ISO/TC 184/SC 4
Industrial data

Email of committee manager: trippdsc4@gmail.com
Secretariat: ANSI (United States)

Supplementary Directives - Rules for the structure and drafting of SC 4 standards for industrial data, edition 5

Replaces:	N 2412
Document type:	Other committee document
Date of document:	2020-07-02
Expected action:	INFO
Background:	This document is a draft for comments of an overall update to the SC 4 supplementary directives that are used to conform SC 4 standards. Committee members will be invited to comment via CIB ballot to review prior to an approval ballot to be scheduled later in 2020.
Committee URL:	https://isotc.iso.org/ecom/livelink/open/tc184sc4

ISO/TC 184/SC 4 N3485
Supersedes ISO/TC 184/SC 4 N2412
2020-06-24

ISO/TC 184/SC 4 STANDING DOCUMENT

Technical Committee 184 for Industrial Automation Systems and Integration
Subcommittee 4 for Industrial Data

SC 4 Supplementary directives — Rules for the structure and drafting of SC 4 standards for industrial data

Edition 5

DRAFT: For Comment

This document is being circulated as a draft for comments in preparation of replacing the SC 4 Supplementary directives — Rules for the structure and drafting of SC 4 standards for industrial data
(SC 4 N2412).

This document is intended for use by editors of SC 4 standards, and by developers of supporting tools such as DTDs, style sheets, XML components, XSLT scripts and templates related to SC 4 standards.

Individuals may use the contents of this document for the purposes of developing ISO standards and other documents.

SC 4 standing documents specify procedures, methods, and guidelines for the development of SC 4 standards for industrial data. SC 4 standing documents, together with additional documents such as templates and checklists are available at **SC 4 Standing Documents Folder.**[1]

Major revisions

Document	Date	Notes
SC 4 N3485	2020-06-24	SC4 Supplementary directives — Rules for the structure and drafting of SC4 standards for industrial data, Edition 5
SC 4 N2412	2008-12-16	SC4 Supplementary directives — Rules for the structure and drafting of SC4 standards for industrial data
SC 4 N1217	2002-11-17	SC4 Supplementary directives — Rules for the structure and drafting of SC4 standards for industrial data
SC 4 N1191	2001-08-08	SC4 Supplementary directives — Rules for the structure and drafting of SC4 standards for industrial data
SC 4 N858	1999-04-29	Draft Supplementary directives for the drafting and presentation of ISO 10303, second edition – submitted for SC 4 standing document ballot
SC 4 N537	1997-03-30	Supplementary directives for the drafting and presentation of ISO 10303 – approved by SC 4 for all ISO 10303

This document will be reviewed again not later than June 2023 or immediately if any change in the ISO/IEC Directives, Part 2 has potential consequences for the content of this document.

[1] https://isotc.iso.org/livelink/livelink?func=ll&objId=11568437&objAction=browse&viewType=1

Contents

Tables of figures

Tables of tables

Foreword

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

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

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

The following standing documents provide guidelines for developing International Standards produced by ISO/TC 184/SC 4:

— ISO/TC 184/SC 4 Handbook (ISO/TC 184/SC 4 N3263, 2018, available at https://isotc.iso.org/livelink/livelink/properties/20850151);

— ISO/TC 184/SC 4 Quality manual (ISO/TC 184/SC 4 N1110, SC 4 Quality manual, 2000);

— ISO/TC 184/SC 4 Supplementary directives (this document).

The following standing documents provide additional guidelines for developing and publishing SC 4 standards:

— Standard industrial data framework;

— Common resources and guidelines for the identification of common resources;

— Procedures for transposing externally developed specifications into ISO deliverables.

The main changes are listed below:

— deletion of all the rules that are already provided by the ISO/IEC Directives, Part 2, 2018;

— update of the SC 4 requirements;

— update, as examples, of the required text given in this document;

— deletion of the duplicated content that is already provided by any other SC 4 standing document.

The detailed changes compared to the previous edition are identified in Annex C.

This document cancels and replaces the SC 4 Supplementary directives — Rules for the structure and drafting of SC 4 standards for industrial data (N2412) and is applicable with immediate effect to all SC 4 projects.

Introduction

This document specifies requirements for the content, layout, and style for the standards developed by ISO/TC 184/SC 4.

This document augments the 2018 edition of ISO/IEC Directives, Part 2 and does not provide the authority to contravene the requirements of that or any subsequent edition of those directives.

The content of this document is structured as follows:

— characteristics of an SC 4 standard (see Clause 4);

— requirements for the structure and content of parts of SC 4 standards (see Clause 5);

— EXPRESS presentation style (see Clause 6);

— ISO 8000 series structure and content requirements (see Clause 7);

— ISO 10303 series structure and content requirements (see Clause 8);

— ISO 13584 series structure and content requirements (see Clause 9);

— ISO 15926 series structure and content requirements (see Clause 10);

— ISO 18876 series structure and content requirements (see Clause 11).

This document includes requirements specific to the standards developed by SC 4. Some of these requirements apply to all SC 4 standards; others apply to specific series. The ISO/IEC Directives Part 2 and these supplementary directives provide the set of rules for creating an SC 4 standard.

This document uses the following conventions to delineate text that is to be included in an SC 4 standard (boilerplate) from the rest of the text.

— Examples of required text, as shown below, are enclosed in a light-yellow box. See 5.3 for implementing the required text.

lorem ipsum …

— Information that is to be supplied by the project editor is specified within angle brackets (< … >).

EXAMPLE <**list of in-scope elements**>

— Examples that show specific formats or layouts to be used in SC 4 standards, as shown below, are distinguished from other text by a light green background.

lorem ipsum …

— Examples of EXPRESS layout, as shown below, is enclosed in light blue box.

lorem ipsum …

This document conforms to its own requirements where they are applicable. However, the format and layout of this document are not intended to be a model for the format and layout of an SC 4 standard; where there is any discrepancy between the text of this document and its format and layout, the text has precedence.

1 Scope

This document specifies requirements for the content, layout, and style for the standards developed by ISO/TC 184/SC 4. These standards include the following:

— ISO 8000, *Data quality*;

— ISO 10303, *Product data representation and exchange*;

— ISO 13584, *Parts library*;

— ISO 14306, *JT file format specification for 3D visualization*;

— ISO 15531, *Industrial manufacturing management data*;

— ISO 15926, *Integration of life-cycle data for process plants including oil and gas production facilities*;

— ISO/PAS 17506, *COLLADA digital asset schema specification for 3D visualization of industrial data*;

— ISO 18629, *Process specification language*;

— ISO 18828, *Standardized procedures for production systems engineering*;

— ISO 18876, *Integration of industrial data for exchange, access, and sharing*;

— ISO 20534, *Formal semantic models for the configuration of global production networks*;

— ISO 22745, *Open technical dictionaries and their application to master data*;

— ISO 23247, *Digital twin framework for manufacturing*;

— ISO 23301, *STEP geometry services*;

— ISO 23952, *Quality information framework, QIF*;

— ISO 24464, *Visualization elements of digital twins*;

— ISO 29002, *Exchange of characteristic data*.

This document specifies the elements that make up SC 4 standards, identifies the location and mechanism for the wording to be used to prepare those elements, and sets out rules for the appearance of those elements when these have not already being defined in the ISO/IEC Directives, Part 2, 2018 or by any other SC 4 standing document.

NOTE In this document, "SC 4 standards" refers to any of the deliverables of SC 4 projects that are published by ISO. These include International Standards, Technical Specifications, Publicly Available Specifications, and Technical Reports. As Technical Reports are informative and cannot contain requirements, the required text has to be adapted as appropriate.

The following are within the scope of this document:

— specification of rules and guidelines that apply to normative elements;

— specification of rules and guidelines that apply to informative elements;

— guidelines for formatting the elements of the standard.

The following is outside the scope of this document:

— requirements on the technical content of SC 4 standards.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC Directives, Part 2, 2018 — *Principles and rules for the structure and drafting of ISO and IEC documents*

ISO/IEC Directives, Part 1 and Consolidated ISO Supplement, 2019 — *Procedures specific to ISO*

The ISO Simple template[2]

ISO/TC 184/SC4 N1110[3] — *SC 4 Quality Manual*

ISO 10303-1, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles*

ISO/TC 184/SC 4 N1337[2] — *Application module development points within the application protocol development process*

ISO/TC 184/SC 4 N1375[2] — *Common resources and guidelines for the identification of common resources*

ISO/TC 184/SC 4 N1548[2] — *Guidelines for the format and layout of SC4 standards using HTML*

ISO/TC 184/SC 4 N1863[2] — *Guidelines for the content of application protocols that use application modules*

ISO/TC 184/SC 4 N1685[2] — *Guidelines for the content of application modules*

ISO/TC 184/SC 4 N2661[2] — *Guidelines for the development of mapping specifications*

ISO/TC 184/SC 4 N3422 — *STEP extended architecture high-level description*

ISO/TC 184/SC 4 N3425 — *STEP extended architecture publication design*

3 Terms and definitions

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

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

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

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

3.1 Terms related to document type

3.1.1
document

ISO or IEC standardization draft or publication

EXAMPLE *International Standards* (3.1.4), *Technical Specifications* (3.1.5), *Publicly Available Specifications* (3.1.6), *Technical Reports* (3.1.8) and *Guides* (3.1.7).

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.1.1]

[2] available at https://isotc.iso.org/livelink/livelink/properties/17851245
[3] These documents will be updated, please check the **SC 4 Standing Documents Folder** available at https://isotc.iso.org/livelink/livelink?func=ll&objId=11568437&objAction=browse&viewType=1

3.1.2
standard

document (3.1.1), established by consensus and approved by a recognized body, that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context

Note 1 to entry: Standards should be based on the consolidated results of science, technology and experience, and aimed at the promotion of optimum community benefits.

[SOURCE: ISO/IEC Guide 2:2004, 3.2]

3.1.3
international standard

standard (3.1.2) that is adopted by an international standardizing/standards organization and made available to the public

[SOURCE: ISO/IEC Guide 2:2004, 3.2.1.1]

3.1.4
International Standard

international standard (3.1.3) where the international standards organization is ISO or IEC

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.1.3]

3.1.5
Technical Specification
TS

document (3.1.1) published by ISO or IEC for which there is the future *possibility* (3.3.6) of agreement on an *International Standard* (3.1.4), but for which at present

— the required support for approval as an International Standard cannot be obtained,

— there is doubt on whether consensus has been achieved,

— the subject matter is still under technical development, or

— there is another reason precluding immediate publication as an International Standard

Note 1 to entry: The content of a Technical Specification, including its annexes, may include *requirements* (3.3.3).

Note 2 to entry: A Technical Specification is not allowed to conflict with an existing International Standard.

Note 3 to entry: Competing Technical Specifications on the same subject are permitted.

Note 4 to entry: Prior to mid-1999, Technical Specifications were designated as *Technical Reports* (3.1.8) of type 1 or 2.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.1.5]

3.1.6
Publicly Available Specification
PAS

document (3.1.1) published by ISO or IEC to respond to an urgent market need, representing either

a) a consensus in an organization external to ISO or IEC, or

b) a consensus of the experts within a working group

Note 1 to entry: A Publicly Available Specification is not allowed to conflict with an existing *International Standard* (3.1.4).

Note 2 to entry: Competing Publicly Available Specifications on the same subject are permitted.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.1.6]

3.1.7
Guide

document (3.1.1) published by ISO or IEC giving rules, orientation, advice or *recommendations* (3.3.4) relating to international standardization

Note 1 to entry: Guides can address issues of interest to all users of documents published by ISO and IEC.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.1.7]

3.1.8
Technical Report
TR

document (3.1.1) published by ISO or IEC containing collected data of a different kind from that normally published as an *International Standard* (3.1.4) or *Technical Specification* (3.1.5)

Note 1 to entry: Such data may include, for example, data obtained from a survey carried out among the national bodies, data on work in other international organizations or data on the "state of the art (**Error! Reference source not found.**)" in relation to standards of national bodies on a particular subject.

Note 2 to entry: Prior to mid-1999, Technical Reports were designated as Technical Reports of type 3.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.1.8]

3.2 Terms related to elements of a document

3.2.1
normative element

element that describes the scope of the *document* (3.1.1) or sets out *provisions* (3.3.1)

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.2.1]

3.2.2
informative element

element intended to assist the understanding or use of the *document* (3.1.1) or that provides contextual information about its content, background or relationship with other documents

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.2.2]

3.2.3
mandatory element

element that has to be present in a *document* (3.1.1)

EXAMPLE The Scope is an example of a mandatory element.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.2.3]

3.2.4
conditional element

element that is present depending on the *provisions* (3.3.1) of the particular *document* (3.1.1)

EXAMPLE The symbols and abbreviated terms clause is an example of a conditional element.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.2.4]

3.2.5
optional element

element that the writer of a *document* (3.1.1) may choose to include or not

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.2.5, modified, EXAMPLE removed]

3.3 Terms related to provisions

3.3.1
provision

expression in the content of a normative *document* (3.1.1) that takes the form of a *statement* (3.3.2), an instruction, a *recommendation* (3.3.4) or a *requirement* (3.3.3)

Note 1 to entry: These forms of provision are distinguished by the form of wording they employ; e.g. instructions are expressed in the imperative mood, recommendations by the use of the auxiliary "should" and requirements by the use of the auxiliary "shall".

[SOURCE: ISO/IEC Guide 2:2004, 7.1 (2)]

3.3.2
statement

expression, in the content of a *document* (3.1.1), that conveys information

Note 1 to entry: Table 5 of ISO/IEC Directives, Part 2, 2018 specifies the verbal forms for indicating a course of action permissible within the limits of the document. Table 6 of ISO/IEC Directives, Part 2, 2018 specifies the verbal forms to be used for statements of *possibility* (3.3.6) and *capability* (3.3.7).

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.2, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.3.3
requirement

expression, in the content of a *document* (3.1.1), that conveys objectively verifiable criteria to be fulfilled and from which no deviation is permitted if conformance with the document is to be claimed

Note 1 to entry: Modified, Requirements are expressed using the verbal forms specified in Table 3 of ISO/IEC Directives, Part 2, 2018.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.3, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.3.4
recommendation

expression, in the content of a *document* (3.1.1), that conveys a suggested possible choice or course of action deemed to be particularly suitable without necessarily mentioning or excluding others

Note 1 to entry: Recommendations are expressed using the verbal forms specified in Table 4 of ISO/IEC Directives, Part 2, 2018.

Note 2 to entry: In the negative form, a recommendation is the expression that a suggested possible choice or course of action is not preferred but it is not prohibited.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.4, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.3.5
permission

expression, in the content of a *document* (3.1.1), that conveys consent or liberty (or opportunity) to do something

Note 1 to entry: Permissions are expressed using the verbal forms specified in Table 5 of ISO/IEC Directives, Part 2, 2018.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.5, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.3.6
possibility

expression, in the content of a *document* (3.1.1), that conveys expected or conceivable material, physical or causal outcome

Note 1 to entry: Possibility is expressed using the verbal forms specified in Table 6 of ISO/IEC Directives, Part 2, 2018.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.6, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.3.7
capability

expression, in the content of a *document* (3.1.1), that conveys the ability, fitness, or quality necessary to do or achieve a specified thing

Note 1 to entry: Capability is expressed using the verbal forms specified in Table 6 of ISO/IEC Directives, Part 2, 2018.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.7, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.3.8
external constraint

constraint or obligation on the user of the *document* (3.1.1) (e.g. laws of nature or particular conditions existing in some countries or regions) that is not stated as a *provision* (3.1.1) of the document

Note 1 to entry: External constraints are referred to using the verbal form specified in Table 7 of ISO/IEC Directives, Part 2, 2018.

Note 2 to entry: Use of the word "must" does not imply that the external constraint referred to is a *requirement* (3.3.3) of the document.

[SOURCE: ISO/IEC Directives, Part 2:2018, 3.3.8, modified, "of ISO/IEC Directives, Part 2, 2018" added to Note 1 to entry.]

3.4 Terms related to SC 4 standards

3.4.1
application

one or more processes creating or using product data

[SOURCE: ISO/DIS 10303-1:2020, 3.1.5]

3.4.2
application interpreted construct
AIC

logical grouping of interpreted constructs that supports a specific function for the usage of product data across multiple application contexts. See 3.4.14 for definition of interpretation

[SOURCE: ISO/DIS 10303-1:2020, 3.1.9]

3.4.3
application module
AM

reusable collection of a scope statement, information requirements, mappings and module interpreted model that supports a specific usage of product data across multiple application contexts

[SOURCE: ISO/DIS 10303-1:2020, 3.1.11]

3.4.4
application protocol
AP

part of ISO 10303 that specifies an application interpreted model satisfying the scope and information requirements for a specific application

Note 1 to entry: This definition differs from the definition used in ISO 7498-2:1989 Information processing systems — Open Systems Interconnection (ISO 7498-2:1989, Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture, 1989) standards because the protocols address different contexts of use.

[SOURCE: ISO/DIS 10303-1:2020, 3.1.17]

3.4.5
application reference model
ARM

information model that describes the information requirements and constraints of an application within an application protocol or module

[SOURCE: ISO/DIS 10303-1:2020, 3.1.18]

3.4.6
application resource

integrated resource whose contents are related to a group of application contexts

[SOURCE: ISO/DIS 10303-1:2020, 3.1.19]

3.4.7
common resources

collection of information models, specified in the EXPRESS language, that can be reused to specify application-specific information models within the domain of industrial data

Note 1 to entry: The resource constructs defined by an application module are those defined in its module interpreted model schema.

Note 2 to entry: The term does not specify a specific series of ISO 10303 parts.

[SOURCE: ISO/DIS 10303-1:2020, 3.1.22]

3.4.8
conformance class

subset of an application protocol for which conformance can be claimed

[SOURCE: ISO/DIS 10303-1:2020, 3.1.25]

3.4.9
data

representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers

[SOURCE: ISO/DIS 10303-1:2020, 3.1.29]

3.4.10
generic resource

integrated resource whose contents are independent of a specific application

EXAMPLE Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation

[SOURCE: ISO/DIS 10303-1:2020, 3.1.38]

3.4.11
information

facts, concepts or instructions

[SOURCE: ISO/DIS 10303-1:2020, 3.1.41]

3.4.12
information model

conceptual model of product data

Note 1 to entry: In ISO 10303, an information model is based on the Object-relationship modelling technique that organizes the product data as represented in different system aspects.

