

Australian Standard™

**Industrial automation systems and
integration—Parts library**

**Part 31: Implementation resources:
Geometric programming interface**



This Australian Standard was prepared by Committee IT-006, Information Technology for Industrial Automation and Integration. It was approved on behalf of the Council of Standards Australia on 23 March 2004 and published on 3 June 2004.

The following are represented on Committee IT-006:

Association of Consulting Engineers Australia
Australian Electrical and Electronic Manufacturers Association
CSIRO Centre for Planning and Design
CSIRO Manufacturing & Infrastructure Technology
Department of Defence (Australia)
Institute of Instrumentation, Control and Automation Australia
Institution of Engineers Australia
Monash University
RMIT University
The University of Melbourne

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about Standards can be found by visiting the Standards Web Shop at www.standards.com.au and looking up the relevant Standard in the on-line catalogue.

Alternatively, the printed Catalogue provides information current at 1 January each year, and the monthly magazine, *The Global Standard*, has a full listing of revisions and amendments published each month.

Australian Standards™ and other products and services developed by Standards Australia are published and distributed under contract by SAI Global, which operates the Standards Web Shop.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to the Chief Executive, Standards Australia International Ltd, GPO Box 5420, Sydney, NSW 2001.

This Standard was issued in draft form for comment as DR 04074.

Australian Standard™

Industrial automation systems and integration—Parts library

Part 31: Implementation resources: Geometric programming interface

First published as AS ISO 13584.31—2004.

COPYRIGHT

© Standards Australia International

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Published by Standards Australia International Ltd
GPO Box 5420, Sydney, NSW 2001, Australia

ISBN 0 7337 5910 6

 **Documents**
Document Sharing Hub

PREFACE

This Standard was prepared by the Standards Australia Committee IT-006, Information Technology for Industrial Automation and Integration.

This Standard is identical with, and has been reproduced from, ISO 13584-31:1999, *Industrial automation systems and integration—Parts library*, Part 31: *Implementation resources: Geometric programming interface*.

The objective of this Standard is to provide an application programming interface that enables an application program to generate geometric models that are independent of the target user system. The interface allows portability of programs that describe parametric shape representations of parts families held in an AS ISO 13584 parts library.

This Standard is Part 31 of AS ISO 13584, *Industrial automation systems and integration*, which is published in parts as follows:

Part 1: Overview and fundamental principles

Part 101: Geometrical view exchange protocol by parametric program

Part 20: Logical resource: Logical model of expressions

Part 24: Logical resource: Logical model of supplier library

Part 26: Logical resource: Information supplier identification

Part 31: Implementation resources: Geometric programming interface (this Standard)

Part 42: Description methodology: Methodology for structuring part families

The terms ‘normative’ and ‘informative’ are used to define the application of the annex to which they apply. A normative annex is an integral part of a standard, whereas an informative annex is only for information and guidance.

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover
- (b) In the source text ‘this International Standard’ should read ‘this Australian Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian Standard</i>	
ISO/IEC		AS/NZS	
8824-1	Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation	8824.1	Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation
ISO		AS	
1539	Information technology—Programming languages—FORTRAN	1486	Information technology—Programming languages—FORTRAN
		90	

- | | | | |
|----------|--|----------|---|
| 10303-11 | Industrial automation systems and integration—Product data representation and exchange—Part 11: Description methods: The EXPRESS language reference manual | 10303.11 | Industrial automation systems and integration—Product data representation and exchange, Part 11: Description methods: The EXPRESS language reference manual |
| 10303-41 | Industrial automation systems and integration—Product data representation and exchange—Part 41: Fundamentals of product description and support | 10303.41 | Industrial automation systems and integration—Product data representation and exchange, Part 41: Fundamentals of product description and support |
| 10303-42 | Industrial automation systems and integration—Product data representation and exchange—Part 42: Integrated generic resources: Geometric and topological representation | 10303.42 | Industrial automation systems and integration—Product data representation and exchange, Part 42: Integrated generic resources: Geometric and topological representation |
| 10303-43 | Industrial automation systems and integration—Product data representation and exchange—Part 43: Integrated resources: Representation structures | 10303.43 | Industrial automation systems and integration—Product data representation and exchange, Part 43: Integrated resources: Representation structures |
| 10303-46 | Industrial automation systems and integration—Product data representation and exchange—Part 46: Integrated generic resources: Visual presentation | 10303.46 | Industrial automation systems and integration—Product data representation and exchange, Part 46: Integrated generic resources: Visual presentation |

CONTENTS

	<i>Page</i>
1 SCOPE AND FIELD OF APPLICATION	1
2 NORMATIVE REFERENCES	1
3 TERMS, DEFINITIONS AND ABBREVIATIONS	2
3.1 Terms defined in ISO 13584-10	2
3.2 Other terms and definitions	3
3.3 Abbreviations	3
4 FUNDAMENTAL CONCEPTS	4
4.1 Requirement for parametrics capabilities	4
4.2 Exchange format for parametric shape description	4
4.3 Internal representation of the data created in the receiving CAD system	4
4.4 Library supplier and LMS user responsibility	5
4.5 Compatibility	5
4.6 Geometry representation accuracy	5
5 INTERFACE PRESENTATION	7
5.1 Specification and conformance	7
5.1.1 Allowed levels of implementation	7
5.1.2 Simulation of missing entities	7
5.2 Interface tables	7
5.3 Creation of product model data	8
5.3.1 Reference coordinate system of a view (OVC)	8
5.3.2 Geometrical units in the OVC	9
5.3.3 Content of a view	9
5.3.4 Temporary database	9
5.3.5 Hidden line removal process	10
5.3.6 The representation process	11
5.4 Entities structure	12
5.4.1 Group structure in the TDB	12
5.4.2 Structure of the entities sent to the CAD system	13
5.5 Geometrical or structured entity name	13
5.6 Coordinate system and transformation	13
5.7 Interface error state	14
5.8 Error handling	14
5.8.1 Error handling methodology	14
5.8.2 Error messages	15
6 LOGICAL MODEL OF THE TARGET MODELLING SYSTEM	18
6.1 Geometric representation item	18
6.1.1 api_abstract_schema	20

6.1.1.1	API_ABSTRACT_SCHEMA constant definition: Geometry representation accuracy.....	20
6.1.2	API_ABSTRACT_SCHEMA type definition : Fundamentals of product description and support ..	21
6.1.2.1	Identifier.....	21
6.1.2.2	Label.....	21
6.1.2.3	Text.....	21
6.1.2.4	Length_measure.....	22
6.1.2.5	Plane_angle_measure.....	22
6.1.2.6	Positive_length_measure.....	22
6.1.2.7	Positive_plane_angle_measure.....	22
6.1.2.8	Parameter_value.....	23
6.1.2.9	Message.....	23
6.1.2.10	Reference.....	23
6.1.3	API_ABSTRACT_SCHEMA type definition : Geometric and topological representations	23
6.1.3.1	Dimension_count.....	23
6.1.3.2	Transition_code.....	24
6.1.3.3	Preferred_surface_curve_representation.....	24
6.1.3.4	Trimming_preference.....	25
6.1.3.5	Axis2_placement.....	25
6.1.3.6	Curve_on_surface.....	25
6.1.3.7	Pcurve_or_surface.....	26
6.1.3.8	Trimming_select.....	26
6.1.3.9	Vector_or_direction.....	26
6.1.4	API_ABSTRACT_SCHEMA type definition: Geometry models	26
6.1.4.1	Boolean_operand.....	26
6.1.4.2	Boolean_operator.....	27
6.1.4.3	Csg_primitive.....	27
6.1.4.4	Csg_select.....	27
6.1.4.5	Geometric_set_select.....	28
6.1.5	API_ABSTRACT_SCHEMA type definition: api specific types for structuring.....	28
6.1.5.1	Api_grouped_item.....	28
6.1.5.2	Api_set_item.....	28
6.1.6	API_ABSTRACT_SCHEMA entities definition : Fundamentals of product description and support	29
6.1.6.1	Shape_representation.....	29
6.1.6.2	Group.....	29
6.1.6.3	Group_assignment.....	30
6.1.6.4	External_source.....	30
6.1.6.5	Pre_defined_item.....	30
6.1.6.6	Externally_defined_item.....	31
6.1.7	API_ABSTRACT_SCHEMA entity definition: Representation structures	31
6.1.7.1	Representation_context.....	31
6.1.7.2	Representation_item.....	32
6.1.7.3	Representation.....	32
6.1.7.4	Representation_map.....	34
6.1.7.5	Mapped_item.....	34
6.1.8	API_ABSTRACT_SCHEMA entity definition: Geometric representation structures	35
6.1.8.1	Geometric_representation_context.....	35
6.1.8.2	Geometric_representation_item.....	35
6.1.9	API_ABSTRACT_SCHEMA entity definition: Geometric mathematical entities.....	37
6.1.9.1	Point.....	37
6.1.9.2	Cartesian_point.....	37
6.1.9.3	Direction.....	38
6.1.9.4	Vector.....	38
6.1.9.5	Placement.....	39
6.1.9.6	Axis1_placement.....	40
6.1.9.7	Axis2_placement_2d.....	40
6.1.9.8	Axis2_placement_3d.....	41
6.1.10	API_ABSTRACT_SCHEMA entity definition: Geometric curves entities.....	43
6.1.10.1	Curve.....	43
6.1.10.2	Line.....	43
6.1.10.3	Bounded_curve.....	44
6.1.10.4	Trimmed_curve.....	45

	<i>Page</i>
6.1.10.5	Composite_curve.....47
6.1.10.6	Composite_curve_segment 48
6.1.10.7	Surface_curve 49
6.1.10.8	Composite_curve_on_surface 51
6.1.10.9	Bounded_surface_curve 52
6.1.11	API_ABSTRACT_SCHEMA entity definition: Geometric conic entities.....52
6.1.11.1	Conic.....52
6.1.11.2	Circle.....53
6.1.11.3	Ellipse 54
6.1.11.4	Hyperbola.....56
6.1.11.5	Parabola.....57
6.1.12	API_ABSTRACT_SCHEMA entity definition: api specific basic curves 59
6.1.12.1	Api_line 59
6.1.12.2	Api_circular_arc.....60
6.1.13	API_ABSTRACT_SCHEMA entity definition: api specific conic arcs 60
6.1.13.1	Api_elliptical_arc 61
6.1.13.2	Api_hyperbolic_arc 62
6.1.13.3	Api_parabolic_arc.....62
6.1.14	API_ABSTRACT_SCHEMA entity definition: curve entities 63
6.1.14.1	Polyline 63
6.1.14.2	Api_contour 64
6.1.15	API_ABSTRACT_SCHEMA entity definition: fill area.....66
6.1.15.1	Annotation_fill_area.....67
6.1.16	API_ABSTRACT_SCHEMA entity definition : Geometric surface entities 69
6.1.16.1	Surface.....69
6.1.16.2	Elementary surface.....69
6.1.16.3	Plane.....70
6.1.16.4	Bounded_surface.....70
6.1.16.5	Curve_bounded_surface.....71
6.1.16.6	Boundary_curve 72
6.1.16.7	Outer_boundary_curve 73
6.1.17	API_ABSTRACT_SCHEMA entity definition : api specific surface entities 73
6.1.17.1	Api_planar_surface.....73
6.1.18	API_ABSTRACT_SCHEMA entity definition : Geometric solid entities.....75
6.1.18.1	Solid_model 75
6.1.18.2	Csg_solid.....75
6.1.18.3	Boolean_result.....76
6.1.18.4	Csg_primitive 76
6.1.18.4.1	Sphere.....77
6.1.18.4.2	Right_circular_cone.....77
6.1.18.4.3	Right_circular_cylinder 78
6.1.18.4.4	Torus.....78
6.1.18.4.5	Block.....79
6.1.18.4.6	Right_angular_wedge 79
6.1.18.5	Swept_area_solid 80
6.1.18.6	Extruded_area_solid.....81
6.1.18.7	Revolved_area_solid.....81
6.1.18.8	Half_space_solid 83
6.1.19	API_ABSTRACT_SCHEMA entity definition : api specific entities for structuring 83
6.1.19.1	Api_group.....83
6.1.19.2	Api_group_assignment 84
6.1.19.3	Api_set 84
6.1.19.4	Api_set_assignment.....85
6.2	Visual appearance of geometric representation items 85
6.2.1	API_ABSTRACT_SCHEMA type definition : Visual presentation 86
6.2.1.1	Presentation_style_select.....86
6.2.1.2	Null_style 86
6.2.1.3	Size_select.....87
6.2.1.4	Curve_font_or_scaled_curve_font_select.....87
6.2.1.5	Curve_style_font_select 87
6.2.1.6	Fill_style_select.....88
6.2.2	API_ABSTRACT_SCHEMA type definition : api specific types for visual presentation.....88

