype) to denote a particular setting of visibilities and position within the same hierarchy.

If none of these flags is specified, a product occurrence is referred to as regular. If the product occurrence has no parent, it is similar to a configuration. A product occurrence with no parent leads to a different FileStructure, unless it is internal, meaning that it represents a part, subassembly, or assembly

© ISO 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014

hierarchy inside the same FileStructure. If the flag is PRC_PRODUCT_FLAG_REGULAR or PRC_PRODUCT_FLAG_CONTAINER all other flags are ignored. Otherwise, the flags are legal in any combination.

Table 63 — *PRCProductFlag*

Value	Type Name
0x00	PRC_PRODUCT_FLAG_REGULAR
0x01	PRC_PRODUCT_FLAG_DEFAULT
0x02	PRC_PRODUCT_FLAG_INTERNAL
0x04	PRC_PRODUCT_FLAG_CONTAINER
0x08	PRC_PRODUCT_FLAG_CONFIG
0x10	PRC_PRODUCT_FLAG_VIEW

8.3.10.3.3 *EPRCProductLoadStatus*

This represents the status of a loading of a product occurence that may exist in a different file. This status represents the availability of that external resource at the time the PRC file was created.

Table 64 — *EPRCProductLoadStatus*

Value	Type Name	Description
0	KEPRCProductLoadStatus_Error	Unknown status
1	KEPRCProductLoadStatus_NotLoaded	Loading error. For example, there is a missing file
2	KEPRCProductLoadStatus_NotLoadable	Not loadable. For example, something prevents the file from being loaded.
3	KEPRCProductLoadStatus_Loaded	The product was successfully loaded

8.3.10.4 *MarkupData*

This is all the data that are related to Markup for a part or a product.

Table 65 — *MarkupData*

Name	Data Type	Data Description
number_of_linked_items	*UnsignedInteger*	(Required)
linked_items	Array <*PRC_TYPE_MISC_MarkupLinkedItem*>[**number_of_lnked_items**]	(Required) array of linked items
number_of_leaders	*UnsignedInteger*	(Required)
Leaders	Array <*PRC_TYPE_MKP_Leader*>[**number_of_leaders**]	(Required) array of leaders
number_of_markups	*UnsignedInteger*	(Required)

© ISO 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 14739-1:2014