Note 2 to entry: In ISO 10303 information models are developed using EXPRESS modelling language.

EXAMPLE Application Resource Model for ISO 10303-242 Managed, model-based engineering.

[SOURCE: ISO/DIS 10303-1:2020, 3.1.42]

3.4.13
integrated resource
IR

part of ISO 10303 that defines a group of resource constructs used as the basis for product data. It includes the two types of resource parts: generic resources and application resources

Note 1 to entry: The 4x to 6x numbering is reserved for integrated generic resources and the 1xx numbering is reserved for integrated application resources

EXAMPLE 1 ISO 10303-42: generic resource: Geometric and topological representation.

EXAMPLE 2 ISO 10303-104: Integrated application resource: Finite element analysis

[SOURCE: ISO/DIS 10303-1:2020, 3.1.43]

3.4.14
interpretation

process of adapting a resource construct to satisfy an application-specific requirement of an application protocol

Note 1 to entry: The interpretation process can involve the addition of restrictions on attributes, the addition of constraints, and the addition of assignments.

[SOURCE: ISO/DIS 10303-1:2020, 3.1.42]

3.4.15
mapping specification

element of an application protocol that shows how the interpretation of integrated resources is used to meet the information requirements of the application

3.4.16
module interpreted model
MIM

information model that uses the common resources necessary to satisfy the information requirements and constraints of an application reference model, within an application module

Note 1 to entry: The term common resources is not meant to imply that all such information models are required to be to be used in a module interpreted model regardless of domain or application.

EXAMPLE Three-dimensional geometry information models are common resources used in many MIMs. However, an application module describing colour will not use three-dimensional geometry information models as a resource.

[SOURCE: ISO/DIS 10303-1:2020, 3.1.45]

3.4.17
product

thing or substance produced by a natural or artificial process

[SOURCE: ISO/DIS 10303-1:2020, 3.1.49]

3.4.18
product data

representation of information about a product in a formal manner suitable for communication, interpretation, or processing by human beings or by computers

[SOURCE: ISO/DIS 10303-1:2020, 3.1.50]

3.4.19
resource construct

collection of EXPRESS language entities, types, functions, rules and references that together define a valid description of an aspect of product data

[SOURCE: ISO/DIS 10303-1:2020, 3.1.55]

3.4.20 STEPmod

collection of reusable XML building blocks for developing standards from information models defined in EXPRESS

Note 1 to entry: An integral part of the repository's XML vocabulary is its representation of EXPRESS language constructs, i.e. its EXPRESS model. This specification shall refer to this EXPRESS model portion of the repository's

XML vocabulary as STEPmod EXPRESS. The STEPmod Specification Reference is available at http://stepmod.sourceforge.net/express_model_spec/

3.4.21
STEPlib

data and tools needed for the development and publication of the components of the STEP Extended architecture

Note 1 to entry:

STEPlib allows users to:

— manage the SysML models during the development of the models;

— prepare content to be fed into STEPmod: AntPub;

— implement the SysML to XSD binding: AntImp;

— transform the ARM EXPRESS model into the ARM-in-SysML model in order build the SysML mappings from SysML models to EXPRESS models: Reeper;

— check the quality of the SysML source content: CheckXMI.

STEPlib does not deprecate STEPmod. The EXPRESS-based modular STEP development is still managed by STEPmod and the WG12 Bugzilla server. This document assumes that STEPmod is a stable development and publication environment.

STEPlib has the sole purpose of managing the development of the models and producing the associated documentation subsets which will be assembled in STEPmod for ISO publication.

STEPlib is also the name of the top-level model of the SysML model tree used for all the STEP APs.

3.5 Abbreviated terms

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

AIC application interpreted construct (see 3.4.2)

AM application module (see 3.4.3)

AP application protocol (see 3.4.4)

ARM application reference model (see 3.4.5)

CC conformance class (see 3.4.8)

CD committee draft

DIS draft international standard

(E) English

FDIS final draft international standard

IR integrated resource (see 3.4.13)

IT information technology

MIM module interpreted model (see 3.4.16)

NP new work item proposal

PDF Portable Document Format

PAS Publicly Available Specification

PWI preliminary work item

SC 4 ISO/TC 184 Subcommittee 4

TC 184 ISO Technical Committee 184

TS Technical Specification

TR Technical Report

URL uniform resource locator

WD working draft

WG working group

4 Characteristics of an SC 4 standard

A standard communicates a technical specification to identified target users.

ISO requires such specifications to be unambiguous to the extent that any pair of those users can create mutually consistent implementations even when those users

— have not taken part in developing the standard, and

— are not working together.

In the case of many SC 4 standards, those implementations will be software applications that conform to the stated requirements of the standard.

Producing a standard involves developing drafts that satisfy requirements of technical accuracy and that communicate all necessary information to intended users.

Project teams are responsible for developing a standard that matches the scope of a balloted new work item proposal. The team produces drafts of the standard and revises them based on industry reviews, ballot results, and ballot comments. Figure 1 summarizes team activities to produce a standard. Several levels of reviews happen before the document goes for balloting:

— internal project reviews;

— reviews by industry sponsors and participants;

— "wide industry reviews" (previously referred to as Committee Draft for Comment);

— peer review by other experts in the same Working Group.

Figure 1 — An overview of creating a standard

Developers of an SC 4 standard should note that their work is driven by the needs of two different target audiences: end-users in industry who formulate and validate the problems and requirements that SC 4 standards address; and information technology providers who implement the standards and deliver solutions to end-users (see Figure 2).

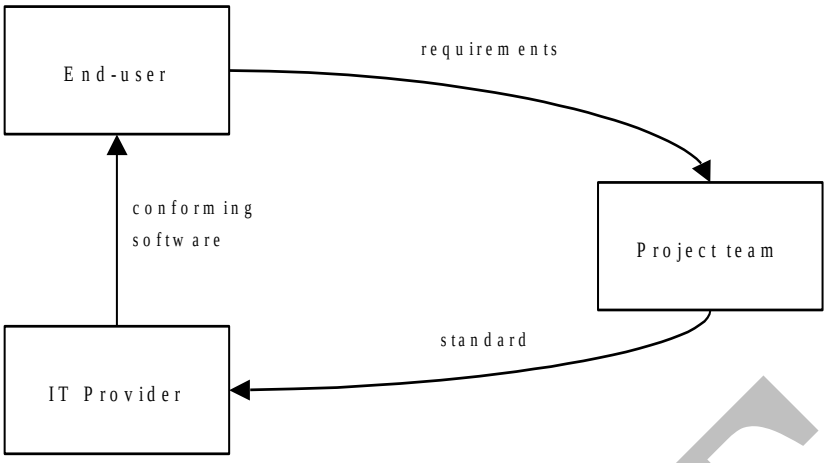

Figure 2 — Interaction between end-users, standards developers, and IT providers

The following points summarize key factors that project teams should take into account when preparing a draft standard.

— Treat writing the draft as a process that involves a series of recursive steps (preparing and planning, researching, organizing, drafting, and revising).

— Treat the user profile as the driving force behind all writing decisions.

— Understand what the user needs to know.

— Help the user understand the material through definitions, visuals, writing style, reduction of ambiguities and informative content such as notes and examples.

— Know what the users will do or what the team wants the users to do with the material.

— Ask that there be as many reviews for clarity of communication as there are for technical content.

5 Requirements for the structure and content of parts of SC 4 standards

5.1 General

For all the standards of the ISO/TC 184/SC 4 subcommittee the following documents provide applicable requirements:

— the ISO/IEC Directives, Part 2, 2018, *Principles and rules for the structure and drafting of ISO and IEC documents*;

— the ISO Simple template.

In addition the exceptions listed in 5.4 apply.

For the ISO 8000 series special requirements are given in Clause 7

For the ISO 10303 series special requirements are given in Clause 8.

For the ISO 13584 series special requirements are given in Clause 9.

For the ISO 15926 series special requirements are given in Clause 10.

For the ISO 18876 series special requirements are given in Clause 11.

To help you, drafting standards resources are available at https://www.iso.org/drafting-standards.html.

NOTE In these resources, the "Rice Model" is a useful example of how to structure the content of international standards but the detailed wording is not up to date. ISO requires editors always to use boilerplate wording from

the latest version of the ISO Simple template (https://isotc.iso.org/livelink/livelink/properties/17851245), which is periodically revised.

5.2 Checklists and approval procedures

This document specifies requirements that apply to the standards developed within SC 4. In order to confirm that each SC 4 standard submitted for ballot or publication satisfies the relevant requirements of this and other standing documents, SC 4 has approved a review and approval procedure. This procedure is specified in the SC 4 Quality Manual and includes requirements for the completion of the following checklists.

— The internal review checklist is used by a member of the project team (preferably not the project leader or the project editor) to check the document against the detailed requirements that apply to it. These requirements include those specified in the ISO/IEC Directives, Part 2, 2018, those specified in this document, and if any, those specified in any SC 4 standing document applicable to a series (e.g. ISO 10303).

— The project leader approval checklist is used by the project leader to check the document against a subset of the requirements that apply to it.

— The convener approval checklist is used by the convener of the working group to which the project is assigned to check the document against the main requirements that apply to it.

The "review and approval procedure" requires that the completed checklists are published (as working group N-numbered documents) and that both the project leader and the convenor have indicated their approval of the document before it can be submitted for ballot or publication.

Updated versions of the checklists are available from the following URL:
https://isotc.iso.org/livelink/livelink?func=ll&objId=11568437&objAction=browse&viewType=1

Any issues or questions regarding the checklists shall be reported to the SC 4 Quality Committee.

5.3 Required text

Many of the elements in an SC 4 standards require specific wording, called required text or boilerplate text.

The required text is managed in the version-controlled repository of ISO/TC 184/SC 4.

The repository is located under [GIT ISO URL]/projects/ISOTC184SC4/repos/boilerplate.

The required text is placed as fragments and is maintained to keep alignment with the ISO Directives (updated and validated using the change management process of ISO/TC 184/SC 4).

Each fragment of required text under version control is identified by the identifier specified in the header of the required text.

The light-yellow box below shows, with a "*lorem ipsum*" example, how we present the required text in this document.

[SC 4 required:SC4_xxxx]

lorem ipsum …

[end required]

Therefore, the required text shown in this document is included as an example, only for illustration.

Each working group is responsible for the management of the required text specific to the series of standards under their control.

The SC 4 Quality Committee is responsible for maintaining the repository.

5.4 SC 4 exceptions

5.4.1 Subdivision of the subject matter within an individual document

Table 1 lists the elements that comprise the content of parts of SC 4 standards. The bolded elements differ from the ISO/IEC Directives, Part 2, 2018; the introduction is a mandatory element for all the SC 4 standards and an index is provided as appropriate.

For non-HTML documents, the final cover page is generated by the ISO Central Secretariat.

Table 1 — Overview of the major subdivisions of a document and their arrangement in the text

Major subdivision	Mandatory/Optional/Conditional for all SC 4 standards
Title	Mandatory
Copyright	Mandatory
Foreword	Mandatory
Introduction	**Mandatory**
Scope	Mandatory
Normative references	Mandatory
Terms and definitions	Mandatory
Symbols and abbreviated terms	Conditional
Technical content	Mandatory/Optional/Conditional
Annexes	Optional
Bibliography	Conditional
Index	**Optional**

5.4.2 Language versions

SC 4 has agreed with ISO Central Secretariat that the parts of each SC 4 standard are prepared and published only in the English language.

NOTE Translation of SC 4 standards into other languages is not recommended. National bodies wishing to translate SC 4 standards into other languages for publication should base the translation on the corrected proof of the English language edition.

5.4.3 Index (optional)

An index, the final element of a standard, is an alphabetical list of the terms defined in the main text and the major topics discussed in the main text.

NOTE 1 The index is not an annex. It is an additional informative element that is the last element in the standard.

The index shall include an entry for each term that is defined in Clause 3 of the document. The entry cites each page where the definition or topic occurs. The index enables readers to find information quickly and easily.

The integrated resources and application protocol series of ISO 10303 have specific requirements for the index. The index shall not contain EXPRESS attribute names. Where there is a good case, other terms that readers of the document would need to find in the index, may be included.

The key to compiling a good index is selectivity. Instead of listing every possible reference to a term or a topic, select references to the formal definition of the term or to the clause, subclause, or paragraph where the topic is discussed fully or where a significant point is made about it. For index entries, chose

words or phrases that best represent a topic. Key terms are those that a reader would most likely look for in an index.

NOTE 2 See ISO 999 (ISO 999:1996) and The Handbook of Technical Writing (ALRED, BRUSAW, & OLIU, 2012) for further information about creating an index.

If the index is present then the terms defined in Clause 3 (Terms and definitions) of the standard shall be listed in the index.

In the "Table of contents", the index shall appear after the bibliography.

5.4.4 Use of ASN.1 Identifiers in SC 4 standards (optional)

5.4.4.1 Information object registration annex

Some parts of SC 4 standards can have an "Information object registration annex". This annex defines the information object identifier for the part as specified by ISO/IEC 8824-1 (ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation, 2015). As a consequence, the SC 4 standard shall specify a reference to ISO/IEC 8824-1.

The status of the annex, in which the information object registration is specified, varies according to its referencing within the text.

The structure of the annex, in which the information object registration is specified, varies according to the nature of the part.

NOTE In ISO 10303 this annex is the last normative annex.

In a part of an SC 4 standard that includes one or more schemas, the annex has two subclauses: Document identification and Schema identification. In a part that does not include schemas, there is no subdivision; the content of the annex corresponds to that of the Document identification subclause in the first case.

The complete mechanism of the information object registration is described in ISO/DIS 10303-1:2020 Clause 7 Information object registration scheme.

5.4.4.2 Document identification

An example of the text to introduce the document identification for standards prepared by SC 4 reads as follows, see 5.3. for implementing the required text.

[SC 4 required:SC4_annex_obj_reg-d]

To provide for unambiguous identification of an information object in an open system, the object identifier

{iso standard 8000 part(065) version(1)}

is assigned to this document. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

[end required]

5.4.4.3 Schema identification

For the parts of SC 4 standards that define schemas or include an electronic insert, include a subclause titled "Schema identification" within the information object registration annex. If the part includes more than one schema or electronic insert then further subdivide this subclause such that each subdivision identifies one schema or one electronic insert. Order these subclauses in the same sequence as the schemas themselves. If the part includes one schema then use the text given below within the "Schema identification" subclause.

For each schema or electronic insert, an example of text is provided below:

[SC 4 required:SC4_annex_obj_reg-s]

To provide for unambiguous identification of the schema-name in an open information system, the object identifier

{iso standard 10303 part(42) version(10) object(1) topology-schema(2)}

is assigned to the topology-schema schema. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

[end required]

A tutorial on the ASN.1 identifier is provided in Annex A.

5.5 Other topics of importance to SC 4 standards

5.5.1 ISO deliverables numbering

ISO/IEC Directives, Part 1, Consolidated ISO Supplement, 2019 define the basic procedures to be followed in the development of International Standards (see Clause 2) and other publications (see Clause 3).

NOTE 1 To help you to choose the better document, when evaluating new work item proposals, ISO CS has prepared a short video, available at https://youtu.be/T--RIs3E7ZE, describing the different ISO deliverables, their uses and how to develop them.

The Table 2 below gives examples of numbering of documents at different stages. If a change occurs in the ISO/IEC Directives, Part 1the ISO/IEC Directives, Part 1, apply.

Table 2 — Example of numbering of documents

Document Status / Stage	Internal Standard IS (Normative)	Technical Specification TS (Normative)	Publicly Available Specification PAS (Normative)	Technical Report TR (Informative)
Working Draft WD (20)	**ISO/WD 10303-210**	**ISO/WD TS 10303-4000**	**ISO/WD PAS 1996-3**	
Committee Draft CD (30)	**ISO/CD 10303-210**	**ISO/CD TS 8000-65**		**ISO/CD TR 10645**
Draft International Standard DIS (40)	**ISO/DIS 10303-1:2020(E)**			
Final Draft International Standard FDIS (50)	**ISO/FDIS 10303-210:2010(E)**			
Publication IS, TS, PAS, TR (60)	**ISO 10303-242:2020(E)**	**ISO/TS 8000-60:2017(E)**	**ISO/PAS 24019:2002(E)**	**ISO/TR 10303-307:2000(E)**

NOTE 2 The complete matrix presentation of project stages is available in the ISO/IEC Directives, Part 1, Consolidated ISO Supplement, 2019, Annex SD

NOTE 3 For more information on the numbering of documents, please refer to ISO/IEC Directives, Part 1, Consolidated ISO Supplement, 2019, Annex SE

NOTE 4 The complete matrix presentation of project simplified diagram of options is available in the ISO/IEC Directives, Part 1, Consolidated ISO Supplement, 2019, Annex F

Editors shall always check publication dates with the SC 4 Secretariat. Documents submitted to ISO CS after 15 September may have to be dated for the following year.

Each document relating to the work of an ISO technical committee or subcommittee circulated to all or some of the member bodies shall have a “N-number”. The N-number is made up of the following two parts separated by the letter N:

— the number of the technical committee (TC) and, when applicable, the number of the subcommittee (SC) to which the working document belongs;

— an overall serial number.

EXAMPLE The reference number of this document will look as follows: **ISO/TC 184/SC 4 N3485**

5.5.2 Table of content

The following rules apply for the table of content:

— terminological entries are not included in the table of contents;

— subdivisions of annexes are not included in the table of contents;

— for the non-HTML document, tables of figures, tables, equations, formulae, parts are not included in the table of contents.

5.5.3 Patent rights

A published document for which patent rights have been identified during the preparation thereof, shall include the following notice in the introduction (see ISO/IEC Directives, Part 2, 2018, Clause 30).

The generic statement in the Foreword needs to be retained whether or not there is an additional statement about identified patents in the Introduction.

5.5.4 Scope

The scope clause shall begin immediately after the title element.

The scope clause should be less than one page long.

NOTE Scope is covered by the ISO/IEC Directives, Part 2, 2018, Clause 14

For all the SC 4 normative standards the scope clause shall start with a specific wording. Examples are shown as follows, see 5.3. for implementing the required text.