	<i>Page</i>
6.2.2.1 Virtual_height_ratio	88
6.2.3 API_ABSTRACT_SCHEMA entities definition : Visual presentation	88
6.2.3.1 Styled_item	88
6.2.3.2 Presentation_style_assignment	89
6.2.3.3 Externally_defined_style	90
6.2.3.4 Curve_style	90
6.2.3.5 Fill_area_style	91
6.2.3.6 Fill_area_style_hatching	92
6.2.3.7 One_direction_repeat_factor	93
6.2.3.8 Colour	94
6.2.3.9 Pre_defined_size	94
6.2.3.10 Pre_defined_curve_font	94
6.2.3.11 Pre_defined_colour	95
6.2.3.12 Annotation_occurrence	95
6.2.3.13 Annotation_fill_area_occurrence	96
6.2.4 API_ABSTRACT_SCHEMA entities definition : externally-defined styles for visual presentation	96
6.2.4.1 Api_externally_defined_point_style	97
6.2.4.2 Api_externally_defined_curve_style	98
6.2.4.3 Api_externally_defined_fill_area_style	99
6.2.4.4 Api_externally_defined_surface_style	99
6.2.5 API_ABSTRACT_SCHEMA entities definition : pre-defined styles for visual presentation	100
6.2.5.1 Api_pre_defined_hatch_width	100
6.2.5.2 Api_pre_defined_hatch_curve_font	101
6.2.5.3 Api_pre_defined_hatch_colour	101
6.2.5.4 Api_pre_defined_occlusion_style	102
6.2.5.5 Api_pre_defined_virtually_sent_style	103
6.3 API_ABSTRACT_SCHEMA function definition	103
6.3.1 API_ABSTRACT_SCHEMA function definition : Geometric and topological representations ...	103
6.3.1.1 Dimension_of	103
6.3.1.2 Associated_surface	104
6.3.1.3 Base_axis	104
6.3.1.4 Build_2axes	105
6.3.1.5 Build_axes	106
6.3.1.6 Orthogonal_complement	106
6.3.1.7 First_proj_axis	107
6.3.1.8 Second_proj_axis	108
6.3.1.9 Cross_product	108
6.3.1.10 Dot_product	109
6.3.1.11 Normalise	110
6.3.1.12 Scalar_times_vector	111
6.3.1.13 Vector_sum	112
6.3.1.14 Vector_difference	113
6.3.1.15 Constraints_composite_curve_on_surface	114
6.3.1.16 Get_basis_surface	115
6.3.1.17 List_to_array	116
6.3.1.18 Make_array_of_array	116
6.3.2 API_ABSTRACT_SCHEMA function definition: Support resources	117
6.3.2.1 Bag_to_set	117
6.3.3 API_ABSTRACT_SCHEMA function definition: Representation structures	118
6.3.3.1 Acyclic_mapped_representation	118
6.3.3.2 Item_in_context	119
6.3.3.3 Using_representations	120
6.3.4 API_ABSTRACT_SCHEMA function definition: api specific functions	121
6.3.4.1 Tree_api_group_structure	121
6.3.4.1.1 Assigned_api_group	123
6.3.4.2 Tree_api_set_structure	123
6.3.4.2.1 Assigned_api_set	124
6.3.4.3 Api_legal_style_number	125
6.4 API_ABSTRACT_SCHEMA global rules	127
6.4.1 Unique_shape_representation	127

	<i>Page</i>
7 INTERFACE FUNCTIONAL SPECIFICATION	127
7.1 Notational conventions	127
7.1.1 Function representation	127
7.1.2 Data type representation.....	129
7.1.3 Entity names and abbreviations.....	129
7.1.4 Function names.....	130
7.2 Logical description of the interface functions and FORTRAN binding.....	131
8 INTERFACE TABLES.....	131
8.1 Interface description table.....	131
8.2 Interface status table	131
9 DIMENSIONS OF INTERFACE IMPLEMENTATION	132
9.1 Minimal dimensions of the different interface buffers and structured data types.....	132
ANNEX A (NORMATIVE) LOGICAL DESCRIPTION OF THE INTERFACE FUNCTIONS AND FORTRAN BINDINGS	133
ANNEX B (NORMATIVE) INFORMATION OBJECT REGISTRATION.....	325
Bibliography.....	326
Index.....	327

Figures

Figure 1 — Absolute coordinate system of a part (parts supplier defined)	9
Figure 2 — Geometric representation items defined in the interface	18
Figure 3 — Axis2 placement 3D.....	43
Figure 4 — Composite curve	48
Figure 5 — Circle.....	54
Figure 6 — Ellipse	55
Figure 7 — Hyperbola.....	57
Figure 8 — Parabola.....	58
Figure 9 — Filling of annotation fill areas.....	68
Figure 10 — Curve bounded surface.....	72
Figure 11 — Right_angular_wedge and its attributes.....	80
Figure 12 — Revolved area solid.....	82
Figure 13 — Fill area style hatching.....	92
Figure 14 — One direction repeat factor	94
Figure A. 1 — Function: Dir_2_Pnt	152
Figure A. 2 — Function: Dir_2_Dir_Angle	154
Figure A. 3 — Function: A1p_Gen	157
Figure A. 4 — Function: A1p_Pnt.....	159

	<i>Page</i>
Figure A. 5 — Function: A2p_3_Pnt.....	161
Figure A. 6 — Function: A2p_2_Dir (in a 3D view)	163
Figure A. 7 — Function: A2p_2_Dir (in a 2D view)	164
Figure A. 8 — Function: A2p_2_Dir_Xy.....	165
Figure A. 9 — Function: Pnt_Cartesian_Relative	170
Figure A. 10 — Function: Pnt_Polar_Relative	173
Figure A. 11 — Function: Pnt_Cylinder_Relative	175
Figure A. 12 — Function: Pnt_Intersection_2_Ent (in a 3D-view)	179
Figure A. 13 — Function: Pnt_Tangential_Arc	181
Figure A. 14 — Function: Pnt_Projection_Ent	184
Figure A. 15 — Function: Pnt_Projection_A2p	186
Figure A. 16 — Function: Lin_2_Pnt	188
Figure A. 17 — Function: Lin_Pnt_Length_Dir	189
Figure A. 18 — Function: Lin_Tangential_Arc.....	191
Figure A. 19 — Function: Lin_Tangential_2_Arc	193
Figure A. 20 — Function: Lin_Chamfer_2_Lin.....	195
Figure A. 21 — Function: Circle_Rad_A2p.....	197
Figure A. 22 — Function: Arc_3_Pnt (in a 3D view)	199
Figure A. 23 — Function: Arc_3_Pnt (in a 2D view)	199
Figure A. 24 — Function: Arc_Rad_2_Angle_A2p.....	201
Figure A. 25 — Function: Arc_Rad_3_Pnt.....	204
Figure A. 26 — Function: Arc_Rad_2_Pnt_A2p	206
Figure A. 27 — Function: Arc_Fillet_2_Ent (lin/lin).....	209
Figure A. 28 — Function: Arc_Fillet_2_Ent (arc/arc)	210
Figure A. 29 — Function: Arc_Tangential_2_Ent.....	213
Figure A. 30 — Function: Arc_Rad_2_Ent.....	217
Figure A. 31 — Function: Arc_3_Ent.....	221
Figure A. 32 — Function: Ellipse_2_Diameter_A2p.....	223
Figure A. 33 — Function: Elc_Gen.....	225
Figure A. 34 — Function: Hyp_Gen	227
Figure A. 35 — Function: Par_Gen	229

	<i>Page</i>
Figure A. 36 — Function: Fsh_Gen.....	236
Figure A. 37 — Function: Hatch_Afa.....	237
Figure A. 38 — Function: Sph_Gen	240
Figure A. 39 — Function: Con_Gen	242
Figure A. 40 — Function: Cyl_Gen	243
Figure A. 41 — Function: Tor_Gen	245
Figure A. 42 — Function: Blk_Gen.....	246
Figure A. 43 — Function: Wdg_Gen	248
Figure A. 44 — Function: Union_Sld	249
Figure A. 45 — Function: Intersection_Sld	251
Figure A. 46 — Function: Difference_Sld	252
Figure A. 47 — Function: Sld_Extrusion	254
Figure A. 48 — Function: Sld_Revolution	255
Figure A. 49 — Function: Sld_Pipe	258
Figure A. 50 — Function: Mirror_Ent (3D view).....	267
Figure A. 51 — Function: Mirror_Ent (2D view).....	267
Figure A. 52 — Function: Dup_Mirror_Ent	269
Figure A. 53 — Function: Chg_Orientation_Ent	276
Figure A. 54 — Function: Chg_Sense_Ent.....	277
Figure A. 55 — Function: Homotetia_Ent.....	278
Figure A. 56 — Function: Start_Angle_Arc.....	289
Figure A. 57 — Function: End_Angle_Arc.....	290
Figure A. 58 — Function: Ref_Sys_3_Pnt (3D view)	292
Figure A. 59 — Function: Ref_Sys_2_Dir (in a 2D view)	294

Tables

Table 1 — Input error messages.....	15
Table 2 — Geometry error messages	16
Table 3 — System error messages	17
Table 4 — Entity structure error messages.....	17
Table 5 — Presentation style error messages	17

	<i>Page</i>
Table 6 — Language binding error messages	17
Table 7 — Externally defined point styles	97
Table 8 — Shapes of the externally defined point styles	97
Table 9 — Externally defined curve styles	98
Table 10 — Externally defined fill area styles.....	99
Table 11 — Externally defined surface style	99
Table 12 — Pre_defined hatch line width	100
Table 13 — Line segment and space lengths for Pre_defined hatch curve font.....	101
Table 14 — Pre_defined hatching colour	101
Table 15 — Pre_defined hidden line style	102
Table 16 — Simple data types	129
Table 17 — Short names for entity types	130
Table 18 — Short names for collections of entity types	130
Table 19 — Abbreviations used for function names	131
Table 20 — Interface description table.....	131
Table 21 — Interface status table.....	132
Table 22 — Dimensions of Interface implementation	132
Table A. 1 — Mapping of logical data types	133
Table A. 2 — List of interface functions according interface level 1.....	135
Table A. 3 — List of interface functions according interface level 2.....	140
Table A. 4 — List of interface functions according interface level 3.....	140
Table A. 5 — Short name strings for ENTNAM	285

INTRODUCTION

ISO 13584 is an International Standard for the computer-interpretable representation and exchange of part library data. The objective is to provide a neutral mechanism capable of transferring parts library data, independent of any application that is using a parts library data system. The nature of this description makes it suitable not only for the exchange of files containing parts, but also as a basis for implementing and sharing databases of parts library data.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 13854 fall into one of the following series: conceptual descriptions, logical resources, implementation resources, description methodology, conformance testing, view exchange protocol, and standardised content. The series are described in ISO 13584-1. This part of ISO 13584 is a member of the (implementation resources) series .

This part of ISO 13584 specifies an interface to enable the creation of product model data inside an user system from an application program that is independent of the target user system.

This interface may be used, outside the context of standardized parts library data, to permit the development of application programs that are independent of the target CAD system. In the context of ISO 10303, this interface may be implemented on the top of the SDAI interface to provide constrained geometry construction facilities.

In the context of parts library data, conforming to the ISO 13584 Standard series, the product model data creation process is an application program provided by parts library suppliers, that creates geometric model inside the user system. The interface ensures its independancy from the target user system.

AUSTRALIAN STANDARD

Industrial automation systems and integration - Parts Library - Part 31: Implementation resources: Geometric programming interface

1 Scope and field of application

This part of ISO 13584 specifies an application programming interface that enables an application program to generate geometric models that are independent of the target user system. The interface allows portability of programs that describe parametric shape representations of parts families held in an ISO 13584 parts library.

The following are within the scope of this International Standard:

- programs to generate geometric representations within a modelling system that are independent of the target system,
- programs that specify geometric representations that are created through constraint-based geometric definitions,
- programs that structure geometric representations created independently of the target system,
- programs that specify presentation style attributes for symbolic visualisation of representations created,
- programs that support technical drawing standard conventions for shape representation, including a 2D hidden line mechanism.

The following are outside the scope of this International Standard:

- The precise control of the image to be displayed on the receiving system devices,
- The precise definition of the data that shall be created on the receiving system,
- The storage of a parametric model on the receiving system.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 13584. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 13584 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references the latest edition of the publication referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 128: 1982	<i>Technical drawings - General principles of presentation.</i>
ISO 1539: 1991	<i>Information technology - Programming languages - FORTRAN.</i>
ISO/IEC 8824-1 ¹⁾	<i>Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.</i>

¹⁾ To be published

- ISO 10303-11: 1994 *Industrial automation systems - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual.*
- ISO 10303-41: 1994 *Industrial automation systems - Product data representation and exchange - Part 41: Fundamentals of product description and support.*
- ISO 10303-42: 1994 *Industrial automation systems - Product data representation and exchange - Part 42: Integrated resources: Geometric and topological representation.*
- ISO 10303-43: 1994 *Industrial automation systems - Product data representation and exchange - Part 43: Integrated resources: Representation structures.*
- ISO 10303-46: 1994 *Industrial automation systems - Product data representation and exchange - Part 46: Integrated generic resources: Visual presentation.*
- ISO 13584-10 *Industrial automation systems - Parts library - Part 10: Conceptual Model of Parts Library.*

3 Terms, definitions and abbreviations

3.1 Terms defined in ISO 13584-10

For the purpose of this part of ISO 13584 , the following terms defined in ISO 13584-10 apply:

Abstract part ;

CAD system ;

EXPRESS ;

Functional view ;

Library Management System (LMS) ;

Library supplier ;

Part ;

Parts library ;

Parts supplier ;

Product ;

Product data ;

Program ;

Representation of a part ;

Supplier ;

Supplier part ;

Supplier library ;

Structure ;

User ;

User library ;

View ;

View control variable ;

View exchange protocol .

3.2 Other terms and definitions

For the purposes of this part of ISO 13584, the following terms and definitions apply.

3.2.1 application programming interface (API)

Set of functions that may be triggered from one program by using the concrete syntax defined in one binding.

3.2.2 binding

description of the concrete syntax that shall be used in a particular programming language to trigger the different functions that constitute an application programming interface.

3.2.3 local coordinate system (LCS)

orthogonal right-handed coordinate system used to orientate and to locate geometrical entities in space. Local Coordinate Systems are modeled by an **axis2_placement** entity.

3.2.4 parameter

variable whose name and type of values are specified.

3.2.5 parametric (shape) model

expression of a parametric shape by means of a data model.

3.2.6 parametric (shape) program

expression of a parametric shape by means of a program referring to an API.

3.2.7 parametric shape

common description of a family of cognate shapes and a set of parameters. A parametric shape specifies a partial function from the domain of the parameters onto the set of shapes.

3.2.8 temporary database

temporary database is a mechanism that permits to store construction or temporary data before transferring to the CAD system.

3.3 Abbreviations

For the purposes of this part of ISO 13584, the following abbreviations apply.

- 2D: Two Dimensional ;
- 3D: Three Dimensional ;
- API: Application Programming Interface ;
- CAD: Computer Aided Design ;
- EPS: Epsilon, see 4.6 ;
- HLI: Hidden Line Involved, see 5.3.5 ;
- LCS: Local Coordinate System ;

- LMS: Library Management System ;
- MAX : Maximal value, see 4.6 ;
- OVC: Object View Coordinate system, see 5.3.1 ;
- SDAI: Standard Data Access Interface ;
- TDB: Temporary Data Base, see 5.3.4.

4 Fundamental concepts

4.1 Requirement for parametrics capabilities

1) The ISO 13584 International Standard shall provide a mechanism that enables the global description of the shapes of all the different parts that belong to the same parts family within an ISO 13584 Library.

EXAMPLE 1 - The ISO 4014 hexagon head bolts Standard [1] specifies thousands of different bolts. Describing separately the shape of each bolt is infeasible.

2) Each global shape description shall be associated with a set of numeric-typed, string-typed or Boolean-typed parameters whose set of values characterise each part of the part family. The mechanism that generates each specific shape out of the global description and from a specific set of values v of the parameters shall be deterministic, i.e., it shall define a partial function f from the domain of the set of parameters D onto the set of shapes S .

$$f : D \mapsto S ; s = f(v).$$

Such a description is called a parametric shape.

EXAMPLE 2 - The global description of the 2D top views of the different bolts of the ISO 4014 Standard may be specified as dependent on two real parameters L and D . For each pair of permitted values (l,d) of L and D , the mechanism shall be able to generate in a deterministic way an unique shape.

3) It is a requirement that a parametric shape shall be specified through graphical user interactions. This implies that the mechanism shall provide for the description of constraint-based geometry, the solver of these constraints being part of the mechanism.

4.2 Exchange format for parametric shape description

1) A program that references an application programming interface (API) may be used for exchanging a global shape description that fulfils the requirements of 4.1. The API will specify the constraint-based geometric functions. The program control structure shall specify the function composition that constitutes the global function. The implementation of the API on a receiving system constitutes the solver of the constraint-based geometric functions. Such a program is called a parametric program.

2) It is assumed that the present technology in the field of Computer Aided Design (CAD) allows for the generation of parametric shapes in terms of a parametric program based on a standardised constraint-based API from an interactively-defined system-specific description of this family of shapes.

NOTE - This assumption clarifies the difference between the exchange format specified in this International Standard (a FORTRAN program that makes some call to a standardised API), and the environment that may be used to create such a description (e.g., an interactive graphic system such as a parametric CAD system).

4.3 Internal representation of the data created in the receiving CAD system

The interface specification shall :

- be precise enough to enable a part supplier to describe the shapes of parts
- avoid any implementation specification to enable portability on any CAD modeller.

In this International Standard, these two goals are achieved by describing a logical model of the target modelling system. This logical model is defined as an information model in the EXPRESS language. Each interface function is specified by reference to this logical model.

4.4 Library supplier and LMS user responsibility.

1) When a part is used in a product, both the representation of the part shape, and the presentation of this shape shall be created by the LMS and sent to the geometric modelling system.

EXAMPLE - If a screw is selected from the LMS by an user, during the insertion into a drawing on the CAD system, the screw shall be presented on the screen in a given colour and a given line width according to the representation selected

2) A library contains geometric descriptions that originate from different suppliers. This library may be used for various application contexts. This interface shall allow the library supplier to specify the shapes of the parts, and the library user to ensure a homogeneous level between the presentations of the parts. In this International Standard, this goal is achieved by allowing logical control of the part supplier on the shape presentation aspect (e.g., choice of a named curve style) and by assuming that:

- through some non-standardised initialisation process of the interface, the LMS user may specify the complete presentation aspect (e.g., width, font and colour values) that corresponds to each logically defined style;
- the shape generated by the LMS is displayed according to the current visualisation of the modelling system.

4.5 Compatibility

1) The representation items created inside a product data model through the interface described in this International Standard shall be exchangeable through an ISO 10303 AP conforming exchange file. All the entities attributes that may not be specified by the library supplier shall be bounded with the entity by the interface with consideration of the initialisation of the interface performed by the library user.

2) If the geometric modelling system supports an ISO 10303-22 SDAI interface [2], the interface specified in this International Standard shall be implemented as an applicative layer on the top of the SDAI interface. This applicative layer must contain :

- the solver for the constraint-based geometric entities definition,
- the default value tables for the attributes, possibly set by the LMS user, should be limited for each entity created through the SDAI.

EXAMPLE - If an arc with a given radius is to be created tangent to two lines, with a style "**plain_solid_line**", the applicative layer contains the solver that computes the tangent circle, its trimming parameters and the table that contains the precise width and colour corresponding to a '**plain_solid_line**'.

4.6 Geometry representation accuracy

Despite the fact that different modelling systems have different numeric accuracy, it shall be possible :

- 1) to ensure that a supplier program will work correctly on any "correct" interface implementation ;
- 2) to ensure that an interface implementation will correctly process a "correct" supplier program.

In this International Standard these two goals are achieved by defining reference numeric bounds for different measures involved in geometric entity definitions.

Three reference numeric bounds are defined :

- 1) **EPS** is the minimal value allowed for certain measures involved in some geometric entity measure.

EXAMPLE 1 - The application program is not allowed to define a line segment whose length is smaller than EPS.

- 2) **MAX** is the maximal value allowed for certain measures involved in some geometric entity measure.

EXAMPLE 2 - The application program is not allowed to define a circular arc whose radius is greater than MAX.

- 3) **ZERO_VALUE** is the maximal value allowed for the (mathematically computed) distance between two points that are asserted to be identical.

EXAMPLE - The application program is not allowed to define a contour (i.e., a closed **composite_curve**) such that the distance between the end point of one **composite_curve_segment**, and the beginning point of the next **composite_curve_segment** is greater than **ZERO_VALUE**.

All these reference numeric bounds are defined in the scaled unit specified for the created geometric representation:

- 1) **view_length_unit** scaled by the **view_length_scale_factor** for **length_measure**.
- 2) **view_angle_unit** for **plane_angle_measure**.

In this International Standard, the following values are defined for the reference numeric bounds:

- 1) **EPS** = 10^{-3}
- 2) **MAX** = 10^{+4}
- 3) **ZERO_VALUE** = 10^{-6}

A program conforming to this International Standard shall fulfil the constraints defined, for each geometric entity, by reference to these numeric bounds. An interface conforming to this International Standard shall be able to process a program conforming to this International Standard.

5 Interface presentation

5.1 Specification and conformance

5.1.1 Allowed levels of implementation

This standard specifies 3 levels of implementation, according to the *geometrical power* of the interface whose values are : 2D, 3D curve, solid. These values are numbered 1, 2 and 3²⁾ . Any interface of power level *i* must contain all the functions of power level *j* for *j* < *i*, hence, it may create views of *geometrical_power_level j* when such a power level is set through view initialisation. Whatever the geometrical power of an interface, a view may also be created with a *geometrical_power_level* equal to 0.

Three levels of interface implementations (1 to 3) have been defined. All the functions have been classified in accordance with these levels and a conforming implementation shall provide, for the selected usage, all the functions of the level it belongs to. The level of an interface may be accessed through an inquire function.

5.1.2 Simulation of missing entities

All the entities defined, for each interface level, in this International Standard shall be (conceptually) implemented in the temporary database. If some entities do not exist in the target product modelling system, they must be simulated by using other available entities. This simulation process is specified, for each entity, in this part of ISO 13584.

5.2 Interface tables

The current characteristics of the interface are stored in interface tables. All the values of these table entries may be interrogated by the application program through inquire functions contained in this part of the ISO 13584 Standard. Two tables are defined:

- 1) The *interface description table* contains all the persistent characteristics of the interface (e.g. *interface_level*, *hidden_line_capability*...). These values may be interrogated but may not be changed by the application program. They are implementation dependent.
- 2) The *interface status table* contains values of modal variables (e.g. visualisation attributes). The initial value of such a variable is defined in this International Standard either as being view dependent by set during interface initialisation process, or as having specific values. The interface status table values may be interrogated and, for all but the view dependent (e.g. *view_length_unit*, *hidden_line*, ...), changed by the application program.

²⁾ Note : In the first version of this standard, level 3 corresponds to the creation of solids with implicit topology (solid primitives, sweeps and Boolean operations). In a later version the creation of explicit topological elements (vertex, edge, face...) may be introduced as a level 4. The former is known as CSG (Constructive Solid Geometry), the later is known as B-Rep (Boundary Representation)

The content of these interface tables are described in clause 8 of this International Standard.