[SC 4 required:SC4_Scope_scope-1]

This document specifies …

[end required]

Then, the following wording to introduce any further information that is within the scope of the part but does not fit into the first list is provided below:

[SC 4 required:SC4_Scope_scope-2]

The following is (are) within the scope of this document:

<list of in-scope elements>

[end required]

And finally, the following conditional wording to introduce information that is outside the scope of the part:

[SC 4 required:SC4_Scope_scope-3]

The following is (are) outside the scope of this document:

<list of out-scope elements>

[end required]

A complete example is shown below.

This document specifies the requirements for the quality identifiers that form part of an exchange of master data. These requirements supplement those of ISO 8000-110.

The following are within the scope of this document:

— the syntax and semantics of quality identifiers to allow the unambiguous identification of the owner of the identifier and any restrictions on the use of the identifier;

— the principles of resolving quality identifiers to the data set they represent;

— the characteristics that define quality identifiers.

The following are outside the scope of this document:

— the methods used for the creation of identifiers;

— the syntax of the query and of the response used in the resolution of identifiers;

— the methods used for the resolution of identifiers.

5.5.5 Terms and definitions clause

5.5.5.1 General

Definitions of terms are required to conform to the provisions of the ISO/IEC Directives, Part 2, 2018, Clause 16 and of ISO 10241-1:2011 (ISO 10241-1:2011, Terminological entries in standards — Part 1: General requirements and examples of presentation). Additional recommended practices from *A Concise Introduction to Logic* by Patrick J. Hurley (HURLEY, 1999) are given in 5.5.5.2.

5.5.5.2 Criteria for lexical definitions

"*Because the function of a lexical definition is to report the way a word is actually used in a language, lexical definitions are the ones we most frequently encounter and are what most people mean when they speak of the 'definition' of a word. Accordingly, it is appropriate that we have a set of rules that we may use in constructing lexical definitions of our own and in evaluating the lexical definitions of others."*

Rule l: A lexical definition should conform to the standards of proper grammar.

Rule 2: A lexical definition should convey the essential meaning of the word being defined.

The aspects mentioned in the definition should be the important or necessary features of the thing defined, not trivial ones.

Rule 3: A lexical definition should be neither too broad nor too narrow.

A definition is too broad if the definition applies to things other than the things that are being defined.

EXAMPLE For the term "data governance", the definition:

— "development and enforcement of policies related to the management of data" is correct;

— "development of policies related to the management of data" is too broad, because this incorrectly includes situations where an organization is not also performing enforcement;

— "development and enforcement of policies related to the management of digital data" is too narrow because organizations also have a responsibility to handle hard-copy documents appropriately.

Rule 4: A lexical definition must not be circular.

A circular definition uses the definiendum in some way in the definition and is thus not genuinely informative.

EXAMPLE In the definitions of "data administrator" and "data technician" below the definition is circular because the two terms use each other in their definitions:

data administrator
person who controls and coordinates the work of ***data technicians*** by defining criteria needed to maintain data quality, designing data schema, and analysing the causes of errors to prevent their recurrence

Note 1 to entry: By providing supporting resources and guidelines to data technicians, the data administrator puts the data quality plan into practice.

data technician
person who creates, reads, modifies, and deletes data in accordance with the guidelines for data quality management set by the ***data administrator***, and measures data quality and corrects data errors found as a result of data quality measurement

The definition of "data technician" has been modified to:

data technician

person who creates, reads, modifies, and deletes data in accordance with the guidelines for data quality management and measures data quality and corrects data errors found as a result of measuring data quality.

Note 1 to entry: The data administrator sets the guidelines for data quality management.

Rule 5: A lexical definition should not be negative when it can be affirmative.

Rule 6: A lexical definition should not be expressed in figurative, obscure, vague, or ambiguous language.

— A definition is figurative if it involves metaphors or tends to paint a picture instead of exposing the essential meaning of a term.

  Example "Architecture means frozen music".

— A definition is obscure if its meaning is hidden. One source of obscurity is overly technical language.

— A definition is vague if its meaning is blurred.

— A definition is ambiguous if it lends itself to more than one distinct interpretation.

Rule 7: A lexical definition should avoid affective terminology.

Affective terminology is any kind of word usage that plays upon the emotions of the reader or listener.

Rule 8: A lexical definition should indicate the context to which the definition pertains.

This rule applies to any definition in which the context of the definition if important to the meaning of the definiendum. Whenever the definiendum is a word that means different things in different context is, a reference to the context is important.

5.5.5.3 Note to a terminological entry

A note to a terminological entry (referred to as "Note # to entry") follows different rules from a note ("NOTE #") integrated in the text. It provides additional information that supplements the terminological data, such as:

— provisions (statements, instructions, recommendations or requirements) relating to the use of a term,

— information regarding the units applicable to a quantity, or

— an explanation of the reasons for selecting an abbreviated form as the preferred term.

Notes to entry shall be numbered starting with "1" within each terminological entry. A single note to entry shall be numbered.

5.5.5.4 References in a terminological entry

The source references for individual entries are informative and are listed in the Bibliography.

EXAMPLE

data element

unit of data that is considered in context to be indivisible

[SOURCE: **ISO/IEC 11179** 1:2015, 3.3.8]

The source references cited in the introductory sentence of Clause 3 such as " For the purposes of this document, the terms and definitions given in ISO #### apply ", are considered to be normative and are listed in Clause 2.

EXAMPLE For the purposes of this document, the terms and definitions given in **ISO 8000-2** and the following apply.

5.5.6 Symbols and abbreviated terms

An example of the wording to use to introduce this element is provided below, see 5.3. for implementing the required text.

[SC 4 required: SC4_symbols]

For the purposes of this document, the following symbols apply:

or [SC 4 required: SC4_abbreviatedTerms]

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

or [SC 4 required: SC4_symbols_abbreviatedTerms]

For the purposes of this document, the following symbols and abbreviated terms apply:

[end required]

The symbols and abbreviations shall be given in two left-justified columns, using normal style, without punctuation (except as part of the abbreviated term or the explanation of the symbol). Capitalization shall correspond to the use of the abbreviation. The right column shall contain the phrase corresponding to the symbol or abbreviation. See 3.5 of this document for an example of an abbreviated terms subclause.

The following abbreviations may be used in the text without definition; do not list these abbreviations.

— ISO;

— IEC.

5.5.7 References to subdivisions of the text

In addition to the ISO/IED Directives, Part 2, 2018, 22.4, use "Clause" only to refer to an entire clause. Do not use the words "subclause" or "reference". Use the following forms:

— in accordance with Clause 3;

— according to 3.1;

— details as given in 3.1.1;

— (see 3.1.1);

— as described in 3.1.2;

— see Annex B.

5.5.8 Punctuation of words in a series

SC 4 recommends adding a comma in a series for clarity. Officially this serial comma (also called an Oxford comma or a Harvard comma) is considered optional, but using it consistently is easier for

international writers. This comma is placed immediately before the coordinating conjunction (usually and or or) in a series of three or more terms.

EXAMPLE The following sentence is an example of this type of punctuation. “In the United States, dogwood, cherry, and redbud are three types of trees that bloom in the spring.”

5.5.9 Avoid abbreviations

Do not use “i.e.” and “e.g.”. Instead, use the EXAMPLE format.

Likewise, do not use “etc.”. End the series prior to the “etc.” being certain to use a serial comma before the “and” (added if not already there). To state that the series is incomplete, use “such as” at the start of the series.

5.5.10 Landscape

Use of landscape mode for figures and tables in SC 4 standards is deprecated with the following exceptions:

— All large size diagrams.

5.5.11 Spelling, hyphenation, words to avoid and frequently used words

These topics, as they are frequently updated, will be managed within the checklists.

6 EXPRESS presentation style

6.1 General

This clause gives rules and guidelines specific to documenting EXPRESS. The subjects covered are as follows.

— Layout: explains how to use indentation, blank lines, and other white space to produce a consistent layout (see 6.2).

— Expression: provides guidance for how to layout the various expressions in EXPRESS. (see 6.3).

— Statements: provides guidance for how to layout the various statements in EXPRESS. (see 6.4).

— Style: deals with naming conventions that are to be used for all schemas developed for SC 4 standards (see 6.5).

— Usage: defines rules for the use of EXPRESS language elements in SC 4 standards (see 6.6).

NOTE See also the *Guidelines for application interpreted model development* (N532:1997), the *Procedures for application interpretation* (BARNARD FEENEY & PALMER (eds), 1998), and *Syntactic and semantic rules* (STEP (Standard for the Exchange of Product Model Data) Resource Integration: Semantic & Syntactic Rules., 1991) for further guidance on the use of EXPRESS in ISO 10303.

— Documentation requirements for definitions of EXPRESS schemas (see 6.7).

— The layout and format to be used for EXPRESS-G diagrams (see 6.8).

— Examples are provided in 6.9.

The following convention is used to display EXPRESS examples:

```
EXAMPLE;
  first  : INTEGER;
  second : STRING;
END_EXAMPLE;
```

6.2 Layout rules

The layout rules explain how to use white space to produce a schema that has a uniform appearance. A uniform appearance helps the reader to find and use the material of interest.

To prevent unusual spacing, use left justification for EXPRESS language statements, not full justification.

Because of the variability of the size of EXPRESS declarations, no hard rule can be given for the number of lines to force on one page. Do not insert extra page breaks.

6.2.1 Organization of the schema elements

Schema elements shall be written in the order specified in Table 3. None of these elements are mandatory; only include these elements as needed (see also 6.7).

Table 3 — Order of declarations of schema elements

Declaration type	EXPRESS keyword
Interface statements	USE FROM or REFERENCE FROM
Constants	CONSTANT
Types	TYPE
Entity data types	ENTITY
Subtype constraints	SUBTYPE_CONSTRAINT
Functions	FUNCTION
Rule	RULE
Procedure	PROCEDURE

6.2.2 Use of the colon

When used in an attribute declaration, at least one space shall be placed before the colon that follows the attribute identifier. One space shall be placed between the colon and the identifier that specifies the domain of the attribute. When there is more than one attribute declaration within an entity data type declaration, additional space can be used between the attribute identifier and the colon to align the colons in tabular fashion, when possible.

EXAMPLE 1 The following example illustrates alignment of the colons in attribute declarations:

```
ENTITY e1;
  first  : INTEGER;
  second : STRING;
END_ENTITY;
```

When used in a label, no space shall separate the label name from the colon.

EXAMPLE 2 The following example illustrates the placement of colons in labels:

```
ENTITY e2;
  first  : INTEGER;
  second : STRING;
UNIQUE
  UR1: second;
WHERE
  WR1: first > 5;
END_ENTITY;
```

6.2.3 Use of the semicolon

A space shall not appear before a semicolon, and a semicolon shall not appear at the beginning of a line.

EXAMPLE 1 The following example illustrates correct placement of semicolons:

```
ENTITY e1;
  first  : INTEGER;
  second : STRING;
END_ENTITY;
```

EXAMPLE 2 The following example illustrates incorrect placement of semicolons:

```
ENTITY e1 ;
  first  : INTEGER
;
  second : STRING
;END_ENTITY      ;
```

6.2.4 Commenting conventions

It should be possible to extract an EXPRESS data specification (one or more schema declarations) defined in an SC 4 standard from the source description of the part of ISO 10303 by regarding all material that is not part of the EXPRESS declarations as comments. Accordingly, EXPRESS declarations shall be separated from the surrounding text by EXPRESS embedded remark markers ("`(*`" and "`*)`"). The closing marker shall appear by itself on the line immediately preceding the EXPRESS object description, and the opening marker shall appear by itself on the line immediately following the description of the EXPRESS object.

EXAMPLE 1 The following example illustrates the use of embedded remark markers:

```
*)
ENTITY e1;
  first  : INTEGER;
  second : STRING;
END_ENTITY;
(*
```

NOTE This implies that to process the complete document as EXPRESS, it may be necessary to add one opening marker at the beginning of the document and one closing marker at the end of the document.

Material not intended to describe a requirement, such as examples or notes that include EXPRESS code, shall not use the "`*)`" and "`(*`" delimiters.

Tail remarks (`-- text`) shall be used to indicate the source of a schema that is specified in an interface statement.

EXAMPLE 2 The following example illustrates the use of tail remarks in interface statements:

```
USE FROM schema1 (e1, e2); -- ISO sssss-ppp
```

Tail remarks may also be used to annotate portions of the code of an EXPRESS PROCEDURE or FUNCTION. The functionality of algorithms may be explained by this method.

EXAMPLE 3 The following example illustrates the use of remark tags:

```
*)
ENTITY ent;
  attr   : INTEGER;
END_ENTITY;
(*"ent.attr" The attr attribute……*)
```

The tagged remark in the example above refers to the attribute `attr` in the scope of `ent`.

6.2.5 Indentation

Use indentation to achieve a consistent layout for EXPRESS declarations. In the requirements that follow, minimum requirements for indentation are stated; the same indentation should be used within one SC 4 standard.

6.2.6 Layout of a schema

The SCHEMA and END SCHEMA keywords shall be flush with the left margin. Contained objects shall also have the begin and end keywords flush with the left margin. The SCHEMA keyword shall be on a line immediately after the File header content.

Interface clauses begin flush with the left margin.

EXAMPLE The following example illustrates the layout of schema, type, and entity data type declarations:

```
SCHEMA schema_name;
USE FROM schema2;
TYPE type1;
END_TYPE;
ENTITY entity1;
END_ENTITY;
END_SCHEMA;
```

6.2.7 Layout of interface statements

An interface begins with the **USE FROM** or **REFERENCE FROM** keywords at the left margin. Followed by the name of the SCHEMA being interfaced.

Optionally, there can be a list of object names that are interfaced from the specified schema. If this list is present, it will begin on the following line with the open parenthesis positioned two characters to the right of the left margin. Each item in the list is written to a line by itself with the first one being written just after the open parenthesis. The items are separated by a comma. Each subsequent item will be aligned with the one above it. The final item will have the close parenthesis following immediately after it.

Finally, the interface specification is terminated by a semicolon.

The source document for the interface shall be provided as a trailing comment immediately after the name of the interfaced schema. If the version of the schema is available in the source document, it shall be provided in the output.

EXAMPLE The following examples illustrates the layout of **USE FROM** interface statements:

```
SCHEMA schema3 'iso standard 10303 part(nn) version(yy)
object(1)geometric_model_schema(3)';
USE FROM schema1  -- ISO 10303-42 schema1 version 10
  (entity1,
   entity2,
   entity3);
USE FROM schema2  -- ISO 10303-41 schema2 version 16
  (entity1,
   entity2,
   entity3,
   entity4);
USE FROM schema3; -- ISO 10303-43 schema3 version 5
REFERENCE FROM schema4; -- ISO 10303-43 schema4 version 7
  (function1,
   function2);
...
END_SCHEMA; -- schema3
```

6.2.8 Layout of a constant declaration

The **CONSTANT** keyword is on a line by itself and flush with the left margin of the enclosing declaration.

On each following line is a constant specification. Indented two characters to the right of the start of the **CONSTANT** keyword is the name of the constant followed by a colon :) followed by the declared type of the constant followed by an assignment operator (:=) followed by an expression which calculates the value for the constant followed by a semicolon (;) to terminate the constant definition.

All of the colons separating the constant name from the constant type can be aligned. There should be at least one space before and after this colon.

All of the colons that are part of the assignment operator which separates the constant type from the initialization expression should be aligned. There should be at least one space before and after the assignment operator.

EXAMPLE The following example illustrates the layout of constraint declarations:

```
CONSTANT
  schema_prefix               : STRING         := 'MATH_FUNCTIONS_SCHEMA.';
  the_integers                : e_space        := mk_e_space(es_integers);
  the_reals                   : e_space        := mk_e_space(es_reals);
  the_numbers                 : e_space        := mk_e_space(es_numbers);
  the_logicals                : e_space        := mk_e_space(es_logicals);
  the_booleans                : e_space        := mk_e_space(es_booleans);
  the_strings                 : e_space        := mk_e_space(es_strings);
  the_binarys                 : e_space        := mk_e_space(es_binarys);
  the_maths_spaces            : e_space        := mk_e_space(es_maths_spaces);
  the_generics                : e_space        := mk_e_space(es_generics);
  the_empty_space             : finite_space   := mk_finite_space([]);
  the_nonnegative_reals       : real_interval_from_min
:= make_real_interval_from_min(0.0, closed);
  the_zero_one_interval       : finite_real_interval
:= make_finite_real_interval(0.0, closed, 1.0, closed);
  the_zero_pi_interval        : finite_real_interval
:= make_finite_real_interval(0.0, closed, pi, closed);
  the_neg1_one_interval       : finite_real_interval
:= make_finite_real_interval(-1.0, closed, 1.0, closed);
END_CONSTANT;
```

6.2.9 TYPE layout

The TYPE and END TYPE keywords are flush with the left margin and each on its own line.

Immediately after the **TYPE** keyword is the name of the type, an equal sign ('=') and the underlying type. If the underlying type is either an ENUMERATION or a SELECT and all of the values will not fit on one line, then the list begins on the next line indented two characters and each value will be written on its own line.

If present, the **WHERE** keyword is flush with the left margin. The individual where rules are indented two characters to the right.

EXAMPLE The following are examples of the format and layout of type declarations:

```
TYPE type1 = STRING;
WHERE
  WR1: <expression>;
END_TYPE; -- type1
TYPE enum_type1 = ENUMERATION OF (on, off, whatever);
END_TYPE; -- enum_type1

TYPE enum_type2 = ENUMERATION OF
  (val1,
   val2,
   val3,
   val4);
END_TYPE; -- enum_type2
TYPE sel_type1 = SELECT (ent1, ent2, ent3);
END_TYPE; -- sel_type1
TYPE sel_type2 = SELECT
  (entity1,
```

entity2,

```
   entity3,
   type4,
   entity5);
END_TYPE; -- sel_type2
```

6.2.10 Entity data type declaration layout

The **ENTITY** and **END ENTITY** keywords are flush with the left margin and each on its own line.

Immediately after the **ENTITY** keyword is the name of the entity.

If there is a **SUPERTYPE OF** specification it is specified next beginning on the next line, indented two characters to the right of the left margin.