5.3 Creation of product model data

This subclause introduces the concept of geometric model creation within a geometric modelling system from an application program. In the context of parts libraries conforming to the ISO 13584 Standard series it is mainly intended for parts library program developers that develop the programs that create the part model data within a CAD system.

When a functional view is selected in a Library Management System (LMS) a part supplier program is called. This functional view is related to the occurrence of the part called object occurrence, that the part supplier has described.

The role of the part supplier program is to populate this functional view by using the interface functions.

5.3.1 Reference coordinate system of a view (OVC)

A functional view created by using the interface functions consists of **geometric_representation_items**. Each view is composed by the application program within its own **geometric_representation_context**, called the *Object_view_modelling_coordinate_system* (OVC). The application program is independent of the relative positioning of the OVC in any CAD coordinate system. The Library Management System (LMS) is in charge of view initialisation, and it is assumed that after the view initialisation function has been performed, all the **geometric_representation_items** sent to the CAD system in their own OVC will be accurately positioned and/or converted.

The "*View_initialisation*" function should therefore activate some unspecified positioning process. In practice, and according to the particular CAD system philosophy, the positioning process may, e.g.:

- 1) define a new local coordinate system if the CAD system uses an instancing mechanism;
- 2) attach the OVC to the cursor for later positioning ;
- 3) accomplish some interactions with the CAD user to interrogate the position and then initialise a transformation matrix in the interface;
- 4) compute the correct position if it results from the object occurrence positioning and then initialise a transformation matrix in the interface;
- 5) do nothing, the view is created at the origin of the global coordinate system and is positioned afterwards by the CAD user.

When a view is initialised as 2D, the 2D space is assumed to be the x, y plane, hence the z coordinate used in the creation function is meaningless for geometric entities. For such entities the z coordinate shall be equal to zero.

In the context of part library data conforming to the ISO 13584 Standard series, when several part supplier programs refer to different functional views of the same part, the OVC used in these different programs are dependent upon each other. An absolute coordinate system shall be associated with the part by the part supplier. All the part supplier programs that produce 3D functional views of this part shall use this absolute coordinate system as their own OVC. All the part supplier programs that produce 2D functional views of this part shall:

- 1) Specify the 2D functional view that is produced by each part supplier program in conformity with ISO 128, (see **Figure 1**) .
- 2) Select as OVC for each part supplier program the coordinate system that results from the part absolute coordinate system and from the specification of the produced 2D functional view (see **Figure 1**).

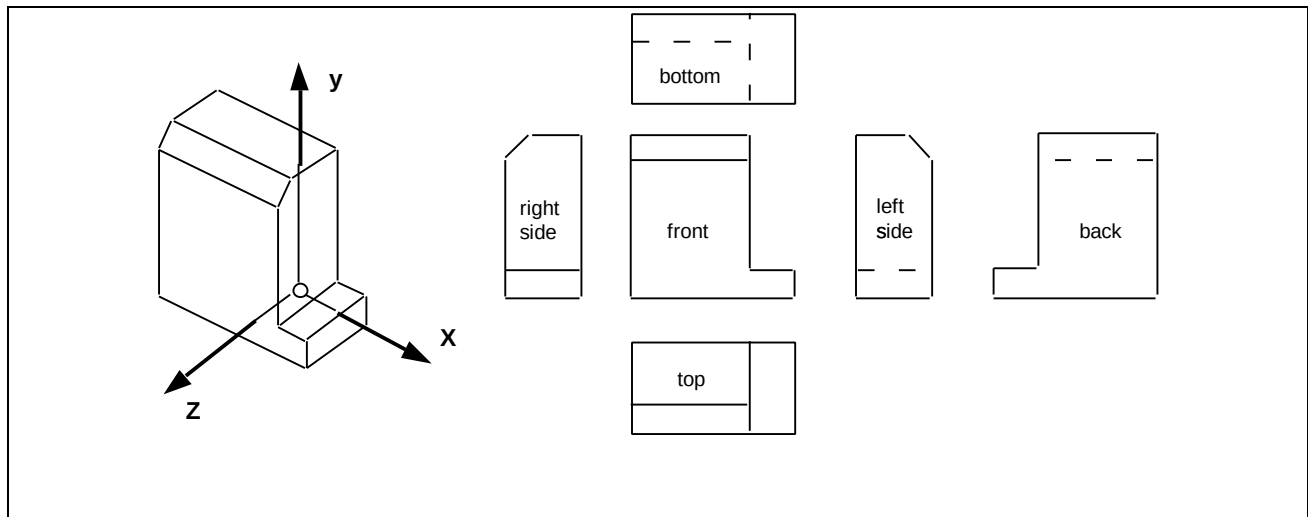


Figure 1 — Absolute coordinate system of a part (parts supplier defined)

5.3.2 Geometrical units in the OVC

The length and plane angle units used in OVC are defined by three interface status table entries: *view_length_unit*, *view_length_scale_factor*, and *view_angle_unit*. The *view_length_unit* entry defines the base length unit used in the view. It may be metre (METRE) or inch (INCH). The *view_length_scale_factor* entry defines the multiplicative scale factor to be applied to the base length unit. The *view_angle_unit* defines the plane angle unit used in the view. It may be radian (RAD), degree (DEG), or grad (GRAD). In this International Standard, the word "OVC unit" refers to either the *view_length_unit* scaled by the *view_length_scale_factor* (*OVC_length_unit*) or to the *view_angle_unit* (*OVC_angle_unit*).

The default values of the OVC unit are specified in clause 8.2 of this International Standard. These default values may be redefined by the part supplier, externally to the program, as part of the functional model the part supplier program belongs to. The default values are set during view initialisation. They may be interrogated, but not changed by the application program. All the geometrical dimensions defined by or returned to the application program are specified in the current OVC units.

The interface ensures correct scaling from OVC units to the CAD system modelling space units. This scaling, called the *OVC-CAD transformation* applies to all the geometric representation items created through interface functions.

5.3.3 Content of a view

The role of the interface functions is to create data inside product modelling system databases. Since product modelling system databases are different from each other, the exact effect of each function may not be described at the physical level. To allow a precise specification of the effect of an interface function, this standard defines the target CAD system database through a logical model defined as an EXPRESS information model (see clause 6). This logical model is assumed to be implemented in some physical way in the target CAD system.

5.3.4 Temporary database

To provide for the creation of intermediate geometry, a *temporary database* (TDB) has to be created. The interface functions allow the creation of geometrical entities either in the Temporary database or as CAD system data. The Temporary data entities may be referenced, modified, used in a geometric construction, or sent to the CAD system. Entities inside the CAD system shall not be referenced. When a temporary database entity is sent to the CAD system, it is no longer referenced as temporary data.

For geometrical entity types having the same visualisation types, attributes may be created both as temporary database entities and in the CAD system.

The visualisation attributes are attached to these entities when they are created on a modal basis, whether this creation occurs in the TDB or in the CAD system. In the TDB, entities attributes may be changed. When an entity is sent from the TDB to the CAD system, visualisation attributes of this entity retain their current values in the TDB.

Temporary data entities may be geometrically moved or duplicated. These geometric manipulations do not change the visualisation attributes: the modified entity, or the duplicated one, preserves the visualisation attributes of the initial one.

The structural relationships available are different in the TDB and in the CAD system. The entities sent to the CAD system are structured by sets. Sets are objects permanently existing in the CAD system database. For this structure a hierarchical set structure is used. For the structure of elements in the TDB, a temporary group structure is used. The group structure may be used to facilitate the creation process of geometric elements. This group structure is also hierarchical. The maximum number of entities that shall be allowed in the TDB by the interface implementation is equal to or greater than that specified in clause 10 of this International Standard.

This document does not specify any implementation form of this temporary database. Only the functionality required from the part supplier functions shall be supported.

5.3.5 Hidden line removal process

For 2D views created by the interface, a hidden line concept is provided.

- 1) In addition to the curve entities, 2D interfaces may create *fill area* that may be "opaque".
- 2) An interface status table entry, called *hidden_line_involved* (HLI) specifies whether or not the curve or fill area entities created through the interface shall be involved in the hidden line removal process. When HLI is equal to *true*, each curve and fill area entity created through an interface function shall be involved in the hidden line removal process and shall have attached an **api_predefined_occlusion_style** visual appearance style. This **api_predefined_occlusion_style** style shall contain a *view_level* attribute which represents the "height" of the entity in some virtual 3D space as a real value, and a *name* attribute that specifies how each entity must be changed if it is hidden. An opaque fill area hides all or part of curve entities that are inside its borders and have a strictly lower value of view level. It shall not hide curve entities with the same value of view level.
- 3) Only the curve and fill area entities created when HLI is equal to *false* shall be sent to the CAD system when the application program requests this transmission with the *Fix_Ent* (fix entities into CAD system) function, or requests their direct creation in the CAD system during view construction. The curve and fill area entities created when HLI is equal to *true* remain in the interface until the hidden line removal process is performed.
- 4) When the application program requests the transmission to the CAD system of an entity that is involved in the hidden line removal process, either with the *Fix_Ent* function, or by requiring its direct creation in the CAD system, this entity is attached to an **api_predefined_virtually_sent_style** and is said to be virtually sent. The **api_predefined_virtually_sent_style** style shall contain an *api_set_name* attribute that shall contain, in the format of a string, the (unique) name of the current open set when the entity is virtually sent.
- 5) At the end of each view construction, the removal process is performed. Only the virtually sent entities are involved in this process. Temporary database entities are not involved.
- 6) When a virtually sent curve entity is partially hidden by a fill area, the visible part shall be presented with the current style of the curve. The parts of the virtually sent curve entity that are hidden are processed according to the *name* attribute of the **api_predefined_occlusion_style** visual appearance style.

If the value of this attribute is :

they are :

no_change

sent without any change ;

dashed

sent as invisible if this capability exists in the CAD system, or else curve entities and fill area borders and hatching are sent as dashed lines ;

invisible

sent as invisible if this capability exists in the CAD system, or else they are not sent.

When a fill area is hidden by another surface, a hidden part removal process is only required by this International Standard on the lines that belong to the fill areas : borders and/or hatching.

The involvement of points is not specified in this International Standard.

The capability of hidden line removal is not mandatory. The *hidden_line_capability* entry of the interface description table states if or if not it is available. The *hidden_line* entry of the interface status table states if or if not the elimination process shall be activated for the next view. The default value of the *hidden_line* entry is set to the value of *hidden_line_capability* (if the capability is available, it is always activated unless it is changed by the application program).

The hidden line removal process may only be activated for views initialised as 2D.

The hidden line removal process shall not change the set structure defined by the application program at the (virtual) sending time. This set structure is recorded in the *api_set_name* attribute of the **api_predefined_virtually_sent_style** style.

5.3.6 The representation process

The role of the interface functions is to create data inside product modelling system databases. The viewing process of such data is assumed to be controlled by the product modelling system and the system user. However, it can be seen that the application program must have some control over the geometrical aspect of entities (for instance to meet the requirements of the applicable technical drafting standard, or to underscore some semantic difference between entities) because the user wishes some similarity between views obtained from various supplier libraries.

These two goals are achieved in the following way (see clause 6.2.4 and clause 6.2.5):

- 1) All the presentation styles are defined either as pre-defined styles or as externally defined styles
- 2) The pre-defined styles are described by this part of ISO 13584. Externally defined styles may be defined either by this part of ISO 13584 or by any part belonging to the exchange series of parts.
- 3) Pre-defined styles or externally defined styles only partially describe the visual appearance of the corresponding style. According to their requirements, they may leave e.g., the colour as implementation dependent
- 4) The interface shall provide a tool to allow the CAD user to set the exact value of all the visual appearance attributes left as implementation dependent each pre-defined or externally defined style.
- 5) When a view exchange protocol referenced in an application program is not supported by some interface implementation, then the first style defined for the current representation item in this part of ISO 13584 shall be used in place of the unknown style and no error shall be reported.

5.4 Entities structure

5.4.1 Group structure in the TDB

In the TDB, entities are gathered into a group defined using the entity, *Entity_structured*. When a function operates on a group, it operates repetitively (and recursively) on each relevant entity of the group. When this function is a duplication function, its result is also a group. This group belongs to the current open group and has the same group structure as the initial one. In this duplication process, the duplicate of an open group shall be a closed group. When the function is a modification function, it shall preserve the existing group structure of the existing entities that are inside the modified group.

Entities shall not be modified and no error shall be reported when a function is triggered on a group that contains geometrical entities not allowed as function input parameters. For instance, if the *Chg_Curve_Style* (Change Presentation Style for Curves) function is triggered on a group that contains points, solids and curve entities, the *curve_style* of the curve entities shall be modified but points and solid entities shall remain unmodified in the same group structure.

The whole TDB is itself a group. It is called the *root group*. This root group is open when the interface is initialised and shall not be closed. Hence, there shall always exist an open group.

With the exception of the root group each entity, geometrical or structured shall belong to exactly 1 group (that may be the root group). Groups are structured according to a hierarchical tree structure. The root of the tree is the root group.

Groups may be :

- created: they belong to the current open group and becomes the current open group;
- reopened: all the entities created in the TDB after the group reopening will belong to this group until its closure;
- closed.

Entities sent to (or created in the) CAD system are removed from the group structure.

To ensure a hierarchical group structure, the open groups are managed through a stack. The top of the stack is the current open group.

When the interface is initialised, the root group is put into the stack. No function shall be allowed to close this group, hence it shall always remain in the stack.

When a group is created (1) it belongs to the current open group and, (2) it is put on the top of the stack. Hence, it becomes the current open group.

Only the group that is at the top of the stack may be closed. In this case, this group is removed from the stack and the new top of the stack become the current open group.

When a group is reopened, it is put on the top of the stack. This does not change the group the reopened group belongs to.

Three functions provide for direct modifications of the group structure. None of these functions change the stack content.

The **remove_ent_grp** (Remove Entity from Group) function allows the removal of an entity (geometrical or structured) from a group. After this function, the entity shall belong to the root group.

The *Gather_Ent_Grp* (Gathering entities into new group) function allows a list of entities (geometrical or structured) to be gathered into a new group. None of these entities shall contain the current open group. All these entities are removed from the group to which they belonged and are put into this new group. This new group shall belong to the current open group.

The *Add_Ent_Grp* (Adding Entity into Group) function allows the addition of an entity (geometrical or structured) to an existing group. This entity shall not contain the group to which it is being added. This entity is first removed from its group and then added to the given group.

Groups are local to the TDB. Their intent is to facilitate geometrical construction. The maximum number of groups that shall be allowed by the interface implementation is equal to or greater than that specified in clause **10** of this International Standard.

5.4.2 Structure of the entities sent to the CAD system

It is assumed that the data stored in the CAD system database are distributed into subsets. Conceptually all the data belong to views. Inside a view, the geometrical data are structured into sets and subsets according to a hierarchical tree structure. The structure to be given to the data sent by the application program is specified to the CAD system in the following manner.

- 1) Before sending any data, a view initialisation must be performed by the LMS. All the data sent to the CAD system between this initialisation and the end of the application program shall belong to the view. A view shall not contain another view.
- 2) The *Open_Set* function opens a set. The set name shall be placed on the top of a set stack, and all the geometrical entities sent to the CAD system shall belong to this set. The set itself is a subset of the previous top of the set stack, or, if the set stack is empty, a subset of the view.

The *Close_Set* function shall only be allowed for a set name that is on the top of the set stack. When the *Close_Set* function is called, this set is closed and its name shall be removed from the stack. If the stack is not empty the top of the stack is the current open set. If the stack is empty there is no open set. A closed set shall never be reopened. The name of each set shall be unique inside a view. The maximum size of the set stack that shall be allowed in the TDB by the interface implementation is equal to or greater than that specified in clause **10** of this International Standard.

The mapping between this conceptual structure and a depth-limited tree structure that may be available on a target CAD system is made as follows. The top level of the CAD tree structure, if any, maps the view structure. The following levels, if they exist, map the first levels of the set-subset tree structure. When a set of the CAD tree structure is terminal (i.e. when it may no longer be subdivided into subsets) all the entities belonging to subsets of the conceptual corresponding set are put into this terminal set.

5.5 Geometrical or structured entity name

To be able to refer to any entity created in the TDB, all the entities created by the interface function are named by a value belonging to some abstract data type called *entity_name_type*. The value of this abstract data type can be either 0 or unknown. When an interface function fails and does not succeed in creating some entity, this function returns 0. When an entity is sent to the CAD system, access to the entity is no longer available and the name of this entity becomes unknown. The unknown value is, in particular, returned by the interface functions when an entity is created directly in the CAD system database. When an interface function is called with zero or unknown entities as arguments, it shall return 0.

All the non zero and non unknown values of entity names returned by interface functions during one session, (i.e. between the time where the LMS triggers an application program and the time where this program returns) shall be unique. The name of an entity shall not be reused, even when the first entity has been sent to the CAD system.

NOTE - In the FORTRAN binding, in FORTRAN, *entity_name_type* is mapped onto INTEGER. The zero value is mapped onto 0. The unknown value is mapped onto a negative value. Hence, only positive integer are accessible entity names.

5.6 Coordinate system and transformation

The interface provides functions to change the reference coordinate system of the OVC modelling space.

Four functions may be used by the application program: *Ref_Sys_3_Pnt*, *Ref_Sys_2_Dir*,

Ref_Sys_Position_Relative and *Ref_Sys_A2p*. All the entities created after such a change, either in the TDB or in the CAD system database, shall be defined with respect to the new reference coordinate system.

To allow the application programmer to preserve the previous reference coordinate system, the *Ref_Sys_A2p* (Reference System by Axis2_placement) function allows the creation of an LCS entity from the current OVC reference coordinate system. Subsequent change of the reference coordinate system to this LCS allows the resetting of the coordinate system of the OVC to its previous value. The coordinate system of the OVC can be reset to its previous value if the reference coordinate system is changed to this LCS.

5.7 Interface error state

A global *error_variable* shall be set when an error condition is detected during execution of an interface function. It is an integer entry from the interface status table, corresponding to the integer error number defined in the function specification. It shall also write in the *error_origin* entry of the interface status table the name of the function that identified the error, and in the *error_text* entry the message associated with the error number. The name of the function shall be the syntactical name in the currently used language (e.g. FORTRAN). The message shall be a translation of the error description given in **5.8.1**. These error variables may be interrogated and reset by the application program.

As long as the *error_variable* is set, the interface shall remain in an error state (*error_state* = true). In this error state, the only interface functions that are allowed and behave as specified in **Annex A** of this International Standard are:

- 1) The inquire functions, and
- 2) the *Reset_Error_State* function,

All the other interface functions are permitted but they do not change anything. They return to the calling application program. When the application program returns while the interface is in an error state, the LMS shall:

- 1) Close all the possible open **sets**,
- 2) Close the open view with *error_state* = true,
- 3) Write to the error file, the *error_variable*, *error_origin* and *error_text* entry values,
- 4) Close the interface.

5.8 Error handling

5.8.1 Error handling methodology

For each interface function, a finite number of error situations is specified that will cause the error variables to be set. Every interface implementation shall support this error checking. The error variables provide an interface between the application program and the standardised interface. The application program may interrogate the error value, interpret the information about the error and reset the *error_variable* to zero, to restore the interface to a non error state (*error_state* = false). The interface error handling strategy is derived from the following classification of errors :

- | | |
|-----------|--|
| Class I | errors resulting in a precisely defined reaction ; |
| Class II | errors resulting in an attempt to save the results or previous operations ; |
| Class III | errors that cause unpredictable results including the crash of the CAD system. |

The interface recognises three situations in which errors are detected :

- Situation A error detected in interface functions ;
- Situation B error detected in functions called from interface (CAD system functions, operating system functions) ;
- Situation C error detected outside the interface.

If errors are detected outside the interface (situation C), either the application program may regain control over the execution, or program execution will be terminated abnormally. In the latter case, results are unpredictable (class III) and may cause the CAD system may crash. If, however, the application program obtains control, it may attempt to return to the LMS to try and close the interface properly (see 5.7). The operations defined in 5.7 may also be performed by the interface itself as a standard error reaction to class II errors.

All errors that are listed explicitly as part of the definition of the interface functions belong to class I. Either they are detected within the interface itself (situation A) or when a function called from the interface has returned control to the corresponding interface function with the appropriate error information (situation B). In all class I cases, the interface sets the *error_variable*, *error_origin*, and *error_text*, to the error values. If the failure occurs during an entity creation function and the entity cannot be created, then the entity name returned by the function is set to zero. If an interface function is invoked with more than one error condition, any one of the relevant error number is set in the error variables.

The *Inq_Error_State* function allows the application program to handle the error. The *Reset_Error_State* function allows the removal of the interface from the error state. To close the open view with *error_state* = true allows the LMS to forewarn the CAD system that a view is wrong .

While in the error-state, the inquire functions behave as specified in their functional description (see annex A), and an action shall not generate a new error. Hence, for inquire functions no errors are specified. An output parameter, called *error_indicator*, is used to report possible difficulty during function performance.

The following error numbers are reserved:

- 1) Unused error numbers less than 1001 are reserved for future standardisation,
- 2) Error numbers 1000 to 2000 are reserved for language binding.

5.8.2 Error messages

Table 1 — Input error messages

error number	error description
1	entity name not defined (zero, or unknown)
2	entity type out of permitted range
3	value for length measure out of permitted range
4	value for plane angle measure out of permitted range
5	integer value out of permitted range
6	string value out of permitted range
7	real value out of permitted range

Table 2 — Geometry error messages

error number	error description
101	attempt to create a degenerated entity
102	magnitude of direction vector out of range [EPS,MAX]
103	distance between two points out of range [EPS,MAX]
104	distance between two contours less than EPS
105	attempt to create a degenerated direction during entity creation
106	attempt to create a degenerated axis2_placement during entity creation
107	attempt to create a degenerated axis1_placement during entity creation
108	attempt to create a degenerated basic curve during entity creation
109	attempt to create a degenerated solid during entity creation
110	attempt to create a point outside the parametric range of curves entity
111	attempt to create a line whose segment length is out of range [EPS,MAX]
112	attempt to create an arc whose segment length is less than EPS
113	attempt to create a self-intersected contour entity
114	attempt to create an overlapping solid
115	given entities are identical
116	given points are linear dependent
117	given directions are parallel
118	given curves entities are parallel/concentric
119	given entities are not in the same plane
120	given cut length too long
121	radius too big/small
122	no intersection of given curves entities
123	an intersection of given contours detected
124	an intersection between axis an plane of surface detected
125	an overlapping of given contours detected
126	axis of revolution not in plane of surface
127	geometrical design is not feasible
128	calculation process for creating a conical arc numerical not stable
129	Approximation process to ensure contour closure failed
130	Boolean operation failed

Table 3 — System error messages

error number	error description
201	temporary database overflow
202	error while sending entity to CAD system
203	function not compatible with implemented interface level
204	function not compatible with current power level
205	maximal number of points per polyline exceeded
206	maximal number of entities per contour exceeded
207	maximal number of inner boundaries exceeded
208	maximal number of groups exceeded
209	maximal number of character per string exceeded
210	group stack overflow
211	set stack overflow
212	usage of entity only allowed for usage within the TDB

Table 4 — Entity structure error messages

error number	error description
301	attempt to close the root group
302	attempt to reopen an already open group
303	entity is member of root group
304	entity contains the current open group
305	attempt to create cyclical group structure
306	name of set not unique
307	attempt to close the root set

Table 5 — Presentation style error messages

error number	error description
401	source of exchange protocol unknown
402	identifier of external style unknown
403	assignment of hatch style failed
404	hidden line occlusion style not attached

Table 6 — Language binding error messages

error number	error description
1001	enumerated value out of range
1002	mismatch of number and list length
1003	mismatch of string length

6 Logical model of the target modelling system

6.1 Geometric representation item

The role of the interface functions is to create representation items either in the TDB or in the CAD system database.

There exists three kind of representation items.

- geometric representation items are geometric or annotation entities used to describe the shape that is created through the interface;
- styles are entities used to describe the visual appearance of the geometric representation items;
- structured entities are used to structure the geometric representation items either in the TDB or in the CAD system database.