If there is a **SUBTYPE OF** specification, it is specified next beginning on the next line, indented two characters to the right of the left margin.

Next is a semicolon to mark the end of the entity header.

The attributes are next with each one on a line by itself, indented two characters to the right of the left margin. The colons that separate the attribute name from the attribute type should be aligned with at least one space before and after the colon.

If there are any derived attributes, they are written next. The **DERIVE** keyword is written on a line by itself flush with the left margin. Each attribute is written on a line by itself indented two characters to the right of the left margin. The colons that separate the attribute name from the attribute type should be aligned and have at least one space before and after. The colons in the assignment operator should be aligned with at least one space before and after the assignment operator. Each attribute should end with a semicolon.

If there are any inverse attributes, they are written next. The **INVERSE** keyword is written on a line by itself, flush with the left margin. Each attribute is written on a line by itself, indented two characters to the right of the left margin.

An inverse attribute has a name followed by a colon and the type of the attribute. The attribute type has an optional **SET** or **BAG** aggregate specification which consists of either the **SET** or **BAG** keyword followed by an optional bounds specification, followed by the OF keyword and then the name of an **ENTITY**, followed by the **FOR** keyword and the specification of the attribute being inverted. The colons between the attribute name and type should be aligned and have at least one space before and after.[4]

If there are any unique rules, they are written next. The **UNIQUE** keyword is written on a line by itself, flush with the left margin. Each unique rule is written on a line by itself, indented two characters to the right of the left margin.

If there are any local rules, they are written next. The **WHERE** keyword is written on a line by itself, flush with the left margin. Each local rule is written on a line by itself, indented two characters to the right of the left margin.

EXAMPLE The following example illustrates the layout and format of an entity data type declaration:

```
ENTITY entity1
  SUPERTYPE OF (entity2 ANDOR entity3)
  SUBTYPE OF (entity5);
  attr1 : type1;
  attr2 : type2;
DERIVE
  der1 : type1 := <expression>;
INVERSE
  inv1 : entity1 FOR attr3;
  inv2 : BAG OF entity2 FOR attr4;
  inv3 : BAG [1:4] OF entity3 FOR entity2.attr3;
UNIQUE
  UR1: attr1;
  UR2: attr1, attr2;
WHERE
  WR1: <expression>;
  WR2: <expression>;
END_ENTITY;
```

6.2.11 Algorithm layout

6.2.11.1 FUNCTION

The **FUNCTION** and **END FUNCTION** keywords are on separate lines and aligned with the left margin. Immediately, following the **FUNCTION** keyword is a space followed by the name of the function.

Immediately after the function name, without a space, is the open parenthesis that marks the beginning of the formal parameters. Each parameter is written on a line by itself with the first one being written on the same line as the open parenthesis with no space between it and the parenthesis.

A semicolon separates each of the formal parameters and is flush against the parameter that it comes after.

After the final formal parameter, is a close parenthesis followed by a colon and then the type for the value that is returned from the function. The colon should have at least one space before and after it. After the return type is a semicolon which marks the end of the function header.

If there are any local **ENTITY**, **SUBTYPE CONSTRAINT**, **FUNCTION**, **RULE**, **PROCEDURE**, or TYPE declarations, they are written next, indented two characters to the right of the left margin.

If there are any **CONSTANT** declarations, they are written next. The **CONSTANT** and **END CONSTANT** keywords are written on separate lines, flush with the left margin. Each constant is written on a separate line indented two characters to the right of the left margin.

If there are any **LOCAL** declarations, they are written next. The **LOCAL** and **END LOCAL** keywords are written on separate lines, flush with the left margin. Each local is written on a separate line indented two characters to the right of the left margin.

Finally, we write any statements that are included in the **FUNCTION**, indented two characters to the right of the left margin.

EXAMPLE The following is an example of function declaration:

```
FUNCTION fun_name (a : INTEGER;
                   b : STRING;
                   c : REAL) : STRING;
  ENTITY fun_sub_ent;
    name : STRING;
  END_ENTITY;
  CONSTANT
  ...
  END_CONSTANT;
  LOCAL
    inst : fun_sub_ent;
  END_LOCAL;
  inst := fun_sub_ent(b);
END_FUNCTION;
```

6.2.11.2 RULE

It starts with the **RULE** keyword written on a new line flush with the left margin. Immediately following the **RULE** keyword is a space followed by the name of the rule, followed by the **FOR** keyword.

Immediately after the **FOR** keyword is an open parenthesis that marks the beginning of the list of entities that this rule applies to. Each entity name is placed on a line by itself with the first one being written on the same line as the open parenthesis with no space between it and the parenthesis.

A comma separates each entity name and is positioned flush against the entity name it follows.

After the final entity name is a close parenthesis followed by a semicolon.

If there are any local **ENTITY**, **SUBTYPE CONSTRAINT**, **FUNCTION**, **RULE**, **PROCEDURE**, or **TYPE** declarations, they are written next, indented two characters to the right of the left margin.

IF there are any **CONSTANT** declarations, they are written next. The **CONSTANT** and **END CONSTANT** keywords are written on separate lines, flush with the left margin. Each constant is written on a separate line indented two characters to the right of the left margin.

If there are any **LOCAL** declarations, they are written next. The **LOCAL** and **END LOCAL** keywords are written on separate lines, flush with the left margin. Each local is written on a separate line indented two characters to the right of the left margin.

The statements that make up the body of the **RULE** are written next, with each one on a separate line and indented two characters to the right of the left margin.

If there are any where rules, they come next. The **WHERE** keyword is written on a separate line, flush with the left margin. Each where is written on its own line, indented two characters to the right of the left margin.

Finally comes the **END RULE** keyword on its own line, flush with the left margin, followed by a semicolon.

EXAMPLE The following is an example of rule declaration:

```
RULE rule_id FOR '(' entity_ref  ',' entity_ref  ')' ';'
  ENTITY rule_sub_ent;
    name : STRING;
  END_ENTITY;
  CONSTANT
    my_pi : REAL := 3.1415926;
  END_CONSTANT;
  LOCAL
    my_ent: rule_sub_ent;
  END_LOCAL;
  inst := rule_sub_ent('what');
WHERE
  WR1: expression;
  ...
END_RULE;
```

6.2.11.3 PROCEDURE

It starts with the **PROCEDURE** keyword written on a new line flush with the left margin. Immediately following the **PROCEDURE** keyword is a space followed by the name of the procedure, followed by no space and an open parenthesis that marks the beginning of the formal parameters.

Each formal parameter is written on its own line, aligned with the one above it. The first formal parameter is written immediately after the open parenthesis with no space between it and the parenthesis.

EXAMPLE The following is an example of procedure declaration:

```
PROCEDURE proc_id (formal_parameter; formal_parameter);
  declarations
  CONSTANT
    my_pi : REAL := 3.1415926;
  END_CONSTANT;
  LOCAL
    my_inst : proc_sub_ent;
  END_LOCAL;
  my_inst := proc_sub_ent(23.0)
END_PROCEDURE;
```

6.2.11.4 Formal parameters

Formal parameters shall be grouped by type; parameters shall be separated by a command followed by a space. Semicolons shall be treated according to 6.2.3 above; colons are treated according to 6.2.2 above except for the case of type labels where no space shall appear before or after the colon. There shall be no space between a function or procedure name and the following open parenthesis.

EXAMPLE 1 The following example illustrates the layout of formal parameters:

```
FUNCTION func_name(a, b, c : INTEGER;
                   d, e, f : REAL;
                   x, y, z : AGGREGATE OF point) : REAL;
```



```
PROCEDURE proc_name(a, b, c : INTEGER;
                    d, e, f : REAL;
                    VAR x, y, z : AGGREGATE OF point);
```

EXAMPLE 2 The following example illustrates a function with type labels:

```
FUNCTION add(a, b : GENERIC:label) : GENERIC:label;
```

The parameters of a rule list the entities to which the rule applies.

EXAMPLE 3 The following example illustrates the layout of the parameters of a rule.

```
RULE rule_name FOR (entity1, entity2, entity3);
```

If this does not fit on one line, follow the layout convention for type declarations.

6.2.11.5 Local variables

Local (variable) declarations shall be indented at least two spaces. Unlike attribute declarations, several local declarations that are of the same type may be declared on the same line.

EXAMPLE The following example illustrates the declaration of a number of local variables, some of the same type:

```
  LOCAL
    i, j, k : INTEGER;
    p       : REAL
  END_LOCAL;
```

6.2.11.6 Code body

A related group of statements shall be separated by one blank line from other groups of statements. A tail remark shall precede the statement group to explain its purpose. Structured statements shall be indented at least two spaces.

EXAMPLE The following example illustrates the layout of a code body:

```
  IF cond THEN
    statement;
  ELSE
    statement;
  END_IF;

  REPEAT ...;
    statement;
    ...
  END_REPEAT;

  -- Choose the appropriate case

  CASE ...;
    label : statement;
    label :
    BEGIN
      statement;
      statement;
      ...
    END;
  END_CASE;

  BEGIN;
    statement;
```

```
   ...
END;
```

6.3 Expressions

If the expression fits on one line, then it is done that way.

If it does not fit on one line, then it will be written to multiple lines with each line aligned with the expression's start point on the first line and written with as many of the elements as possible on each line.

6.3.1 Unary MINUS Expression

Write a hyphen ('-') followed by either a parenthetical expression, a literal value, or a qualifiable factor. There is no space between the hyphen and the following item.

```
-42
-(42 * 5 / 3)
```

6.3.2 NOT Expression

Write the NOT keyword, followed by either a parenthetical expression, a literal value, or a qualifiable factor. There is no space between the hyphen and the following item.

```
NOT(x + 5)
NOT y
```

6.3.3 Parenthetical Expression

Write a left parenthesis, followed by the expression, followed by a right parenthesis. If the expression spans multiple lines, then each subsequent line is aligned so that its first non-white space character is in the column immediately following the left parenthesis that began the Parenthetical Expression.

```
 (5+3)
(42 DIV 5 * 64 *
 23 + 15)
```

6.3.4 Unary Plus Expression

Write a plus sign ('+') followed by either a parenthetical expression, a literal value, or a qualifiable factor. There is no space between the plus sign and the following item.

```
+42
+(12 / 3)
```

6.3.5 Equal Expression

Write an expression followed by an equal sign ('=') followed by an expression. There will be at least one space before and after the equal sign.

```
x = y + 15
```

6.3.6 Exponent Expression

Write an expression followed by a double asterisk ('**') followed by an expression. There will be no space before or after the double asterisk symbol.

```
x**2
```

6.3.7 Greater Than Expression

Write an expression followed by a greater than symbol ('>') followed by an expression. There will be at least one space before and after the greater than symbol.

x>y

6.3.8 Greater Than Equal Expression

Write an expression followed by a greater than or equal symbol ('>=') followed by an expression. There will be at least one space before and after the greater than or equal symbol.

x >= y

6.3.9 IN Expression

Write an expression followed by the IN keyword followed by an expression. There will be at least one space before and after the IN keyword.

'this' IN ['some', 'that', 'this', 'other']

6.3.10 Instance Equal Expression

Write an expression followed by the instance equal symbol (':=:') followed by an expression. There will be at least one space before and after the instance equal symbol.

x :=: y

6.3.11 Less Than Expression

Write an expression followed by the less than symbol ('<') followed by an expression. There will be at least one space between the less than symbol.

x<y

6.3.12 Less Than or Equal Expression

Write an expression followed by the less than or equal symbol ('<=') followed by an expres- sion. There will be at least one space before and after the less than or equal symbol.

x <= y

6.3.13 Like Expression

Write an expression followed by the LIKE keyword followed by an expression. There will be at least one space before and after the LIKE keyword.

x LIKE y

6.3.14 Not Equal Expression

Write an expression followed by the not equal symbol ('<>') followed by an expression. There will be at least one space before and after the not equal symbol.

x <> y

6.3.15 Not Instance Equal Expression

Write an expression followed by the not instance equal symbol (':<>:') followed by an expression. There will be at least one space before and after the not instance equal symbol.

x :<>: y

6.3.16 Multiply Expression

This expression object can have more than two elements.

Write each expression separated by the multiply symbol ('*'). There will be at least one space before and after the multiply symbol.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the multiply symbol and the subsequent lines aligned with the first expression element on the line above.

```
x*y*z
```

```
X*y*
z*q
```

6.3.17 Addition Expression

This expression object can have more than two elements.

Write each expression separated by the addition symbol ('+'). There will be at least one space before and after the addition symbol.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the addition symbol and the subsequent lines aligned with the first expression element on the line above.

```
x+y+z
```

```
x+y+
z+q
```

6.3.18 Subtraction Expression

This expression object can have more than two elements.

Write each expression separated by the subtraction symbol ('-'). There will be at least one space before and after the subtraction symbol.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the subtraction symbol and the subsequent lines aligned with the first expression element on the line above.

```
x-y-z
```

```
x-y-
z-q
```

6.3.19 Division Expression

This expression object can have more than two elements.

Write each expression separated by the division symbol ('/'). There will be at least one space before and after the division symbol.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the division symbol and the subsequent lines aligned with the first expression element on the line above.

```
x/y/z
```

```
x/y/
z/q
```

6.3.20 AND Expression

This expression object can have more than two elements.
Write each expression separated by the AND keyword. There will be at least one space before and after the AND keyword.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the AND keyword and the subsequent lines aligned with the first expression element on the line above.

x AND y AND z

x AND y AND

z AND q

6.3.21 Compose Expression

This expression object can have more than two elements.

Write each expression separated by the compose symbol ('||'). There will be at least one space before and after the compose symbol.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the compose symbol and the subsequent lines aligned with the first expression element on the line above.

x || y || z

x || y ||
z || q

6.3.22 DIV Expression

This expression object can have more than two elements.
Write each expression separated by the DIV keyword. There will be at least one space

before and after the DIV keyword.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the DIV keyword and the subsequent lines aligned with the first expression element on the line above.

x DIV y DIV z

x DIV y DIV

z DIV q

6.3.23 MOD Expression

This expression object can have more than two elements.
Write each expression separated by the MOD keyword. There will be at least one space

before and after the MOD keyword.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the MOD keyword and the subsequent lines aligned with the first expression element on the line above.

x MOD y MOD z

x MOD y MOD

z MOD q

6.3.24 OR Expression

This expression object can have more than two elements.
Write each expression separated by the OR keyword. There will be at least one space before

and after the OR keyword.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the OR keyword and the subsequent lines aligned with the first expression element on the line above.

```
x OR y OR z
```

```
x OR y OR
z OR q
```

6.3.25 XOR Expression

This expression object can have more than two elements.
Write each expression separated by the XOR keyword. There will be at least one space

before and after the XOR keyword.

If all of the elements can't fit on one line then as many lines as necessary are used. Each line contains as many elements as possible with line breaks occurring just after the XOR keyword and the subsequent lines aligned with the first expression element on the line above.

```
x XOR y XOR z
```

```
x XOR y XOR
z XOR q
```

6.3.26 Aggregate Initializer Expression

This is a sequence of expressions each separated by a comma and enclosed in square brackets.

Each expression is written with a comma separating any two adjacent expressions. At least one space is written after the comma with no space before the comma. The first expression starts just after the left square bracket and the right square bracket occurs just after the last expression.

If the aggregate initializer expression can't fit on one line then line breaks are placed just after the comma. Each element is written to a separate line aligned with the expression above it.

```
[42, x + y, 'this', .that.]
```

```
[15,
 a + very + long + expression,
 'this string',
 .enum_val.]
```

6.3.27 Call Function Expression

Write the name of the FUNCTION. If there are any parameters to the function, then the parameter expressions are enclosed in parentheses and separated by commas.

If there are parameters then the left parenthesis is written just after the function name with no space separating it from the function name. Each expression is written with a comma separating any two adjacent expressions. There will be at least one space after the comma and no space before the comma.

If more than one line is needed then the expressions are written one to a line with the line break occurring just after the comma. Each expression is aligned with the expression above it. The first expression starts just after the left parenthesis and the right parenthesis is written just after the final expression.


```
fun1(42, .t., 'this')
fun1(42,
     .t.,
     'this')
```

6.3.28 ENTITY Constructor Expression

Write the name of the ENTITY to be constructed.

If there are values needed to initialize any attributes, then the list of expressions is written just after the entity name and enclosed in parentheses and separated by commas.

This list begins with a left parenthesis just after the ENTITY name with no space between the parenthesis and the entity name. Each expression is written with a comma separating them. There is at least one space after the comma and no space before the comma. The right parenthesis follows immediately after the last expression.

If all of the expressions won't fit on one line then each expression is placed on its own line with line breaks occurring just after the comma. The first expression occurs just after the

left parenthesis and on the same line. Each subsequent expression is aligned with the one above it.

```
entity1('this', 42, (1,2,4,6), .enum_val1.)
entity1('this',
        42,
        (1,2,4,6),
        .enum_val1.)
```

6.3.29 Enumeration Reference Expression

If the name of the TYPE declaration is provided, write the name of the TYPE declaration followed by a period ('.').

Write the enumeration value.

```
enum1
type1.enum2
```

6.3.30 Interval Expression

Write an open curly brace { . Write the low value expression. Write the first operator. Write the item value expression. Write the second operator. Write the hi value expression. Write a close curly brace } .

If the expression can't fit on one line, then write the low value expression on the line with the left curly brace followed by the first operator. Write the item value expression on the next line followed by the second operator. The item value expression will be aligned with the low value expression on the line above. Finally, on the next line write the hi value expression followed by the right curly brace. The hi value expression will be aligned with the item value expression on the line above.

```
{1 < x < 10}
{1 <
 x<
 10}
```

6.3.31 Qualifiable Factor Expression

This qualifiable-factor object contains both the factor and the qualifiers. Write the factor. For each qualifier, write the qualifier.

If they need more than one line, put as many qualified on each line as possible and align each subsequent line with the first qualifier on the line above.

```
var1\ENTITY1.attr1[23]
var1\ENTITY1
    .attr1[23]
```

6.3.32 QUERY Expression

Write the QUERY keyword followed by an open parenthesis. Next write the query variable id followed by the <* symbol.
Next write the source expression followed by the | symbol.
Next write the query expression followed by the close parenthesis.