The geometric representation items that may be created through the interface functions are classified according the following tree (see **Figure 2**).

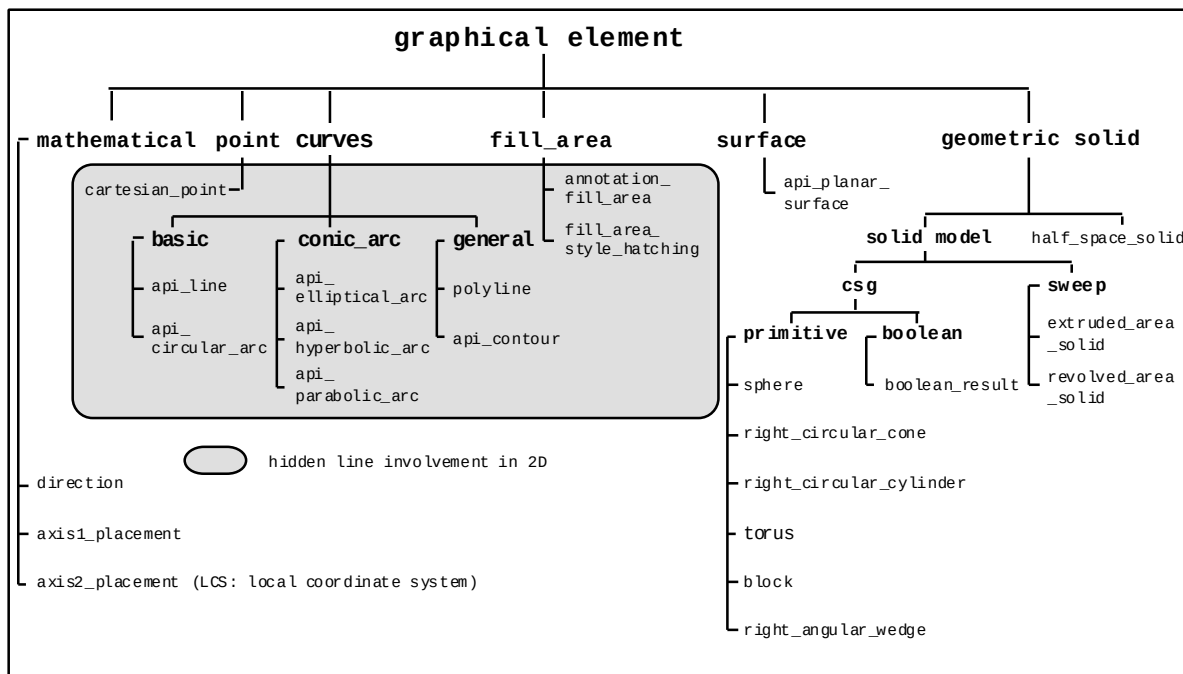


Figure 2 — Geometric representation items defined in the interface

This structure is used both to describe the styles of the various entities and the range of some interface functions.

The implementation of the geometric representation items inside the TDB or in the CAD system is not standardised. Nevertheless, a model of this implementation is defined in this Standard to specify the geometric behaviour of each entity during entity manipulation. This model is defined through an abstract data model specified in EXPRESS. This abstract data model, called **api_abstract_schema**, uses a subset of the generic resources defined in the integrated resource series (Part 41, 42, 43 and 46) of part of ISO 10303 to specify product model data. These resources are referred to as "ISO 10303 generic resources". This abstract data model is not necessarily implemented in the TDB, nor in the CAD system.

entities created through the interface shall behave as if their implementations conform to this abstract data model.

If an interface implementation is intended to create entities conforming to some ISO 10303 application protocol that specialises the ISO 10303 generic resources, the same specialisation shall be applied to the subset of these resources used in the **api_abstract_schema**. Any additional information shall be generated by the interface.

In the definition of the **api_abstract_schema**, the types and entities defined in the ISO 10303 generic resources preserve the names they have in ISO 10303, even if some additional constraints or restriction of allowed subtypes, are added in their definition. These constraints shall be checked by the interface and, if they are satisfied, the created entities shall conform to the definition given in ISO 10303. The additional WHERE RULES that express the constraints specific to **api_abstract_schema** are identified by a name prefixed by the string "api_".

Some entities are also defined by explicitly subtyping entities defined in the ISO 10303 integrated resources. This subtyping is used to specify the range of some interface functions. The name of such entities is prefixed by the string "api_". The subtyping generally consists of restricting the entities defined in the ISO 10303 integrated resources. Such entities may be implemented as an instance of their supertype, or as instances of the specialisation of these supertypes defined in some ISO 10303 Application Protocol.

Finally, some entities are defined by generalisation of entities defined in the generic resources of ISO 10303 by adding new attributes. Such entities are mainly used for defining structure and visual appearance. When the target CAD system is a repository conforming to some ISO 10303 Application Protocol, the interface shall ensure the mapping of these entities onto the resources available within this application protocol. The mapping is textually described in the definition of the api-specific entity.

When some additional constraints are added to the EXPRESS specification of an ISO 10303 generic resource, a Note documents the nature of this restriction. When no restrictions are documented, the resource definition conforms to the definition of the ISO 10303 generic resource.

In the integrated resource series of parts of ISO 10303, some generic resources explicitly refer to other generic resources that are neither used nor referenced in **api_abstract_schema** and whose instance may not appear in a population conforming to the **api_abstract_schema**. These resources are REFERENCED from the relevant EXPRESS schema in the ISO 10303 integrated resources in order to preserve the structure of the generic resources (particularly the existing WHERE RULES) while insuring the formal correctness of the schema. These entities are only referenced in the WHERE RULES duplicated from the ISO 10303 generic resources, not from entities belonging to the **api_abstract_schema**. Therefore, this REFERENCE is only formal.

The entities created by the interface shall not be degenerated. The concept of degeneration shall be independent of any particular interface implementation. In this International Standard, constraints are defined for each entity that may be created by the interface. An entity that does not fulfil these constraints is called a degenerated entity. When a function tries to create a degenerated entity, an error shall occur, the entity shall not be created, and an error message shall be generated.

The degeneration of entities is defined by reference to an absolute minimal allowed value called EPS that is expressed in the (current) *view_length_unit* scaled by the *view_scale_factor*. The value defined for:

$$\text{EPS is } 10^{-3} \text{ view_length_unit} \times \text{view_scale_factor}.$$

When a particular interface implementation does not have the capability to create entities whose extent is as small as EPS for a particular choice of the *view_length_unit* by the application program, an error shall occur during the "set_ovc_length_unit" function, that is a function triggered by the LMS.

The constant, ZERO_VALUE, defines the real values that shall be identified with zero by the interface on any interface implementation. When the distance between two points is less than ZERO_VALUE, these two points shall be considered as identical by the interface.

For instance, when the distance between two trimming points of two **composite_curve_segments** that belong to the same **composite_curve** is less than ZERO_VALUE, the interface shall ensure the continuity of the **composite_curve**, regardless of the required precision of the target CAD system. The ZERO_VALUE is expressed in the (current) *view_length_unit* scaled by the *view_scale_factor*. The value defined for:

ZERO_VALUE is 10^{-6} *view_length_unit* x *view_length_scale_factor*.

NOTE - The definition of these two values acknowledges the impracticability of exact real calculus that lead to ambiguities. The solution retained, that is often used in practice, consists of defining a range [ZERO_VALUE, EPS] of forbidden real values.

6.1.1 api_abstract_schema

This subclause defines requirements for the **api_abstract_schema**. The following EXPRESS declaration introduces the **api_abstract_schema** block and reference the external resources required for formal consistency with the ISO 10303 generic resource definition. Instances of such referenced entities shall not appear in a population created using interface functions.

EXPRESS specification:

```
*)
SCHEMA api_abstract_schema;
  REFERENCE FROM geometry_schema
    (pcurve);
  REFERENCE FROM measure_schema
    (measure_with_unit,
    descriptive_measure);
  REFERENCE FROM presentation_appearance_schema
    (surface_style_usage,
    presentation_style_by_context,
    fill_area_style_colour,
    fill_area_style_tiles,
    pre_defined_hatch_style,
    pre_defined_presentation_style,
    pre_defined_tile_style,
    externally_defined_hatch_style,
    externally_defined_curve_font,
    externally_defined_tile_style,
    curve_style_font,
    curve_style_font_and_scaling,
    text_style,
    point_style,
    symbol_style,
    approximation_tolerance);
(*)
```

NOTE - The schema referenced above can be found in the following part of ISO 10303:

geometry_schema	ISO 10303-42
measure_schema	ISO 10303-41
presentation_appearance_schema	ISO 10303-46

6.1.1.1 API_ABSTRACT_SCHEMA constant definition: Geometry representation accuracy

This subclause declares the named constants used in **api_abstract_schema** as reference numeric bounds for geometry representation accuracy.

EXPRESS specification:

```

*)
CONSTANT
  EPS      : REAL := 1.E-3;
  ZERO_VALUE : REAL := 1.E-6;
  MAX      : REAL := 1.E+4;
END_CONSTANT;
( *

```

NOTE - In the context of the **api_abstract_schema**, **EPS**, **ZERO_VALUE** and **MAX** shall be expressed in *view_length_unit* scaled by *view_length_scale_factor*. for **length_measure** and in *view_angle_unit* for **plane_angle_measure**.

6.1.2 API_ABSTRACT_SCHEMA type definition : Fundamentals of product description and support

This subclause declares the generic type resources defined in ISO 10303-41 that are part of ISO 13584 **api_abstract_schema**.

6.1.2.1 Identifier

An **identifier** is an alphanumeric string that allows an individual thing to be identified. It may not provide natural language meaning.

EXAMPLE - A part number would be an identifier.

EXPRESS specification:

```

*)
TYPE identifier = STRING;
END_TYPE;
( *

```

6.1.2.2 Label

A **label** is the term by which something may be referred to. It is a string that represents the human-interpretable name of something and shall have a natural language meaning.

EXAMPLE - "Smith", "Widget Inc.", and "Materials Test Laboratory" are examples of labels.

EXPRESS specification:

```

*)
TYPE label = STRING;
END_TYPE;
( *

```

6.1.2.3 Text

A **text** is an alphanumeric string of characters that is intended to be read and understood by a human being. It is for information purposes only.

EXPRESS specification:

```

*)
TYPE text = STRING;
END_TYPE;
( *

```

6.1.2.4 Length_measure

A **length_measure** is the value of a distance.

EXPRESS specification:

```
*)
TYPE length_measure = REAL;
END_TYPE;
(*
```

NOTE - In the context of the **api_abstract_schema**, **length_measure** shall be expressed in *view_length_unit* scaled by *view_length_scale_factor*.

6.1.2.5 Plane_angle_measure

A **plane_angle_measure** is the value of an angle in a plane.

EXPRESS specification:

```
*)
TYPE plane_angle_measure = REAL;
END_TYPE;
(*
```

NOTE - In the context of the **api_abstract_schema**, **plane_angle_measure** shall be expressed in *view_angle_unit*.

6.1.2.6 Positive_length_measure

A **positive_length_measure** is a **length_measure** that is greater than zero.

EXPRESS specification:

```
*)
TYPE positive_length_measure = length_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;
(*
```

Formal propositions:

WR1: The value shall be positive.

6.1.2.7 Positive_plane_angle_measure

A **positive_plane_angle_measure** is a **plane_angle_measure** that is greater than zero.

EXPRESS specification:

```
*)
TYPE positive_plane_angle_measure = plane_angle_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;
(*
```

Formal propositions:

WR1: The value shall be positive.

6.1.2.8 Parameter_value

A **parameter_value** is the value that specifies the amount of a parameter in some parameter space.

EXPRESS specification:

```
*)
TYPE parameter_value = REAL;
END_TYPE;
(*
```

6.1.2.9 Message

A **message** is a communication that is addressed to a system in order to trigger some action. The result of such an action is an **externally_defined_item**.

NOTE - The legal values for the message are specified within an application interpreted model.

EXPRESS specification:

```
*)
TYPE message = STRING;
END_TYPE;
(*
```

6.1.2.10 Reference

A **reference** is a means of identifying and retrieving an **externally_defined_item**.

EXPRESS specification:

```
*)
TYPE source_item = SELECT (identifier, message);
END_TYPE;
(*
```

6.1.3 API_ABSTRACT_SCHEMA type definition : Geometric and topological representations

This subclause declares the generic type resources defined in ISO 10303-42 that are part of the **api_abstract_schema**.

6.1.3.1 Dimension_count

A **dimension_count** is a positive integer used to define the coordinate space dimensionality of a **geometric_representation_context**.

EXPRESS specification:

```
*)
TYPE dimension_count = INTEGER;
WHERE
  WR1: SELF > 0;
END_TYPE;
(*
```

Formal propositions:

WR1: A **dimension_count** shall be positive.

6.1.3.2 Transition_code

This type conveys the continuity properties of a composite curve or surface. The continuity referred to is geometric, not parametric continuity.

EXPRESS specification:

```
*)
TYPE transition_code = ENUMERATION OF
  (discontinuous,
   continuous,
   cont_same_gradient,
   cont_same_gradient_same_curvature);
END_TYPE;
(*
```

Enumerated item definitions:

discontinuous: The segments, or patches, do not join. This is permitted only at the boundary of the curve or surface indicating that it is not closed.

continuous: The segments, or patches, join but no condition on their tangents is implied.

cont_same_gradient: The segments, or patches, join, and their tangent vectors, or tangent planes, are parallel and have the same direction at the joint; equality of derivatives is not required.

cont_same_gradient_same_curvature: For a curve, the segments join, their tangent vectors are parallel and in the same direction and their curvatures are equal at the joint; equality of derivatives is not required. For a surface, this implies that the principal curvatures are the same and the principal directions are coincident along the common boundary.

NOTE - In the context of the **api_abstract_schema** there are only composite curves.

6.1.3.3 Preferred_surface_curve_representation

This type is used to indicate the preferred form of representation for a surface curve, that is either a curve in geometric space or in the parametric space of the underlying surfaces.

EXPRESS specification:

```
*)
TYPE preferred_surface_curve_representation = ENUMERATION OF
  (curve_3d,
   pcurve_s1,
   pcurve_s2);
END_TYPE;
(*
```

Enumerated item definitions:

curve_3d: The curve in three-dimensional space is preferred.

pcurve_s1: The first pcurve is preferred.

pcurve_s2: The second pcurve is preferred.

6.1.3.4 Trimming_preference

This type is used to indicate the preferred way of trimming a parametric curve where the trimming is multiply defined.

EXPRESS specification:

```
*)
TYPE trimming_preference = ENUMERATION OF
  (cartesian, parameter,
   unspecified);
END_TYPE;
(*
```

NOTE - In the context of **api_abstract_schema**, the **trimming_preference** is implementation dependent.

Enumerated item definitions:

cartesian: Indicates that trimming by cartesian point is preferred.

parameter: Indicates a preference for the parameter value.

unspecified: Indicates that no preference is communicated.

6.1.3.5 Axis2_placement

This select type collects together both versions of axis2 placement as used in two-dimensional or in three-dimensional Cartesian space. This enables entities requiring this information to reference them without specifying the space dimensionality.

EXPRESS specification:

```
*)
TYPE axis2_placement = SELECT
  (axis2_placement_2d,
   axis2_placement_3d);
END_TYPE;
(*
```

6.1.3.6 Curve_on_surface

A **curve_on_surface** is a curve on a parametric surface. It may be any of the following

- a **pcurve** or
- a **surface_curve**, including the specialised subtypes of **intersection_curve** and **seam_curve**, or
- a **composite_curve_on_surface**.

The **curve_on_surface** select type collects these curves together for reference purposes.

EXPRESS specification:

```
*)
TYPE curve_on_surface = SELECT
  (pcurve,
   surface_curve,
   composite_curve_on_surface);
END_TYPE;
(*
```

6.1.3.7 Pcurve_or_surface

This select type enables a surface curve to identify as an attribute the associated surface or pcurve.

EXPRESS specification:

```
*)
TYPE pcurve_or_surface = SELECT
  (pcurve,
   surface);
END_TYPE;
(*
```

6.1.3.8 Trimming_select

This select type identifies the two possible ways of trimming a parametric curve, by a cartesian point on the curve, or by a REAL number defining a parameter value within the parametric range of the curve.

EXPRESS specification:

```
*)
TYPE trimming_select = SELECT
  (cartesian_point,
   parameter_value);
END_TYPE;
(*
```

6.1.3.9 Vector_or_direction

This type is used to identify the type of entity that can participate in vector computations.

EXPRESS specification:

```
*)
TYPE vector_or_direction = SELECT
  (vector,
   direction);
END_TYPE;
(*
```

6.1.4 API_ABSTRACT_SCHEMA type definition: Geometry models

This subclause declares the generic type resources for geometry models defined in ISO 10303-42 that are part of the **api_abstract_schema**.

6.1.4.1 Boolean_operand

This select type identifies all those types of entities that may participate in a Boolean operation to form a CSG solid.

EXPRESS specification:

```
*)
TYPE boolean_operand = SELECT
  (solid_model,
   half_space_solid,
   csg_primitive,
   boolean_result);
```

```
END_TYPE;
(*
```

6.1.4.2 Boolean_operator

This type defines the three Boolean operators used in the definition of CSG solids.

EXPRESS specification:

```
*)
TYPE boolean_operator = ENUMERATION OF
  (union,
   intersection,
   difference);
END_TYPE;
(*
```

Enumerated item definitions:

union: The operation of constructing the regularised set theoretic union of the volumes defined by two solids.

intersection: The operation of constructing the regularised set theoretic intersection of the volumes defined by two solids.

difference: The regularised set theoretic difference between the volumes defined by two solids.

6.1.4.3 Csg_primitive

This select type defines the set of CSG primitives that may participate in Boolean operations. The CSG primitives are **sphere**, **right_circular_cone**, **right_circular_cylinder**, **torus**, **block** and **right_angular_wedge**.

EXPRESS specification:

```
*)
TYPE csg_primitive = SELECT
  (sphere,
   block,
   right_angular_wedge,
   torus,
   right_circular_cone,
   right_circular_cylinder);
END_TYPE;
(*
```

6.1.4.4 Csg_select

This type identifies the types of entity that may be selected as the root of a CSG tree including a single CSG primitive as a special case.

EXPRESS specification:

```
*)
TYPE csg_select = SELECT
  (boolean_result,
   csg_primitive);
END_TYPE;
(*
```

6.1.4.5 Geometric_set_select

This set select identifies the types of entities that can occur in a **geometric_set**.

EXPRESS specification:

```
*)
TYPE geometric_set_select = SELECT
  (point,
   curve,
   surface);
END_TYPE;
(*
```

6.1.5 API_ABSTRACT_SCHEMA type definition: api specific types for structuring

This subclause declares the api-specific type resources defined for structuring the geometric representation items created by the interface functions.

6.1.5.1 Api_grouped_item

The **api_grouped_item** type specifies those objects that can be part of a **group**

EXPRESS specification:

```
*)
TYPE api_grouped_item = SELECT
  (direction,
   vector,
   placement,
   annotation_fill_area,
   fill_area_style_hatching,
   geometric_set_select,
   solid_model,
   half_space_solid,
   csg_select,
   api_group);
END_TYPE;
(*
```

6.1.5.2 Api_set_item

The **api_set_item** type specifies those objects that can be part of a **api_set**

EXPRESS specification:

```
*)
TYPE api_set_item = SELECT
  (direction,
   vector,
   placement,
   annotation_fill_area,
   geometric_set_select,
   solid_model,
   half_space_solid,
   csg_select,
   api_set);
END_TYPE;
(*
```

6.1.6 API_ABSTRACT_SCHEMA entities definition : Fundamentals of product description and support

This subclause declares the generic entities resources defined in ISO 10303-41 that are part of **api_abstract_schema**.

6.1.6.1 Shape_representation

A **shape_representation** is a specific kind of **representation** that represents a shape.

NOTE 1 - In the context of the **api_abstract_schema**, only one **shape_representation** shall exist. This **shape_representation** corresponds to the shape of the product that is created through the interface in the CAD system database. This is documented by the api-specific associated GLOBAL RULE.

NOTE 2 - In the context of the **api_abstract_schema**, the **context_of_items** shall be a **geometric_representation_context**. This is documented by the api-specific WHERE RULE.

EXPRESS specification:

```
*)
ENTITY shape_representation
  SUBTYPE OF (representation);
WHERE
api_WR1: 'API_ABSTRACT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT' IN
  TYPEOF (SELF\representation.context_of_items);
END_ENTITY;
(*
```

Attribute definitions:

SELFrepresentation.items: A set of **representation_items** that represent the shape of the product

SELFrepresentation.context_of_items: The OVC **representation_context** in which the **items** are related to form the shape of the product.

Formal propositions:

api_WR1: The **context_of_items** of the **shape_representation** shall be a **geometric_representation_context**.

Associated global rule:

The following global rule is associated with this entity and restrict its use or its relationships with other entities:

unique_shape_representation: The **unique_shape_representation** rule requires there exists an unique **shape_representation** entity in the population of the **api_abstract_schema**. This **shape_representation** corresponds to the shape of the product that is created through the interface in the CAD system database

6.1.6.2 Group

A **group** is an identification of a collection of elements.

EXPRESS specification:

```
*)
ENTITY group;
  name      : label;
  description : text;
```

END_ENTITY;
(*

Attribute definitions:

name: The word, or group of words, by which the **group** is referred to.

description: Text that relates the nature of the **group**.

6.1.6.3 Group_assignment

A **group_assignment** is an association of a **group** with product data.

EXPRESS Specification:

```
*)
ENTITY group_assignment
  ABSTRACT SUPERTYPE;
  assigned_group : group;
END_ENTITY;
(*
```

Attribute definitions:

assigned_group: The **group** that is to be associated with the product data.

6.1.6.4 External_source

An **external_source** is the identification of a source of product data that is not the application protocol to which the exchange conforms.

NOTE - In the context of the **api_abstract_schema**, **external_sources** are the locations for **externally_defined_styles** of geometric representation items.

EXPRESS specification:

```
*)
ENTITY external_source;
  source_id : source_item;
END_ENTITY;
(*
```

Attribute definitions:

source_id: The identification of the **external_source**.

6.1.6.5 Pre_defined_item

A **pre_defined_item** is the identification of information that is not explicitly represented in a given exchange but that is defined in the application protocol to which the exchange conforms.

NOTE - In the context of the **api_abstract_schema** some specific entity styles are defined as **pre_defined_items**.

EXPRESS specification:

```
*)
ENTITY pre_defined_item;
```

```

    name : label;
END_ENTITY;
( *

```

Attribute definitions:

name: The words or group of words by which the **pre_defined_item** is referred to.

6.1.6.6 Externally_defined_item

An **externally_defined_item** is the identification of information that is not explicitly represented in a given exchange and that is not defined in the application protocol to which the exchange conforms.

NOTE - In the context of the **api_abstract_schema** entity styles are defined as **externally_defined_items**.

EXPRESS specification:

```

*)
ENTITY externally_defined_item;
    item_id : source_item;
    source : external_source;
END_ENTITY;
( *

```

Attribute definitions:

item_id: The identification of the referent item.

source: An **external_source** that contains the referent item.

6.1.7 API_ABSTRACT_SCHEMA entity definition: Representation structures

This subclause declares the generic entity resources defined in ISO 10303-43 that are part of the **api_abstract_schema**.

6.1.7.1 Representation_context

A **representation_context** is a context in which a collection of **representation_items** are related.

Two **representation_contexts** are separate and have no relationship unless a relationship is separately specified.

EXPRESS specification:

```

*)
ENTITY representation_context;
    context_identifier : identifier;
    context_type       : text;
INVERSE
    representations_in_context : SET [1:?] OF representation
    FOR context_of_items;
END_ENTITY;
( *

```

Attribute definitions:

context_identifier: An identifier of the **representation_context**.

context_type: A description of the type of a **representation_context**.

representations_in_context At least one **representation** shall be associated with each **representation_context**

6.1.7.2 Representation_item

A **representation_item** is an element of product data that participates in one or more **representations** or contributes to the definition of another **representation_item**.