If the query expression can't fit on one line then put line breaks just after the <* and | symbols. The source expression will be aligned with the query variable id. The query expression will be aligned with the source expression.

```
QUERY(var1 <* [1,2,3,4,5] | var1 < 3)
QUERY(var1 <*
    [1,2,3,4,5] |
    var1 < 3)
```

6.4 Statements

If the statement fits on one line, then it is done that way.

If it does not fit on one line, then it will use as many lines as necessary and will be formatted appropriately for the specific statement.

6.4.1 ALIAS Statement

Write the keyword ALIAS followed by the variable id followed by the keyword FOR followed by a general reference and set of qualifiers for the thing being aliased. Finish up with a ';' (semicolon) to end it.

```
ALIAS var_id FOR ent1\attr1;
```

6.4.2 Assignment Statement

Write a pretty version of the general ref followed by the assignment keyword ':=' followed by a pretty version of the expression followed by a ';' (semicolon). At least one space should be included before and after the assignment keyword ':='.

Multiple lines will be used depending on the expression itself.

```
i := 23 * 42 + x;
```

6.4.3 Call PROCEDURE Statement

Write the name of the procedure.

If there are parameters to be passed to the procedure, then write an open parenthesis followed by a list of the parameter expressions each separated by a comma. Finally write a close parenthesis.

Finish up with a semicolon.

If the expressions won't fit on a single line, then place line breaks just after the comma. Each expression will be placed on its own line aligned with the expression on the line above. The first expression is placed on the line with the left parenthesis and just after the parenthesis.

```
proc_name(42,'this',.T.,.THAT.);
proc_name(42,
    'this',
    .T.,
    .THAT.);
```

6.4.4 CASE Statement

Write the keyword CASE followed by the expression that is the <selector> followed by the keyword OF.

Write each of the case actions starting on a separate line indented two characters to the right from the left edge of the CASE keyword.

After the CASE Actions, if there is an otherwise clause, write the keyword OTHERWISE on a separate line, indented two characters to the right of the left edge of the CASE keyword. Follow the OTHERWISE keyword with a colon and the statement that is the otherwise clause. The colon should be right next to the OTHERWISE keyword with one or more spaces after the colon.

Finally, on a new line write the keyword END CASE aligned with the CASE keyword. Follow the END CASE keyword with a semicolon.

```
CASE <selector> OF
 label1: <stmt>
 label2, label3:
  <stmt>
 OTHERWISE: <stmt>
END_CASE;
```

6.4.5 CASE Action

Each CASE Action begins on its own line, indented two characters to the right of the left edge of the CASE keyword.

Write each specified label separated by a comma.

Next write a colon next to the last label. The colon should have at least one space after it.

Next pretty print the statement that is the body. If the statement can't fit on the line with the label(s), then start it on the next line indented two characters to the right of the left edge of the first label.

```
label1, label2: <stmt>
label3, label4, label5:
 <stmt>
```

6.4.6 Compound Statement

The compound statement begins with the BEGIN keyword and ends with the END keyword. Each of these keywords should be on its own line and aligned with each other. Following the END keyword there should be a semicolon.

Each of the enclosed statements should be written on a separate line indented two characters to the right of the left edge of the BEGIN keyword.

```
BEGIN
 a := 42 * i;
 RETURN(a);
```

```
END
```

6.4.7 ESCAPE Statement

The ESCAPE statement is self-contained and consists of just the ESCAPE keyword followed by a semicolon.

```
ESCAPE;
```

6.4.8 IF Statement

The IF statement consists of the keyword IF followed by the expression that is the condition. Next follows the THEN keyword and the statements that constitute the then portion.. If there is an else part of the if then the ELSE keyword is written followed by the else statements. The THEN and ELSE keywords each have their own line and are aligned with the IF keyword. The THEN and ELSE statements are indented two characters to the right of the left edge of the associated keyword.

Finally, on a new line, aligned with the IF keyword we write the END IF keyword followed by a semicolon.

```
IF <condition>
THEN
 a := 42 * i;
 RETURN(a);
ELSE
 a := x / 15;
 RETURN(a);
END_IF;
```

6.4.9 NULL Statement

The NULL statement is self-contained and consists of only a semicolon.

```
;
```

6.4.10 REPEAT Statement

The REPEAT Statement consists of the REPEAT keyword followed by the Increment Controls, followed by any While Controls, followed by any Until Controls, followed by a semi- colon, followed by the statements that should be executed during each loop. Finally, there is a END REPEAT keyword written on its own line aligned with the REPEAT keyword and followed by a semicolon.

The Increment Controls, While Controls, and Until Controls should be indented four characters to the right of the left edge of the REPEAT keyword.[6]

```
  REPEAT i := 0 TO 15 BY 1
    WHILE x > 20
    UNTIL y < 50;
   a := 42 * x / y;
   b := x * i DIV y;
END_REPEAT;
```


6.4.11 Increment Control

An Increment Control consists of a variable id followed by an assignment operator followed by an expression that calculates the initial value of the variable followed by the TO keyword, followed by an expression that calculates the end value of the variable, optionally, followed by the BY keyword and an expression the calculates the increment value.

```
a := 1 TO 5
a := 1 TO 5 BY 0.5
```

6.4.11.1 While Control

A While Control consists of the keyword WHILE followed by an expression that returns false when the REPEAT should stop.

```
WHILE a < 5
```

6.4.11.2 Until Control

An Until Control consists of the keyword UNTIL followed by an expression that returns true when the REPEAT should stop.

```
UNTIL x > 50
```

6.4.12 RETURN Statement

The RETURN statement is fairly simple and consists of the keyword RETURN optionally followed by an open parenthesis, an expression which calculates the value to be returned, a close parenthesis.

Finally, it is terminated with a semicolon.

```
RETURN;
RETURN(42);
```

6.4.13 SKIP Statement

The SKIP statement is self-contained and consists of the keyword SKIP followed by a semicolon.

```
SKIP;
```

6.5 Style rules

These style rules specify capitalization and choosing and using names for objects in a schema. Except for attribute names, EXPRESS identifiers shall be unique across all parts of ISO 10303.

NOTE The namespace for unique identifiers extends to parts of other SC 4 standards that are deemed to be "common resources".

6.5.1 Use of case

EXPRESS reserved words shall be written in upper case letters; everything else should be written in lower case.

6.5.2 Names

An EXPRESS schema may use a large number of identifiers. Because these identifiers are very suggestive, clarity, avoidance of ambiguity, similarity, and uniqueness are important considerations in choosing an identifier name. The names chosen for EXPRESS elements should reflect the meaning of the element and should complement the natural language definition of the element. However, names should not be a substitute for a definition, or repeat the definition. Obviously, names should not conflict with or contradict the definition of the element.

The following general rules apply to naming of elements in an EXPRESS schema.

— Avoid confusion with similar names in the immediate context.
— Use the shortest possible names. Do not use prefixes and suffixes such as is_, a_, the_, _set, _array, and _list.
— Use plural name forms for aggregates;
— Use nouns or noun phrases for names of types, entity data types, and attributes that will appear in an EXPRESS long form;
— Use verbs or verb phrases for names of actions; and
— Separate name components by underscores.

6.5.2.1 SCHEMA names

Schema names in the ISO 10303 integrated resources shall end with "_schema". Schema names in ISO 10303 application interpreted constructs shall begin with "aic_". ARM schema names in ISO 10303 modules shall end with "_arm". MIM schema names in ISO 10303 modules shall end with "_mim".

6.5.2.2 Names of types

The name chosen for a type should reflect the nature of the type.

If the type defines a constraint on a base type, the name should reflect the constraint.

EXAMPLE 1 A type whose base type is `INTEGER` and that constrains the domain to be even numbers is named "even_number".

If the type is intended to add meaning to a base type, the name should reflect the meaning.

EXAMPLE 2 A type whose base type is `STRING` and is intended to be used to identify types of part is named "part_identifier".

A select type should be named in terms of the role that the different members of the type can play.

EXAMPLE 3 A select type whose domain is the entity data types **person** and **organization** and indicates that a person or an organization can own something is named "owner".

NOTE 1 The suffix "_select" may be included in the name of a select type if this suffix helps to disambiguate the type from other EXPRESS elements in a given schema or collection of schemas.

NOTE 2 Particular conventions apply to the naming of select types that are used in completion of the management resource templates in ISO 10303-41 (10303-41:2019). These select types are named by combining a word or phrase that stands for the administrative concept that is associated with product data and the suffix "_item".

EXAMPLE 4 A select type whose domain is the set of entity data types with which an approval can be assigned is named "approval_item".

The names used of an enumeration data type shall be distinct from the names of schemas, entity data types, and types. It is often appropriate for these to be adjectival; for example, use "planar" rather than "plane" as the name of an enumeration data type. However, different enumeration types may contain the same name.

6.5.2.3 Names of entity data types

The name chosen for an entity data type should reflect the class of "real world" objects or concepts that the entity data type represents. The name should make use of accepted terminology in the field supported by the schema in which the entity data type is defined.

NOTE This approach is not always feasible in generic schemas (such as the ISO 10303 integrated resources) that are intended to support a wide range of fields. In this case it is important to choose a name that is neutral with respect to the possible applications of the entity data type, and to provide a definition that accurately convey the generic meaning of the entity data type.

Do not prefix entity data type names with definite or indefinite articles "the_" or "a_".

EXAMPLE 1 An entity data type that represents documents should be named “document”, not “the_document” or “a_document”.

Entity data type names should be singular, not plural.

EXAMPLE 2 An entity data type that represents documents should be named “document”, not “documents”.

The name of an entity data type may reflect the name of other entity data types to which it is related in a subtype/supertype hierarchy. The name may reflect the following:

— how a subtype differs from its supertype;
— how one subtype differs from other subtypes (of a common supertype).

In an ISO 10303 integrated resource, the name of a subtype should reflect the specialized meaning or intended usage of the subtype.

EXAMPLE 3 ISO 10303-41 defines an entity data type named **product_definition_relationship** that represents associations between instances of the **product_definition** entity data type. A subtype of **product_definition_-relationship** defined in ISO 10303-44 specializes its meaning such that one of the related instances of **product_-definition** represents a component in an assembly that is represented by the other instance of **product_-definition**. The name of this subtype is **assembly_component_usage**.

In the application interpreted model of an ISO 10303 application protocol, or a module interpreted module, the name of a subtype of an entity data type defined in the integrated resources may reflect the usage of the resource construct in the application protocol or module.

Do not use the schema name as the prefix to an entity data type name: entity data type **unit** of schema **x** is referred to as **x.unit** from another schema. However, it is sometimes necessary to use the name of a supertype as a suffix for subtype entity data types.

EXAMPLE 4 The suffixes “_curve” and “_surface” are necessary in the entity data type names **b_spline_curve** and **b_spline_surface** because “b_spline” alone would cause a name clash.

In a like manner, the name of a supertype sometimes is useful as a prefix to a subtype name.

EXAMPLE 5 The supertype name is used as a prefix in the name of the entity data type **surface_of_revolution**.

6.5.2.4 Names of attributes

The name chosen for an attribute should reflect the role that the domain of the referenced type plays in the entity data type in which the attribute is declared. Use singular names for single-valued attributes. Use a plural name for an attribute whose domain is an aggregate.

Do not use the entity data type name as a prefix of an attribute name. Attribute **test** of entity **x** is referred to as **x.test** from outside scope of that entity data type.

Do not append “_set”, “_array”, “_bag”, or “_list” to an attribute type.

EXAMPLE Use “knots” instead of “knot_array”.

Do not use the type name as the attribute name even though EXPRESS allows it. The attribute name should reflect the role the type is playing in the definition of the entity data type.

6.5.2.5 Names of rules, functions, and procedures

When choosing a name, the use that will be read most often takes priority. For example, the executable code of a function will not be reviewed very often, but the function name will be. Consequently, the name should read naturally in a domain (`WHERE`) rule rather than following the keyword `FUNCTION`. The choice does not depend on which happens to be written first: the one containing its use or the one containing its declaration. For the purposes of choosing a name, the use takes precedence.

For example, rather than using

```
FUNCTION is_extension_supplied ()
```

whose invocation would be

```
IF is_extension_supplied ( ) THEN
```

the "is_" prefix should be removed, allowing the call to be read as a phrase in English as well as in EXPRESS. In this case, the function declaration should be

```
FUNCTION extension_supplied ()
```

so that its invocation is

```
IF extension_supplied ( ) THEN
```

6.5.2.6 Clashes of attribute and function names

Attributes shall not have the same names as functions; such name conflicts are confusing in domain rules.

6.5.2.7 Clashes of entity data type names and enumeration values

Entity data types shall not have the same name as enumeration type values; such name conflicts are illegal according to ISO 10303-11.

6.5.2.8 Length of names

Although EXPRESS schemas are published as digital files, the documentation of the schema shall conform to publishing requirements based on the rules and guidelines for page size and fonts specified in the ISO/IEC Directives, Part 2, 2018. In order to avoid line breaks in EXPRESS identifiers, there is a practical limit of 60 characters on the names used for EXPRESS elements. Although such long names should be avoided, conventions for naming of entity data types and rules can produce names that exceed this limit. In this case abbreviations may be used.

EXAMPLE 1 Some SC 4 projects use a convention for naming rules that combines the name of the entity data type that is constrained by the rule with a phrase that describes the constraint applied. This convention can lead to very long names, such as "product_definition_relationship_requires_primary_classification" (63 characters). To avoid such long names, identifiers of constrained entity data types name may be abbreviated as they are used in rules. In this case, the name "pdr_requires_primary_classification" could be used.

If this approach is adopted, then the following requirements apply:

— the abbreviation shall be used consistently;

EXAMPLE 2 In the example above, all rules in the schema that constrain the product_definition_relationship entity data type should be prefixed "pdr_", not just those whose unabbreviated names would exceed 60 characters.

— the abbreviations shall be defined, and their use explained at the start of the clause or subclause in which the schema is defined.

To define and explain the use of the abbreviated names, an example of the text to use reads as follows, see 5.3. for implementing the required text.

[SC 4 required:SC4_express_abbr]

Abbreviated names are used in the identifiers of the <entity data types, rules, …> declared in this schema. Prefixes used in these identifiers have the following meanings:

<list the abbreviated forms and the full names>

[end required]

EXAMPLE 3 The following example illustrates the use of this text:

Abbreviated names are used in the identifiers of the rules declared in this schema. Prefixes used in these identifiers have the following meanings:

cgs	character_glyph_symbol
pdr	product_definition_relationship
pdu	product_definition_usage

6.5.2.9 Abbreviations and acronyms

Abbreviations and acronyms can be confusing so avoid them if possible. When you have to use them (to avoid layout problems), take care to use them consistently and to document the full meaning. When an abbreviation or acronym is used in an EXPRESS declaration, a comment shall be provided within the declaration to give the full spelling.

6.6 EXPRESS usage style requirements

6.6.1 Use of local domain (WHERE) rules

WHERE rules are used for at least three different purposes:

— To state aspects of the definition of an entity data type in a formal manner. Typically, these are tests that will be true by definition, whatever the attribute values are.

— To define the desired behaviour of the entity with respect to certain known queries.

— To define tests that can be applied to an instance of the entity to ensure that it is a valid instance.

EXAMPLE The following contains one of each of these uses (in the same order):

```
ENTITY line;
  start : point;
  dir   : direction;
WHERE
  WR1: arc_length_extent(line) = infinity;
  WR2: coordinate_space(line) = coordinate_space(start);
  WR3: start.x > 0.0;
END_ENTITY;
```

Of these three, only the third shall be used. The first constraint should be added to the definition of the entity data type while the second constraint should either be treated using derived attributes or left until a future version of EXPRESS provides a clearer means of establishing the behavioural characteristics of entity implementations.

Large functions that test several aspects of an entity should not be used. Split the function into several smaller functions.

6.6.2 Labelling UNIQUE and WHERE rules

Each label within the scope of an entity data type shall not be used by more than one UNIQUE rule within that scope. The form of the label shall be URn where n is an integer giving the position of the rule in the list. The list shall not have any gaps in the sequence from the first entry to the last.

Each label within the scope of an entity data type shall not be used by more than one WHERE rule within that scope. The form of the label shall be WRn where n is an integer giving the position of the rule in the list. The list shall not have any gaps in the sequence from the first entry to the last.

NOTE Several legacy resources use a short English word for a label. That usage is acceptable only for those existing labels.

6.6.3 Definition of constraints

An entity data type can be constrained by any of the following three methods:

— definition of the structure of the entity data type;

— formal proposition: a computable constraint written in EXPRESS, for example local and global rules, subtype constraints, and inverse attributes;

— informal proposition: constraints that cannot be documented formally.

The constraints expressed in the definition of the entity data type, in the formal propositions, and in the informal propositions shall not be redundant. There shall be as little overlap as possible between the restrictions placed on the entity data type among these three types of constraints.

Any characteristics of the entity data type that are part of its definition and are unaffected by attribute values are true at all times and do not need to be reiterated in the constraints and propositions.

6.7 EXPRESS schema documentation

NOTE Any of the requirements below that only apply to integrated resources are indicated as such.

6.7.1 General requirements

The title of each clause or subclause in which a schema is declared shall be the schema name (without the "_schema" suffix, if this is part of the name), with underscores replaced by spaces, and with the initial character in upper case.

EXAMPLE 1 The title of the clause that documents a schema whose name is "dynamic_fluid_flow" is "Dynamic fluid flow".

For parts of ISO 10303 that are in the integrated resources series, each schema shall be documented in a separate clause.

When referring to the elements of the EXPRESS language, EXPRESS reserved words (such as `SCHEMA`, `ENTITY`, `WHERE`) shall appear as all caps in a monospaced font.

NOTE 1 Such references in the documentation of a schema should not be necessary, as these words refer to the elements of the language itself.

When referring to elements of the schema, words and phases such as "type", "schema", and "entity data type" shall not be capitalized.

EXAMPLE 2 In text that refers to an entity data type, use the phrase "the <entity data type name> entity data type, not "the <entity data type name> `ENTITY`".

In the documentation of a schema, the same English language words may be used to refer to an object in the real world or to a concept, and as the name of an EXPRESS data type that represents this object or concept. Use the following typographical convention to distinguish between these. If the referent is the object or concept, the word or phrase occurs in the same typeface as narrative text. If the referent is the EXPRESS data type, the word or phrase occurs in a bold typeface. Use bold typeface for the names of EXPRESS schemas, functions, rules, procedures, and attributes of entity data types where they appear in general text.

The name of an EXPRESS data type may be used to refer to the data type itself, or to an instance of the data type. The distinction between these uses is normally clear from the context. If there is a likelihood of ambiguity, the phrase "entity data type" or "instance(s) of" is included in the text.

NOTE 2 See the SC 4 Quality manual for the procedure for validation of EXPRESS schemas.

6.7.2 Components of a clause that specifies an EXPRESS schema

Each clause that specifies an EXPRESS schema shall contain the following subclauses in the order given below:

x <schema name>

x.1 Introduction

x.2 Fundamental concepts and assumptions

x.3 <schema name> type definitions

x.4 <schema name> entity data type definitions

x.5 <schema name> rule definitions

x.6 <schema name> function definitions

If there is nothing contained in one of the subclauses, the subclause shall be omitted, and the remaining subclauses renumbered accordingly.

The subclause containing declarations of entity data types may also be titled "<schema name> entity definitions". If this form is used, it shall be used consistently for all schemas defined in one part.

6.7.3 Schema documentation requirements

The following rules shall apply to documenting the schema.

An example of the wording to introduce the schema clause is given below, see 5.3. for implementing the required text.

[SC 4 required:SC4_express_schemaClause]

This clause defines the information requirements to which implementations shall conform using the EXPRESS language as defined in ISO 10303-11. The following EXPRESS declaration begins the <schema name> and identifies the necessary external references.

[end required]

The <schema name> should be replaced by the name of the schema as it appears in the schema declaration. This wording shall appear immediately following the clause heading. If there are no external references, the words "and identifies the necessary external references" shall be omitted.

The above text shall be followed by the schema declaration. The declaration shall include any interface statements (`USE FROM` or `REFERENCE FROM`) necessary for the schema. An example of the required text for the declaration is given below, see 5.3. for implementing the required text.

[SC 4 required:SC4_ express_schema-1]

EXPRESS specification:

```
*)
SCHEMA <schema_1 name>;

REFERENCE FROM <schema_2 name>
  (<e1 name,
   e2 name>);
(*
```

[end required]

Where interface statements occur, place a note following the schema declaration that identifies the source(s) of the interfaced schema(s). These sources may be any of the following:

— another clause of the same part;
— another part of the same SC 4 standard;

— a different standard.

In the second and third cases the standard in which the interfaced schema is declared shall be listed as a normative reference (see ISO/IEC Directives, Part 2, 2018, Clause 15).

An example of the notes is given below:

[SC 4 required: SC4_ express_schema-2]

NOTE X The schemas referenced above are specified in the following documents:

<schema_1> Clause <n> of this document

<schema_2> ISO <ISO standard and part number>

NOTE X See Annex x for a graphical representation of this schema.

[end required]

The elements of the note are arranged in two left-justified columns, without punctuation. The schema name shall be presented in bold face.

EXAMPLE The following example illustrates this type of note:

NOTE The schemas referenced above can be found in the following parts of ISO 10303:

product_definition_schema	ISO 10303-41
geometry_schema	ISO 10303-42
representation_schema	ISO 10303-43

6.7.4 Introduction to schema

The introduction to the schema shall include the objectives of the schema and a description of its major components and key concepts.

This subclause is primarily text but may contain figures, such as an EXPRESS-G diagram that presents an overview of the entity data types contained in the schema. If included, this figure shall be referenced from a note.

An example of the required text to begin the introduction to each schema for schemas defined in integrated resources parts of ISO 10303 is given below, see 5.3 for implementing the required text.

[10303 required: 10303_express_schema-3]

The subject of the <schema name> is . . . .

[end required]

6.7.5 Fundamental concepts and assumptions

The fundamental concepts and assumptions are declarations of fact about the subject area of the schema. These facts have been used as the basis for developing the integrated resource and are essential to the reader's understanding and using the part.

Fundamental concepts and assumptions may be expressed in a general or structured form. The general form shall be text that describes the concepts and assumptions that underlie the schema. The structured form shall be a list formatted as described in ISO/IEC Directives, Part 2, 2018, Clause 23.

Fundamental concepts that apply to the entire part covering multiple schemas shall be documented in an additional clause immediately following the terms and abbreviations clause. Extra clauses may be included if appropriate to precede collections of related schemas.

6.7.6 Documentation of formal propositions

Formal propositions follow the EXPRESS declaration (types and entity data types), the definition of enumeration items (types only), or argument definitions (rules). Formal propositions are constraints that are computable, are written in EXPRESS, and are placed within the `WHERE` clause of the declaration of a type, entity data type, or rule. The following rules apply to formal propositions in the documentation of types (see 6.7.8), entity data types (see 6.7.9), or rules (see 6.7.10).

— The formal propositions shall be preceded by the underlined title "Formal propositions:".

— When there is a local rule label in the EXPRESS specification, each formal proposition shall start with the local rule label and be followed immediately by a colon and a single space. The label shall be in boldface. The colon shall not be boldface.

EXAMPLE The following examples illustrate the layout and format of formal propositions:

WR1: The value of x shall be positive.

UR1: The name shall be unique.

— The ISO required verbal forms "shall" (see ISO/IEC Directives, Part 2, 2018, Clause 7) shall be used.

— Any additional explanation or examples shall be provided as notes (see ISO/IEC Directives, Part 2, 2018, Clause 24) or examples (see ISO/IEC Directives, Part 2, 2018, Clause 25).

— The order of the formal propositions shall be the same as the order of the constraint specifications in the EXPRESS declarations.

— There shall be a one-to-one correspondence between the local rules stated in the EXPRESS declaration (`WHERE`, `INVERSE`, and `UNIQUE` constraints) and the elements in the list of formal propositions.

— If a local rule uses a call to an EXPRESS function, the effect of the tests within that function as they are applied to this type or entity data type shall be briefly described. A statement that the function shall return the value `TRUE` is not adequate. A local rule is satisfied if the evaluated result of the rule expression is `TRUE` or `UNKNOWN`; a function used within a local rule may never return `UNKNOWN`.

6.7.7 Documentation of informal propositions

Informal propositions are un-computable constraints that cannot, or cannot reasonably, be written in EXPRESS, although each informal proposition still represents a requirement. If an EXPRESS declaration exists or EXPRESS-like pseudo-code has been written, it may be included in an informative annex as a technical discussion. Each informal proposition shall be presented as follows.

— The informal propositions shall be preceded by the underlined title "Informal propositions:".

— Each informal proposition shall be given a label, corresponding to the local rule labels that appear in formal propositions. By convention, informal propositions in ISO 10303 parts are labelled IP1, IP2, and so on.

NOTE Several legacy resources use a short English word for a label. That usage is acceptable only for those existing labels.

— The ISO required verbal forms "shall" (see ISO/IEC Directives, Part 2, 2018, Clause 7) shall be used.

— Any additional explanation or examples shall be provided as notes (see ISO/IEC Directives, Part 2, 2018, Clause 24) or examples (see ISO/IEC Directives, Part 2, 2018, Clause 25).

The explanation for each information proposition shall state the conditions and requirements that shall be met by instances of the type or the entity data type.

6.7.8 Type documentation requirements

The following rules apply to the documentation of types.

— Document each type in the "<schema name> type definitions" subclause in a separate subclause. The title of the subclause shall be the name of the type exactly as it appears in the EXPRESS declaration (lower case with underscores).

— The title shall be followed by a textual definition of the type and any supporting material necessary to define the intent of the type. In particular, this text should demonstrate how this type is different from any other similar type.

— The EXPRESS declaration shall be given next using the format described in 6.2.9, separated from the text by comment markers as described in 6.2.4. The title "EXPRESS specification:" shall be placed immediately before the close-comment marker.

— If the type is an enumeration of items, the items may be defined following the EXPRESS declaration. Definitions of enumerated items shall be given for clarity, unless the item corresponds exactly to a term defined in the terms and definitions clause (see ISO/IEC Directives, Part 2, 2018, Clause 16) of the standard. If the enumeration item corresponds to a defined term, a reference to the definition of the term shall be included as a note. The title "Enumerated item definitions:" shall precede the definitions of the enumeration items. Each enumerated item definition shall consist of the identifier of the item in boldface, a colon, one space, and the definition of the item.

— If the type is an extensible select, the definition shall begin with a required wording, followed by any necessary explanation of the domain concepts. An example is given below, see 5.3 for implementing the required text.

[10303 required:10303_express_schema-select]

The <name of extensible select> type is an extension of the <name of select> type. It adds the data types <list of data types> to the list of alternate data types.

NOTE The list of entity data types may be extended in application modules that use the constructs of this module.

[end required]

— If the type is an extensible enumeration, the definition shall begin with the following wording, followed by any necessary explanation of the domain concepts.

[10303 required:10303_express_schema-enum]

The <name of extensible select> type is an extension of the <name of select> type. It adds the enumeration items <list of enumeration items > to the list of alternate enumeration items.

NOTE The list of enumerations may be extended in application modules that use the constructs of this module.

[end required]

— Formal propositions (see 6.7.6) follow the EXPRESS declaration or the definition of enumeration items.
— Informal propositions (see 6.7.7) follow the formal propositions

6.7.9 Entity data type documentation

6.7.9.1 General requirements

The entity data types declared in a schema shall be documented in a subclause titled ““<schema name> entity data type definitions” or “<schema name> entity definitions”. Entity data types may be collected into logical groups in order to enhance the readability and understandability of the schema. If such groups are used (there shall be at least two such groups), the following structure should be used for the entity definition subclause.

x.y.1 <schema name> entity data type definitions: <logical group name 1>

x.y.2 <schema name> entity data type definitions: <logical group name 2>

x.y.3 <schema name> entity data type definitions: <logical group name 3>

…

x.y.n <schema name> entity data type definitions: <logical group name n>

All EXPRESS entity data types shall be at the same subclause level within each group.

All EXPRESS entity data types within a given functional grouping should be presented in an order that will aid understanding. An obvious and common ordering will present the EXPRESS entity data types according to the subtype/supertype hierarchy relationships among the entity data types.

If there is no other reasonable order, the entities shall appear in alphabetical order.

6.7.9.2 Documenting a single entity data type

NOTE 1 See 6.9.2 for an example of the documentation of an entity data type.

The following rules apply to documenting an entity data type.

- Each entity data type definition shall be a new subclause. The title of the subclause shall be the name of the entity data type exactly as it appears in the EXPRESS declaration (lower case with underscores). See 6.5.2.3 for the requirements that apply to naming of entity data types.
- The definition of an entity data type shall state clearly the following:
  - the concept that the entity data type represents;
  - the information about the concept that is represented in the data structure and constraints defined by the entity data type.

  The name of the entity data type without underscore and in normal text (not boldface) may be used to stand for the concept that the entity data type represents.
- The follow convention may be used to simplify entity data type definitions. The phrase “an **<entity_data_type_name>** is/represents …” may be used as a short hand for “An instance of the **<entity_data_type_name>** entity data type is/represents …”.

  If this convention is used, it shall be used consistently for all entity data type definitions in the standard; the convention itself shall be described in the introduction of the standard.
- Examples may be provided to clarify the concept that is represented by the entity data type or to illustrate the population of the entity data type and its attributes. It shall be clear whether each example refers to the concept represented by the entity data type or the data that is governed by the entity data type. Examples follow the prose definition.
- Extra explanations and references to other sources for explanations should be given as one or more notes.

— Tables or figures may be included in the definition of an entity data type. If the information conveyed in the table or figure is essential to understanding of the entity data type, the table or figure shall be referenced from the normative text of the definition so that it is itself normative. If the information conveyed in the table or figure enhances but is not essential to understanding the entity data type, the table or figure shall be referenced from a suitable note or example so that it is itself informative.

The EXPRESS declaration for the entity data type follows the definition. The declaration shall be introduced by the underlined title "EXPRESS specification:" and delimited by comment markers as specified in 6.2.4.

Following the EXPRESS declaration, all attributes (both explicit and derived) shall be documented. The attribute definitions shall be introduced by the underlined title "Attribute definitions:". The following rules apply to the documentation of attribute definitions:

— The attributes shall be documented in the same order as they appear in the EXPRESS declaration.
— The attribute definitions shall be presented as follows.
    — Each attribute definition shall start with the attribute name exactly as given in the EXPRESS declaration (complete with underscores), in boldface, and followed by a colon.
    — The definition of the attribute shall follow the name of the attribute, starting on the same line.
    — The definition of the attribute shall describe the role of that attribute in the entity data type. If the attribute uses another entity data type or type, there is no need to give a definition for the referenced item.

NOTE 2 If it appears necessary to redefine the referenced item, indicating that meaning of the referenced type varies according to its use, consider defining a new intersection entity data type.

— Additional explanation may be given as notes.

— Examples that illustrate the population or usage of the attribute may also be given.

Formal propositions (see 6.7.6) follow the attribute definitions. Formal propositions shall only be included where the result of the evaluation depends on the values of the attributes, the complex type of instances, or both. If the formal proposition always returns true, it shall not be included as a formal proposition but rather should be included as part of the definition of the entity data type.

Informal propositions (see 6.7.7) follow the formal propositions.

6.7.9.3 References to attributes declared in supertypes

A reference to an attribute declared in a supertype may be explained with a note following the first use of the attribute name.

EXAMPLE The following example illustrates the wording of such a note:

NOTE The attribute <a_name> is declared in the <e_name> entity data type of which this entity data type is a subtype.

Phrases that reflect particular implementation considerations, such as "inherited attribute", should not be used.

6.7.9.4 Plurals of

Avoid using plurals of EXPRESS object names by an alternate usage, such as "several instances of the **vertex** entity data type." If necessary, plurals of EXPRESS object names may be made by adding an "s" (not in boldface) to the end of the name. This includes names for which the plural in English changes the spelling of the word.

EXAMPLE The following example illustrates the addition of an "s" to refer to multiple instances of the vertex entity data type: "An **open_path** visits its **vertex**s exactly once.".

NOTE The wording specified in the example above is much clearer if changed to: "Each instance of **open_path** visits each of its instances of **vertex** exactly once.".

6.7.10 Rule documentation requirements

The following requirements apply to documenting rules.

— All rules shall be declared/defined in the "<schema name> rule definitions" subclause.

— Each rule shall constitute a new subclause. The title shall be the name of the rule exactly as it appears in the EXPRESS declaration (lower case with underscores). This name should not be abbreviated and should comprise, where possible, proper English words (see 6.5.2.8 for constraints on the maximum length of EXPRESS identifiers).

— If there is only one rule declaration in a schema, the rule declaration shall appear in a single subclause titled "<schema name> rule definition: <rule name>".

— The title shall be followed by a prose definition and any supporting text necessary to state the intent of the rule.

— The EXPRESS declaration shall follow the definition, preceded by the underlined title "EXPRESS specification:".

— The arguments of the rule shall be defined following the EXPRESS declaration, preceded by the underlined title "Argument definitions:".

— The argument definitions shall be presented as follows.

  — Each argument definition shall start with the argument name exactly as given in the EXPRESS declaration (complete with underscores), in boldface, followed by a colon.
  — The definition of the argument shall follow the name of the argument, starting on the same line.

— Each constraint within the `WHERE` clause of the rule shall have a unique label. Unless an appropriate short English word can be used, the form of the label shall be WRn where n is an integer giving the position of the rule in the list.

NOTE By convention, schemas defined in parts of ISO 10303 use the WRn form only.

— Each constraint within the `WHERE` clause of the rule shall be documented as a formal proposition (see 6.7.6). The formal propositions follow the argument definitions.

— If a constraint is dependent on an unelaborated function or procedure (see 6.7.12), this should be stated in a note.

EXAMPLE The following example illustrates the wording of such a note.

NOTE This rule is based on an unelaborated EXPRESS function.

6.7.11 Subtype constraint documentation requirements

The following requirements apply to documenting subtype constraints.

— All subtype constraints shall be declared/defined in the "<schema name> subtype constraint definitions" subclause.

— Each subtype constraint shall constitute a new subclause. The title shall be the name of the subtype constraint exactly as it appears in the EXPRESS declaration (lower case with underscores). This name should not be abbreviated and should comprise, where possible, proper English words (see 6.5.2.8 for constraints on the maximum length of EXPRESS identifiers).

— If there is only one subtype constraint declaration in a schema, the declaration shall appear in a single subclause titled "<schema name> subtype constraint definition: < subtype constraint name>".

— The title shall be followed by a prose definition and any supporting text necessary to state the intent of the subtype constraint.

— The EXPRESS declaration shall follow the definition, preceded by the underlined title "EXPRESS specification:".

6.7.12 Function (procedure) documentation requirements

6.7.12.1 General requirements

The following rules apply to documenting function (or procedure) definitions:

— All functions shall be declared/defined in the "<schema name> function definitions" subclause.

— Each function shall constitute a new subclause. The title shall be the name of the function exactly as it appears in the EXPRESS declaration (lower case with underscores). The name should not be abbreviated and should comprise, where possible, proper English words (see 6.5.2.8 for constraints on the maximum length of EXPRESS identifiers).

— If there is only one function declaration in a schema, it should appear in a single subclause titled <schema name> function definition: "<function name>".

— The title shall be followed by a definition and any supporting text necessary to define the intent of the function.

— The EXPRESS declaration shall follow the definition, preceded by the underlined title "EXPRESS specification:".

— The arguments of the function shall be defined following the EXPRESS declaration, preceded by the underlined title "Argument definitions:".

— The argument definitions shall be presented as follows.

  — Each argument definition shall start with the argument name exactly as given in the EXPRESS declaration (complete with underscores), in boldface, followed by a colon.
  — The definition of the argument shall follow the name of the argument, starting on the same line. Each definition shall include whether the argument is an input, output, or both, and enumerate and define any error conditions that may result from the function.

EXPRESS functions and procedures and their application to entities may be documented by three different methods according to the completeness of the specification included in the part. These methods are as follows:

— functions and procedures that are fully specified in EXPRESS (see 6.7.12.2);

— functions and procedures that cannot, or cannot easily, be specified in EXPRESS but can be implemented within a specific application system (see 6.7.12.3);

— functions or procedures that cannot be implemented at all, either within EXPRESS or within an application system (see 6.7.12.4).