A **representation_item** contributes to the definition of another **representation_item** when it is referenced by that **representation_item**.

EXPRESS specification:

```
* )
ENTITY representation_item;
  name : label;
WHERE
  WR1 : SIZEOF(using_representations(SELF)) > 0;
  api_WR2: SIZEOF(using_representations(SELF)) = 1;
END_ENTITY;
(*
```

NOTE - In the context of the **api_abstract_schema**, every **representation_item** shall be associated with only one **representation_context**. It is documented by **api_WR2**.

Attribute definitions:

name: An identifier of the **representation_item**.

Formal propositions:

WR1: Every **representation_item** shall be associated with at least one **representation_context**.

api_WR2: A **representation_item** shall be referenced by one **representation_context**.

6.1.7.3 Representation

A **representation** is a collection of one or more **representation_items** that are related in a specified **representation_context**. The use of a **representation**, i.e., that is being represented, is not specified in the part of ISO 10303.

This relationship of **representation_item** to **representation_context** is the basis for distinguishing which entities from the set of all **representation_items** are related.

NOTE 1 - Consider the context in which a set of **geometric_representation_items** is related to represent the shape of something. The **items** related in this context are used in this **representation**. All other **geometric_representation_items** are specifically not included. This is the basis for distinguishing which **geometric_representation_items** in the set of **representation_items** are related. This distinction is not otherwise included in the specification of **representation_items**.

The members of the set of **items** plus all **representation_items** indirectly referenced by that set are related to the **context_of_items** by a **representation**. Indirect reference to a **representation_item** occurs when it is referenced through any number of intervening entities, each of type **representation_item**.

NOTE 2 - **representation** relates a **representation_context** to trees of **representation_items** with each tree rooted on one member of the set of **items**. A **representation_item** is one node in the tree, and the reference of one **representation_item** to another is a branch.

The set of **representations** directly referenced as **items**, related to the **context_of_items**, is the **representation**. The **representation_items** indirectly referenced as described above support the definition of the **items** and are all related in the same **representation_context**.

NOTE 3 - In the **representation** of the shape of a cube by a set of **line** entities, the set of **lines** is the only **representation_items** representing the shape. A **line** in turn references a **cartesian_point** and a **direction** that support the **line** definition and are related with each other and the **line** in the referenced **geometric_representation_context**. The shape, however, is not represented by these **cartesian_points** and **directions**.

A **representation** is specified to meet the needs of an application. Often a **representation** is incomplete and does not fully model the concept that is represented.

NOTE 4 - Consider a collection of two dimensional **geometric_representation_items** used to represent the shape of a machined part. It is not a complete description of the shape, but is suitable for certain applications such as computer aided drafting.

One **representation_item** may be related to more than one **representation_context**. Two **representations** are not related solely because the same **representation_item** is referenced directly or indirectly from their sets of **items**.

NOTE 5 - Consider a **surface** that is used in the **representations** of the shape of a casting die and of the shape of the part cast in that die. The **surface** is geometrically founded in two distinct **geometric_representation_contexts**, one for the die and one for the part. However, the **representations** are not related. Instead, each is a separate geometric founding of the common surface. The two unrelated **representations** simply share a common **representation_item**.

The same **representation_item** may be related multiple times to the same **representation_context** as it is used directly or indirectly in several **representations**, each referencing the same **representation_context**. This does not have the meaning that each **representation** is creating a new instance of the same **representation_item** in the same **representation_context**. Rather, each **representation** reasserts one instance of the **representation_item** in the **representation_context** for different uses.

NOTE 6 - Consider two **representations**, each having the same value for **context_of_items**. One is a **representation** of the shape of a cube and indirectly references a **line** as one of its edges. The second simply references the **line** as its **items**. There are not two occurrences of the **line** and its sub-tree of referenced **geometric_representation_items** in the **geometric_representation_context**. Rather the single use of the **line** in that **geometric_representation_context** has been asserted twice, once in each **representation**. The first might exist as representing the shape of the whole cube. The other might exist as representing the shape of an edge of the same cube.

EXPRESS specification:

```
*)
ENTITY representation;
  name      : label;
  items     : SET[1:?] OF representation_item;
  context_of_items : representation_context;
END_ENTITY;
(*
```

Attribute definitions:

name: An identifier of the **representation**.

items: A set of **representation_items** that are related in the **context_of_items**.

context_of_items: A **representation_context** in which the **items** are related to form a **representation** of some concept.

NOTE 7 - In the context of the **api_abstract_schema** all the geometric representation items created through the interface function are founded with the OVC as **context_of_item**.

6.1.7.4 Representation_map

A **representation_map** is the identification of a **representation** and a **representation_item** in that **representation** for the purpose of mapping. The **representation_item** defines the origin of the mapping. The **representation_map** is used as the source of a mapping by a **mapped_item**.

NOTE 1 - The definition of a mapping that is used to specify a new **representation_item** comprises a **representation_map** and a **mapped_item** entity. Without both entities, the mapping is not fully defined. Two entities are specified to allow the same source representation (**representation_map.mapped_representation**) to be mapped into multiple new representations (**mapped_items**).

EXPRESS specification:

```
*)
ENTITY representation_map;
  mapping_origin      : representation_item;
  mapped_representation : representation;
INVERSE
  map_usage           : SET[1:?] OF mapped_item FOR mapping_source;
WHERE
  WR1: item_in_context(SELF.mapping_origin,
    SELF.mapped_representation.context_of_items);
END_ENTITY;
(*
```

Attribute definitions:

mapping_origin: A **representation_item** about which the **mapped_representation** is mapped.

NOTE 2 - Consider the Cartesian mapping of one geometric **representation** to another. The **mapping_origin** might be an **axis2_placement** in the context of **mapped_representation** that defines the position about which it is mapped.

mapped_representation: A **representation** that is mapped to at least one **mapped_item**.

map_usage: The set of one or more **mapped_items** to which the **representation_map** is mapped.

Formal propositions:

WR1: The **mapping_origin** shall be in the **representation_context** of the **mapped_representation**.

6.1.7.5 Mapped_item

A **mapped_item** is the use of a representation, the **mapping_source.mapped_representation**, as it participates in a **representation_map** as a **representation_item**.

NOTE 1 - A **mapped_item** is a subtype of **representation_item**. It enables a representation to be used as a **representation_item** in one or more other representations. The **mapped_item** allows for the definition of a **representation** using other **representations**.

The mapping is achieved through an operator that is implicitly defined by the **mapping_source.mapping_origin** and the **mapping_target** attributes. In this respect, the mapping is specified in the same way as for an **item_defined_transformation** (see ISO 10303-43, 4.4.7 for more information).

EXPRESS specification:

```
*)
ENTITY mapped_item
  SUBTYPE OF (representation_item);
  mapping_source : representation_map;
  mapping_target : representation_item;
```

```

WHERE
  WR1: acyclic_mapped_representation(using_representations(SELF), [SELF]);
END_ENTITY;
( *

```

Attribute definitions:

mapping_source: A **representation_map** that is the source of the **mapped_item**;

mapping_target: A **representation_item** that is the target onto which the **mapping_source** is mapped.

Formal propositions:

WR1: A **mapped_item** shall not be self-defining by participating in the definition of the **representation** being mapped.

NOTE 2 - The details of how any particular mapping is achieved is left to various specialisation' s of **mapped_item** and **representation_map**.

EXAMPLE - Consider the Cartesian mapping of one geometric **representation** to another. The **mapping_source** might be a **representation_map** referencing a **representation** and an **axis_placement** founded in the **geometric_representation_context** of the referenced **representation**. The **mapped_item** might be a reference to this **representation_map** and a second **axis_placement**. The **mapped_item** would then be a **representation_item** that is a mapping of the referenced **representation** such that the **representation_map.mapping_origin** is overlaid onto the **mapped_item.mapping_target**.

6.1.8 API_ABSTRACT_SCHEMA entity definition: Geometric representation structures

This subclause declares generic entity resources defined in ISO 10303-42 for geometric representation structures that are part of the **api_abstract_schema**.

6.1.8.1 Geometric_representation_context

A **geometric_representation_context** is a **representation_context** in which **geometric_representation_items** are geometrically founded.

A **geometric_representation_context** is a distinct coordinate space, spatially unrelated to other coordinate spaces except as those coordinate spaces are specifically related by an appropriate transformation.

EXPRESS specification:

```

*)
ENTITY geometric_representation_context
  SUBTYPE OF (representation_context);
  coordinate_space_dimension : dimension_count;
END_ENTITY;
( *

```

Attribute definitions:

coordinate_space_dimension: The integer **dimension_count** of the coordinate space that is the .

NOTE - In the context of the **api_abstract_schema** the OVC constitutes the **geometric_representation_context** in which all the geometric representation items are geometrically founded. The **dimension_count** may be 2 or 3 according to the *geometrical_power_level* of the current open view.

6.1.8.2 Geometric_representation_item

A **geometric_representation_item** is a **representation_item** that has the additional meaning of having geometric position or orientation or both. This meaning is present by virtue of:

— being a **cartesian_point** or a direction;

- referencing directly a **cartesian_point** or a **direction**;
- referencing indirectly a **cartesian_point** or a **direction**.

NOTE 1 - An indirect reference to a **cartesian_point** or **direction** means that a given **geometric_representation_item** references the **cartesian_point** or **direction** through one or more intervening attributes. In many cases this information is given in the form of an **axis2_placement**.

EXAMPLE 1 - Consider a circle. It gains its geometric position and orientation by virtue of a reference to **axis2_placement** that in turn references a **cartesian_point** and several **directions**.

EXAMPLE 2 - A **manifold_solid_brep** is a **geometric_representation_item** that through several layers of **topological_representation_items**, references **curves**, **surfaces** and **points**. Through additional intervening entities, **curves** and **surfaces** reference **cartesian_point** and **direction**. See the EXPRESS definition of **manifold_solid_geometry**, **topological_representation_item** and **surface** in ISO 10303-42.

NOTE 2 - The intervening entities, that are all of type **representation_item**, need not be of subtype **geometric_representation_item**. Consider the **manifold_solid_brep** case from the above example. One of the intervening levels of **representation_item** is a **close_shell**. This is a **topological_representation_item** and does not require a **geometric_representation_context** in its own right. When used as part of the definition of a **manifold_solid_brep** that itself is a **geometric_representation_item**, it is founded in a **geometric_representation_context**. See the EXPRESS definition of **close_shell** in ISO 10303-42.

NOTE 3 - A **geometric_representation_item** inherits the need to be related to a **representation_context** in a **representation**. The rule **compatible_dimension** ensures that the **representation_context** is a **geometric_representation_context**. When in the context of geometry, this relationship causes the **geometric_representation_item** to be geometrically founded. See the definition of the term geometrically founded in ISO 10303-42.

EXPRESS specification:

```

*)
ENTITY geometric_representation_item
  SUPERTYPE OF (ONEOF(point, direction, vector, placement, curve,
    annotation_fill_area, surface, solid_model,
    boolean_result, sphere, right_circular_cone,
    right_circular_cylinder, torus, block,
    right_angular_wedge, half_space_solid,
    fill_area_style_hatching,
    one_direction_repeat_factor))
  SUBTYPE OF (representation_item);
DERIVE
  dim : dimension_count := dimension_of(SELF);
WHERE
  api_wr1: SIZEOF (QUERY (using_rep <* using_representations (SELF) |
    NOT ('API_ABSTRACT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT' IN
    TYPEOF (using_rep.context_of_items)))) = 0;
END_ENTITY;
(

```

Attribute definitions:

dim: The coordinate **dimension_count** of the **geometric_representation_item**.

NOTE 3 - The **dim** attribute is derived from the **coordinate_space_dimension** of a **geometric_representation_context** in which the **geometric_representation_item** is geometrically founded. See the definition of the term geometrically founded in ISO 10303-42.

NOTE 4 - A **geometric_representation_item** is geometrically founded in one or more **geometric_representation_contexts**, all of them having the same **coordinate_space_dimension**. See the rule **compatible_dimension** in ISO 10303-42 section 4.5.1.

NOTE 5 - In the context of the **api_abstract_schema** all the geometric representation items are subtypes of **geometric_representation_item**.