When the EXPRESS declaration of a function is not or cannot be explicitly specified, an EXPRESS comment should replace the body of the function stating the following:

— why the appropriate EXPRESS language statements are missing;

— what the function is intended to do.

6.7.12.2 Functions and procedures fully specified in EXPRESS

Functions and procedures fully specified in EXPRESS shall be documented as follows.

— The full EXPRESS specification of the function or procedure shall be documented in an appropriate subclause.

— The function or procedure shall be used as part of the definition of a constraint in one or more types or entity data types.

EXAMPLE 1 The following example illustrates a fragment of the EXPRESS declaration for a fully specified function:

```
*)
FUNCTION function_name (x:INTEGER): LOGICAL;

<function body in EXPRESS>

END_FUNCTION;
(*
```

EXAMPLE 2 The following example illustrates a fragment of the EXPRESS declaration of an entity data type that uses the function illustrated above to define a constraint:

```
*)
ENTITY foo;
...
WHERE
  WR1: function_name(...);
END_ENTITY;
(*
```

…

Formal propositions:

WR1: xxx;

6.7.12.3 Functions and procedures that can be implemented within a specific application system

Functions and procedures that cannot, or cannot easily, be implemented in EXPRESS but can be implemented within a specific application system, shall be documented as follows:

— The EXPRESS specification of the function or procedure shall include the phrase “unelaborated function/procedure” as a tail remark together with text describing the intent of the function.

— The function or procedure shall be used as part of the definition of a constraint in one or more types or entity data types.

EXAMPLE 1 The following example illustrates a fragment of the EXPRESS declaration for an unelaborated function:

```
*)
FUNCTION function_name (x:INTEGER): LOGICAL;

-- unelaborated function

(* <text that explains the intent of the function. Note that the text is
commented out between the function head and tail.>  *)

END_FUNCTION;
(*
```

The tail comment shall be inserted as shown so that it will occur in the EXPRESS listing of the schema.

An explanation of why the function is not elaborated may be placed in a note following the textual description of the function.

EXAMPLE 2 The following example illustrates a fragment of the EXPRESS declaration of an entity data type that uses the function illustrated above to define a constraint:

```
*)
ENTITY foo;
...
WHERE
  WR1: function_name(...);
END_ENTITY;
(*

...

Formal propositions:

WR1: xxx;

NOTE This proposition is based upon an unelaborated EXPRESS function or procedure.
```

The note following the formal proposition shall be included to indicate that an EXPRESS declaration for the function exists, but that it has not been elaborated.

6.7.12.4 Constraints that cannot be implemented

Not all constraints can be implemented in a uniform manner, either within EXPRESS or within an application system.

Such constraints shall be documented as follows:

— If the constraint applies to a single entity data type, it shall be defined in an informal proposition (see 6.7.7) within the documentation of that entity data type.
— If the constraint applies to two or more entity data types, it shall be defined as a pseudo-function, in a separate subclause. However, no EXPRESS declaration shall be given.

6.7.13 End of schema declaration

The EXPRESS declaration shall be the last item of the last subclause within each schema clause. The text below is shown as example, see 5.3 for implementing the required text.

```
[SC 4 required:SC4_express_schema-end]
*)
END_SCHEMA;  -- <schema name>
(*
[end  required]
```

6.8 EXPRESS-G diagram style

Each EXPRESS-G diagram shall follow the format in Annex D of ISO 10303-11 with additional rules listed below.

— For the entity data type names in EXPRESS-G diagrams, the recommended font is Times New Roman, 10pt.

— For the attribute names and labels in EXPRESS-G diagrams, the recommended font is Times New Roman, 9pt.

— If possible, each schema shall occupy one page. A schema may occupy more than one page, if necessary, and two schemas may occupy a single page if each fits onto one-half page.

— There shall be no boxes around EXPRESS-G diagrams.

— All entity data types declared in the schema shall be included.

— All select, enumeration, and defined types shall be included.

— All subtype constraints need to be included.

— Inter-schema references shall be included for all interfaced constructs.

— The abbreviations L, S, B, and A shall be used for aggregate types of attributes and types to indicate list, set, bag, and array respectively.

— Cardinality of attributes and types shall be indicated when appropriate by appending the bound specification to the aggregate abbreviation.

— Inverse and derived attributes shall be included in the EXPRESS-G diagrams and indicated by "(INV)" and "(DER)" respectively.

— The "to" end of an emphasized direction of a relationship shall be indicated by an open circle. The relationship line shall not enter the circle.

— Each attribute whose domain is a simple data type (`BINARY`, `BOOLEAN`, `LOGICAL`, `STRING`, `NUMBER`, `INTEGER`, or `REAL`) may omit the simple data type and terminate with the same open circle as used for emphasized direction.

— In ISO 10303, each attribute whose domain is one of the defined types **identifier**, **label**, or **text** may omit the defined type and terminate with the same open circle as used for emphasized direction.

NOTE The types **identifier**, **label**, and **text** are defined in the support_resources_schema in ISO 10303-41.

— The weight (thickness or width) of lines forming the symbols used for entity data types, select types, defined types, enumeration types, base types, subtype constraints, schema references, and off-page connectors shall be approximately 1mm.

— The thickness of a SUPERTYPE relationship line, select type and enumeration type BASED_ON lines shall be at least twice that of the attribute relationship lines.

— All dashed lines shall be comprised of lines and gaps with a "unit" length between 2mm and 4 mm.

— All relationship lines shall be oriented either vertically or horizontally (no diagonal lines or curves).

6.9 Examples

6.9.1 Documentation of an EXPRESS ARM

This example illustrates the required documentation for ISO 10303 application protocol ARMs that are defined in EXPRESS-G (see 6.8). The sample documentation that follows is based on the ARM EXPRESS-G diagram shown in Figure 3.

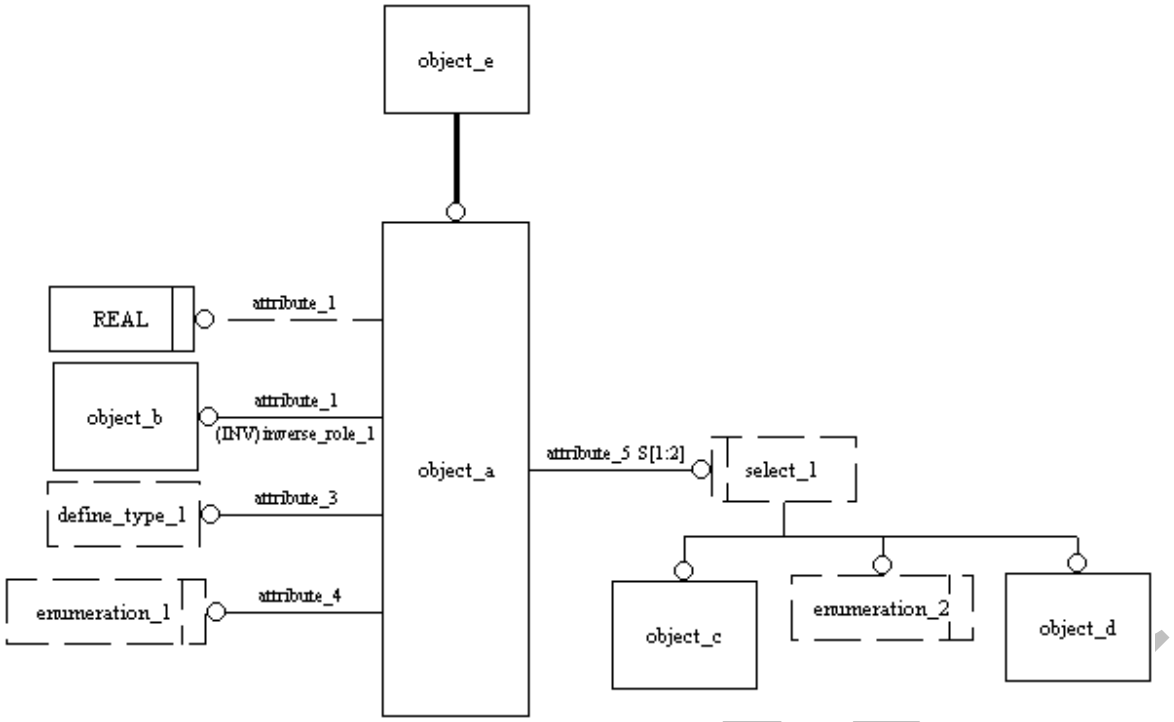

Figure 3 — ARM diagram 1 of 1 in EXPRESS-G

4.2.1 Object_a

An Object_a is a type of Object_e (see 4.2.5) that is <definition>. The data associated with an Object_a are the following:

— attribute_1;

— attribute_2;

— attribute_3;

— attribute_4;

— attribute_5.

4.2.1.1 attribute_1

The attribute_1 specifies <definition>. The attribute_1 need not be specified for a particular Object_a.

4.2.1.2 attribute_2

The attribute_2 specifies <context or role within object>. See 4.3.1 for the application assertion.

4.2.1.3 attribute_3

The attribute_3 specifies <define type definition>.

4.2.1.4 attribute_4

The attribute_4 specifies <definition>. The value of attribute_4 is one of the following:

— type_1;

— type_2.

NOTE See 4.2.1.4.1 and 4.2.1.4.2 for the definition of each allowable value for attribute_4.

4.2.1.4.1 type_1

type_1: <definition>.

4.2.1.4.2 type_2

type_2: <definition>.

4.2.1.5 attribute_5

The attribute_5 specifies <definition>. Each attribute_5 may be one of the following: object_c (see 4.2.3), object_d (see 4.2.4). See 4.3.2 and 4.3.3 for the application assertion. Or the value of attribute_5 shall be one of the following:

— type_3;

— type_4.

NOTE See 4.2.1.5.1 and 4.2.1.5.2 for the definition of each allowable value for attribute_5.

4.2.1.5.1 type_3

type_3: <definition>.

4.2.1.5.2 type_4

type_4: <definition>.

4.2.2 Object_b

An Object_b is <definition>.

4.2.3 Object_c

An Object_c is <definition>.

4.2.4 Object_d

An Object_d is <definition>.

4.2.5 Object_e

An Object_e is <definition>. An Object_e may be an Object_a (see 4.2.1).

4.3 Application assertions

4.3.1 Object_a to Object_b

Object_a has attribute_2 defined by exactly one Object_b. Object_b defines inverse_role_1 for exactly one Object_a.

4.3.2 Object_a to Object_c

Object_a has attribute_5 defined by one or two Object_c objects. Object_c defines attribute_5 for zero, one, or many Object_a objects.

4.3.3 Object_a to Object_d

Object_a has attribute_5 defined by one or two Object_d objects. Object_d defines attribute_5 for zero, one, or many Object_a objects.

6.9.2 Example entity data type declaration

This example illustrates the required documentation for entity data types (see 6.7.9).

x.x.x foo

A **foo** is xxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

EXPRESS specification:

```
*)
ENTITY foo;
  first  : INTEGER;
  second : REAL;
UNIQUE
  UR1: first,second;
WHERE
  WR1: first > 2;
END_ENTITY;
(*
```

Attribute definitions:

first: xxxxxxxxxxxxxxxxx

second: xxxxxxxxxxxxxxx

Formal propositions:

UR1: xxxxxxxxxxxxxxxxxxxxxx

WR1: xxxxxxxxxxxxxxxxxxxxxx

Informal propositions:

IP1: xxxxxxxxxxxxxxxxxxxxxx

7 ISO 8000 series: additional requirements for structure and content of parts

7.1 General

For all parts in the ISO 8000 series, the following documents provide applicable requirements:

— the ISO/IEC Directives, Part 2, 2018, *Principles and rules for the structure and drafting of ISO and IEC documents*;

— The ISO Simple template.

In addition, Clause 5 of this document (Requirements for the structure and content of parts of SC4 standards) applies.

7.2 Specific requirements

Each part of ISO 8000 should:

— include notes and examples to clarify the understanding of all key concepts in the document, typically at the point in the document when the concept first appears;

NOTE Appropriate normative content for standards consists of precise, technical statements, which are not necessarily immediately clear to the uninitiated reader. A note or example provides a less abstract view of the topic to illustrate the correct interpretation of normative text. Such notes and examples are not, however, normative, so the document is usable without them.

— describe the scope of activities to which the document applies by including an IDEF0 model;

NOTE ISO/IEC/IEEE 31320-1 (ISO/IEC/IEEE 31320-1:2012, Information technology — Modeling Languages — Part 1: Syntax and Semantics for IDEF0) specifies the syntax and semantics of IDEF0 models.

— consist of text that follows appropriate style of language to assist in the consistent and easy interpretation of meaning;

EXAMPLE Ensuring that each new sentence within a paragraph begins with a concept that has already appeared in that paragraph. This approach is shown by the underlined words in the following: "The content of ISO 9000 explains that quality is not an abstract concept of absolute perfection. Quality is actually the conformance …".

— make use of terms in a way that is consistent with the corresponding definitions in ISO 8000-2 (ISO 8000-2:2018, Data quality — Part 2: Vocabulary);

NOTE ISO 8000-2 exists to harmonize all terms and definitions across all parts of the ISO 8000 series. Project teams add new entries and propose changes as necessary to incorporate each part into this harmonized view of the vocabulary for ISO 8000.

— prior to publication include entries for relevant terms in Clause 3 and a note "Prior to publication of this document as an International Standard, the following terms and definitions will be published in ISO 8000 2 and removed from this document";

— include appropriate cross-references to other parts of the ISO 8000 series to support and promote the aggregated capability of the whole series;

— include references to appropriate entries in the Bibliography, where such entries provide additional information on the implementation and associated benefits of the document.

The published parts of ISO 8000 shall not include any terms in Clause 3.

NOTE ISO 8000-2 includes all terms for all parts of ISO 8000.

7.3 ISO 8000 series required text

For all the documents of the ISO 8000 series a specific wording is required.

This specific wording is defined and managed by the working group responsible for the series and is available in a shared repository.

The repository is located under [GIT ISO URL]/projects/ISOTC184SC4/repos/boilerplate.

Each writer shall ensure that the boilerplate text in an SC 4 standard conforms to the boilerplate text released in the repository.

A guide "How to use the required text" is at Annex B of this document.

8 ISO 10303 series: additional requirements for structure and content of parts

8.1 General

For all parts in the ISO 10303 series, the following documents provide applicable requirements:

— ISO/IEC Directives, Part 2, 2018, *Principles and rules for the structure and drafting of ISO and IEC documents;*

— the ISO Simple template;

— ISO/DIS 10303-1:2020, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

In addition, Clause 5 of this document (Requirements for the structure and content of parts of SC4 standards) applies.

The documentation of the integrated resources series is detailed in 8.2

The documentation of the application protocol series is detailed in 8.3

The documentation of the application module series is detailed in 8.4

The documentation of the domain model series is detailed in 8.5

8.1.1 Use of quotation marks

Single quotation marks are used to denote literal text string values. Double quotation marks follow standard British usage, for direct quoted material presented within other material, and titles of works following a specific bibliographic format. While quotation marks may be used to identify words being discussed as words, this usage is unlikely within the context of ISO 10303.

8.1.2 ISO 10303 series required text

For all the documents of the ISO 10303 series a specific wording is required.

This specific wording is defined and managed by the working group responsible for the series and is available in a shared repository.

The repository is located under [GIT ISO URL]/projects/ISOTC184SC4/repos/boilerplate.

Each writer shall ensure that the boilerplate text in an SC 4 standard conforms to the boilerplate text released in the repository.

A guide “How to use the required text” is at Annex B of this document.

8.1.3 Documentation of Introduction

The start of the introduction of each part of ISO 10303 is a required text. See 5.3 for implementing the required text.

An example reads as follows:

[ISO 10303 required:10303_introduction]

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for retention and archiving.

[end required]

8.1.4 Computer interpretable listing annex

8.1.4.1 Documents that include EXPRESS

This annex shall provide electronic access to the list of short names provided in the annex "short names" and the EXPRESS specified in this part. This access is provided through the specification of URLs that identify the location of these files on the Internet. The EXPRESS file shall not contain any intervening prose; the EXPRESS listing for all schemas shall be found in one file. The listing shall not contain any comment delimiters of the kind "*)" and "(*" that separate the EXPRESS declarations from the main body of the prose. However, tail comments (those beginning with "--") may be included.

The text of this annex is an ISO 10303 required text. See 5.3 for implementing the required text.

An example reads as follows:

[ISO 10303 required:10303_annex-comp-int]

This annex references a listing of the EXPRESS entity names and corresponding short names as specified or referenced in this document. It also provides a listing of each EXPRESS schema specified in this document without comments or other explanatory text. These listings are available in computer-interpretable form in Table C.1 and can be found at the following URLs:

Short names:	http://standards.iso.org/iso/10303/tech/short_names/short-names.txt
EXPRESS:	http://standards.iso.org/iso/10303/smrl/v8/tech/smrlv8.zip

<insert table C.1– EXPRESS listings>

NOTE The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this document is normative.

[end required]

8.1.4.2 Documents that don't include EXPRESS

Please refer to the N Document guidelines or adapt the text as appropriate.

8.1.5 EXPRESS-G diagram annex

The EXPRESS-G diagrams describing the schema(s) defined in the part shall be included as a set of figures in this annex. Rules for formatting these diagrams are found in 6.8.

The text used to introduce the EXPRESS-G diagrams and to list the captions of each figure containing an EXPRESS-G diagram is a required text. See 5.3 for implementing the required text.

An example reads as follows:

[ISO 10303 required:10303_annex-expg-1]

The diagrams in this annex correspond to the EXPRESS schemas specified in this document. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in Annex D of ISO 10303-11.

[end mandatory]

[ISO 10303 required:10303_annex_expg-1]

Figure <X>.<n> — EXPRESS-G diagram of the <schema_name> (<x> of< y>)

[end required]

where <X> is the annex number, <n> is the diagram number, and <x>,<y> are the ranges of the related figures for one schema.

EXAMPLE From ISO 10303-42:2019

Figure D.1 — EXPRESS-G diagram of the geometry_schema (1 of 16);

8.2 Documentation of Integrated resources series (IR)

8.2.1 General

For the documentation of the integrated resources series of parts for ISO 10303 the following documents apply:

— Documents listed in 8.1;

— ISO/TC 184/SC 4 N1548 — Guidelines for the format and layout of SC4 standards using HTML.

Use of STEPmod (see 3.4.20) and STEPlib (see 3.4.21) by IR, AP and AM development teams is mandatory. See the STEPmod Tutorial (Eurostep, available at stepmod/help/STEPmod_Tutorial_v008.pdf) and the STEPlib SysML Guidelines (STEP Extended Architecture Team, 2019, available at http://cvs.boost-lab.net/sphinx/STEPlib_User_Guide/html/).

A set of integrated resources (IRs) (see 3.4.13) shall provide the specification of a representation of product information. Each IR comprises a set of descriptions, written in a formal data specification language, applicable to product data known as resource constructs. One set may be dependent on other sets for its definition. A single resource construct (See 3.4.19) may represent similar information for different applications.

The IRs in ISO 10303 are divided into two groups: generic resources (See 3.4.10) and application resources (See 3.4.6). The generic resources are independent of applications and may reference other resources. The application resources may reference other resources and may add other resource constructs for use by a group of similar applications. The IRs may reference product data descriptions written using EXPRESS from other International Standards.

Each part of ISO 10303 in the integrated resources series includes a clause for each schema that is documented in the part. See Clause 6 for requirements that apply to documenting EXPRESS schemas.

EXAMPLE The clauses 4 to 7 of the ISO 10303-42 are the following:

4 Geometry, which is the Geometry schema

5 Topology, which is the Topology schema

6 Geometric model, which is the Geometric model schema

7 Scan data 3d shape model, which is the Scan data 3d shape model schema

This section uses the following convention to delineate annexes that are to be included, they are enclosed in light grey box, the header gives the “mandatory/optional/conditional” status.

8.2.2 Content of an integrated resource

The contents of an integrated resource shall be composed of the elements listed in the *Table 1 — Overview of the major subdivisions of a document and their arrangement in the text* and at least of the following mandatory annexes:

[ISO 10303 mandatory]

A Short names of entities (Normative)

B Information object registration (Normative)

C Computer interpretable listings (Informative)

D EXPRESS-G diagrams (Informative)

[end mandatory]

Conditional informative annexes may be included. They shall be lettered sequentially. Such annexes may include the following:

[ISO 10303 conditional]

E Change history (Informative)

[end conditional]

Additional informative annexes may be included if they provide additional information that helps the reader to understand the schemas documented in the part. They shall be lettered sequentially. Such annexes may include the following:

[ISO 10303 optional]

F Technical discussions (Informative)

G Examples (Informative)

H Additional scope information that crosses multiple schemas or integrated resources parts.

[end optional]

8.2.3 Documentation of Introduction

The start of the introduction of each part of ISO 10303 that is a member of the integrated resources series is an ISO 10303 required text. See 5.3 for implementing the required text.

An example reads as follows:

[ISO 10303 required:10303_introduction]

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for retention and archiving.

[end required]

For the parts with more than one schema, use the following:

[ISO 10303 required:10303_IR_introduction-1]

Major subdivisions of this document are: <use list format to list the names of the schemas, or descriptions of groups of schemas>.
[end required]

This wording may be followed by one or more paragraphs that provide an overview of the schema or schemas, without stating requirements. If information from the scope clause is repeated, use the same wording as in the scope.

In an integrated resources part, include a schema level model that illustrates the schema(s) specified in the context of the integrated resources as a whole, introduced by the following wording:

[ISO 10303 required:10303_IR_introduction-2]

The relationships of the schemas in this document to other schemas that define the integrated resources of ISO 10303 are illustrated in Figure <figure number> using the EXPRESS-G notation. EXPRESS-G is defined in Annex D of ISO 10303-11.

The <list schemas from other parts depicted in the diagram> are specified in <list the parts>. The schemas illustrated in Figure <figure number> are components of the integrated resources.
[end required]

8.2.4 Documentation of scope

The scope clause of a part of ISO 10303 in the integrated resources series provides a clear explanation of the technical boundaries addressed by the schemas specified in the part. For an integrated generic resource part, the scope describes the domain of applicability. For an integrated application resource part, the scope describes the application area.

An example of the wording to introduce the scope clause of a part of ISO 10303 that is a member of the integrated resources series is given below, followed by the SC 4 wording (See 5.5.4 for general SC 4 requirements for the wording of the scope clause). See 5.3 for implementing the required text.

[ISO 10303 required:10303_IR_scope-1]

This document specifies the integrated generic resource constructs for

[end required]

Then, the following wording to introduce any further information that is within the scope of the part but does not fit into the first list is provided below:

[SC 4 required:SC4_Scope_scope-2]

The following is (are) within the scope of this document:

<list of in-scope elements>

[end required]

And finally, the following wording to introduce information that is outside the scope of the part:

[SC 4 required:SC4_Scope_scope-3]

The following is (are) outside the scope of this document:

<list of out-scope elements>

[end required]

The primary mechanism for defining the scope of each part in the integrated resources series is a simple text explanation of the information content of the part. At the minimum, an integrated resource part shall contain a narrative description of the scope of the information represented by the schema. It

should describe the application area being addressed and any boundaries, limits, or rules used to determine whether something is in scope.

8.2.5 Documentation of normative references

Each part of ISO 10303 that is related to the integrated resource constructs being documented shall be identified as a normative reference. A part of ISO 10303 containing a schema that is identified by a **REFERENCE FROM** EXPRESS statement in the integrated resource being documented is related and shall be listed as a normative reference.

8.3 Documentation of the application protocol series of parts for ISO 10303

For the documentation of the application protocol series of parts for ISO 10303 the following documents apply:

— Documents listed in 8.1;

— ISO/TC 184/SC 4 N1548 — *Guidelines for the format and layout of SC4 standards using HTML*;

— ISO/TC 184/SC 4 N1863 — *Guidelines for the content of application protocols that use application modules*;

— ISO/TC 184/SC 4 N2661 — *Guidelines for the development of mapping specifications*;

— ISO/TC 184/SC 4 N3422 — *STEP extended architecture high-level description*;

— ISO/TC 184/SC 4 N3425 — *STEP extended architecture publication design*.

Editors shall also take note of the following documents:

— ISO/TC 184/SC 4 N1337 (ISO/TC 184/SC 4 N1337, Application module development points within the application protocol development process) — *Application module development points within the application protocol development process.*

Use of STEPmod and STEPlib by IR, AP and AM development teams is mandatory. See the STEPmod Tutorial (Eurostep, available at stepmod/help/STEPmod_Tutorial_v008.pdf) and the STEPlib SysML Guidelines (STEP Extended Architecture Team, 2019, available at http://cvs.boost-lab.net/sphinx/STEPlib_User_Guide/html/).

8.4 Documentation of the application module series of parts of ISO 10303

For the documentation of the application module series of parts for ISO 10303 the following documents apply:

— Documents listed in 8.1;

— ISO/TC 184/SC 4 N1337 — *Application module development points within the application protocol development process*;

— ISO/TC 184/SC 4 N1548 — *Guidelines for the format and layout of SC4 standards using HTML*;

— ISO/TC 184/SC 4 N1685 — *Guidelines for the content of application modules*;

— ISO/TC 184/SC 4 N2661 — *Guidelines for the development of mapping specifications*.

Use of STEPmod and STEPlib by IR, AP and AM development teams is mandatory. See the STEPmod Tutorial (Eurostep, available at stepmod/help/STEPmod_Tutorial_v008.pdf) and the STEPlib SysML Guidelines (STEP Extended Architecture Team, 2019, available at http://cvs.boost-lab.net/sphinx/STEPlib_User_Guide/html/).

8.5 Documentation of the domain model series of parts for ISO 10303

For the documentation of the domain model series of parts for ISO 10303 the following documents apply:

— Documents listed in 8.1;

— ISO/TC 184/SC 4 N1337 — *Application module development points within the application protocol development process*;

— ISO/TC 184/SC 4 N1548 — *Guidelines for the format and layout of SC4 standards using HTML*;

— ISO/TC 184/SC 4 N3422 — *STEP extended architecture high-level description;*

— ISO/TC 184/SC 4 N3425 — *STEP extended architecture publication design*.

Use of STEPmod and STEPlib by IR, AP and AM development teams is mandatory. See the STEPmod Tutorial (Eurostep, available at stepmod/help/STEPmod_Tutorial_v008.pdf) and the STEPlib SysML Guidelines (STEP Extended Architecture Team, 2019, available at http://cvs.boost-lab.net/sphinx/STEPlib_User_Guide/html/).

9 ISO 13584 series: additional requirements for structure and content of parts

9.1 General

For all parts in the ISO 13584 series, the following documents provide applicable requirements:

— the ISO/IEC Directives, Part 2, 2018, *Principles and rules for the structure and drafting of ISO and IEC documents*;

— The ISO Simple template.

In addition, Clause 5 of this document (Requirements for the structure and content of parts of SC4 standards) applies.

9.2 Documentation of the Scope

The scope clause of each part of ISO 13584 shall clearly convey the scope of the part in terms that are understandable to users, library managers, and implementers with little or no PLIB experience, and shall be understandable to the PLIB community.

The scope clause of each part of ISO 13584 shall include an unordered list of in-scope items.

The scope clause of each part of ISO 13584 shall include an unordered list of out-of-scope items.

The first paragraph of the scope statement of each ISO 13584 view exchange protocol shall start with one of the required two paragraphs. See 5.3 for implementing the required text. Examples are provided below.

[ISO 13584 required:13584-scope-1a]

This document specifies a particular representation category, called <name of category>. This representation category captures the <essence of category>. This representation category may be associated with any of the items defined in a parts library. This document also defines how representations that belong to this representation category may be exchanged within a library exchange context by means of <means used to exchange category>.

[end required]

[SC 4 required:SC4_Scope_scope-2]

The following is (are) within the scope of this document:

<list of in-scope elements>

[end required]

[SC 4 required:SC4_Scope_scope-3]

The following is (are) outside the scope of this document:

<list of out-scope elements>

[end required]

[ISO 13584 required:13584-scope-1b]

This document specifies how representations that belong to the <name of a representation category> may be exchanged by means of <means used to exchange category>.

[end required]

[SC 4 required:SC4_Scope_scope-2]

The following is (are) within the scope of this document:

<list of in-scope elements>

[end required]

[SC 4 required:SC4_Scope_scope-3]

The following is (are) outside the scope of this document:

<list of out-scope elements>

[end required]

Each ISO 13584 view exchange protocol shall include required notes in its scope clause. See 5.3 for implementing the required text. An example is shown as follows:

[ISO 13584 required:13584-scope-2]

NOTE 1 The structure of a library delivery file is defined by a library integrated information model specified in one of the logical resource series parts of ISO 13584.

NOTE 2 The ISO13584_f_m_iim_schema, documented in ISO 13584-24, is a library integrated information model that defines the structure of a library delivery file. Such a library delivery file may contain instance values that reference the representation category and/or the library external files defined in this document.

[end required]

9.3 ISO 13584 series required text

For all the documents of the ISO 13584 series a specific wording is required.

This specific wording is defined and managed by the working group responsible for the series and is available in a shared repository.

The repository is located under [GIT ISO URL]/projects/ISOTC184SC4/repos/boilerplate.

Each writer shall ensure that the boilerplate text in an SC 4 standard conforms to the boilerplate text released in the repository.

A guide "How to use the required text" is at Annex B of this document.

10 ISO 15926 series: additional requirements for structure and content of parts

10.1 General

For all parts in the ISO 15926 series, the following documents provide applicable requirements:

— the ISO/IEC Directives, Part 2, 2018, *Principles and rules for the structure and drafting of ISO and IEC documents*;

— The ISO Simple template.

In addition, Clause 5 of this document (Requirements for the structure and content of parts of SC4 standards) applies.

10.2 Documentation of the introduction

The introduction of each part of ISO 15926 shall start with a required wording. See 5.3 for implementing the required text. An example is shown as follows:

[ISO 15926 required:15926-introduction]

ISO 15926 is an International Standard for the representation of process plant lifecycle information. This representation is specified by a generic, conceptual data model that is suitable as the basis for implementation in a shared database or data warehouse.
The data model is designed to be used in conjunction with reference data: standard instances that represent data common to a substantial number of experts from the process plant engineering supply chain.
The support for a specific life-cycle activity depends on the use of an appropriate selection of reference data in conjunction with an appropriate data model derived from the ISO 15926 data model ontology.

ISO 15926 is organized as a number of parts, each published separately. This document specifies:;

— ISO 15926 – 1: Overview and fundamental principles, published in 2004

— ISO 15926 – 2: *Data model,* published in 2003

— ISO/TS 15926 – 3: *Reference data for geometry and topology,* published in 2009

— ISO/TS 15926 – 4: *Initial reference data,* published in 2007

— ISO/TS 15926 – 6: *Methodology for the development and validation of reference data,* published in 2013

— ISO/TS 15926 – 7: *Implementation methods for the integration of distributed systems: Template methodology,* published in 2013

— ISO/TS 15926 – 8: Implementation methods for the integration of distributed systems: Web Ontology Language (OWL) implementation, published in 2013

— ISO/TS 15926 – 9: Implementation methods for the integration of distributed systems – Façade implementation, not yet in ballot

— ISO/TS 15926 – 10: Conformance testing, planned published in 2018

— ISO/TS 15926 – 11: Methodology for simplified industrial usage of reference data, published in 2015

— ISO/TS 15926 – 12: Life cycle integration ontology represented in OWL, planned published in 2018

— ISO 15926 – 13: Integrated asset planning life-cycle (ILAP), planned published in 2018

— ISO/TR 15926-14: ISO 15926-2 Data model adapted for OWL 2 Direct Semantics planned published in 2019

— ISO 15926-5 has been replaced by an annex to ISO TC184/SC4: Procedure for development and maintenance of reference data in database format.

[end required]

The introduction of each part of ISO 15926 shall also include the following information:

— a summary of the structure of the part;

— the target audience(s) for the part;

— any typographical or other conventions used in the part.

NOTE ISO 15926-1 has a slightly different introduction, as its introduction covers the whole of ISO 15926 as well as that specific part.

10.3 ISO 15926 series required text

For all the documents of the ISO 15926 series a specific wording is required.

This specific wording is defined and managed by the working group responsible for the series and is available in a shared repository.

The repository is located under [GIT ISO URL]/projects/ISOTC184SC4/repos/boilerplate.

Each writer shall ensure that the boilerplate text in an SC 4 standard conforms to the boilerplate text released in the repository.

A guide “How to use the required text” is at Annex B of this document.

11 ISO 18876 series: additional requirements for structure and content of parts

11.1 General

For all parts in the ISO 18876 series, the following documents provide applicable requirements:

— the ISO/IEC Directives, Part 2, 2018, *Principles and rules for the structure and drafting of ISO and IEC documents*;

— The ISO Simple template.

In addition, Clause 5 of this document (Requirements for the structure and content of parts of SC4 standards) applies.

11.2 Documentation of the introduction

The introduction of each part of ISO 18876 shall include the following information:

— an overview of ISO 18876;

— a summary of the structure of the part;

— the target audience(s) for the part;

— any typographical or other conventions used in the part.

NOTE ISO 18876-1 has a slightly different introduction, as its introduction covers the whole of ISO 18876 as well as that specific part.

Annex A
(informative)

Tutorial on ASN.1 identifiers

A.1 Introduction

SC 4 uses three separate terms to manage the various components of its standards. These terms are as follows:

- edition;
- version;
- release.

The term “edition” identifies a published document. The method of identifying the edition is by the year of publication. Thus, we refer to Part 21 as ISO 10303-21:1994. When a second edition of Part 21 is published, its identifier will differ by the year of its publication.

The term “version” identifies the normative content of a standard. For the initial edition of each part of an SC 4 standard, there is a one to one correspondence between the edition and the version. However, if technical corrigenda or amendments to the standard are published, the version of the technical content changes. In general, there is not a simple relationship between edition and version. The version is meant to identify the technical content to which conformance may be claimed.

For example, because the published version of Part 21 contained technical errors, it was necessary to issue a technical corrigendum to this part. The version of the original publication is 1; the version of effective standard, after applying the technical corrigendum is 2. Note that version 2 of Part 21 is documented in the two publications, the IS of Part 21, and the TC to Part 21, and not just a single document. Similarly, if an amendment to Part 21 is adopted, the version of the effective standard will be 3, and will be documented in the three publications taken together.

The version is defined as part of the object identifier defined in an annex of each part of each SC 4 standard. The value of that object identifier is described below.

The term “release” is used by SC 4 to manage the publications of groups of parts; historically, this term has been primarily applied to ISO 10303, although future planning for SC 4 standards includes releases that comprise parts of several standards. The release is not explicitly defined within any part of a standard. It is used solely for managing the development of STEP.

A.2 Object identifiers

An object identifier is a primitive data type defined in ASN.1, ISO/IEC 8824-1. The value identifies a node in a tree structure by providing a sequence of (positive) integers, each of which identifies a link in the tree. The notation used in SC 4 is the value notation defined in ISO/IEC 8824.

The syntax of this value notation is a sequence of node specifiers enclosed in braces (curly brackets) and separated by spaces. The syntax of each node specifier is one of the following:

— a number;

— a symbol;

— a symbol followed by a number in parentheses.

Whichever syntax is chosen, the resulting value must reduce to a sequence of integers. These choices are described below.

— A number. This syntax is self-identifying; the value of the node identifier is the value of the number. Any object identifier can optionally be written as a series of numbers. See the example below.

— A symbol. This syntax can be used only for the first or second node, and the only symbols that may be used are those defined in the annexes of ISO/IEC 8824-1. For our purposes, this restriction means that the first part of all object identifiers must be

```
{ iso standard }
```

which is equivalent to the object identifier

```
{ 1 0 }
```

— A symbol followed by a number in parenthesis. If this form is used, the value of the node is the value of the number. The symbol is a local variable that is automatically assigned the value of the number. Because there are no other uses for this symbol in the syntax, the only utility of this form is to give a human readable idea of the meaning of the node. Thus, we will use “version(1)” to indicate that we are dealing with the first version of something. We can equally refer to this node as “1”. Both forms evaluate to 1; the first form associates the semantics of “version” with this value.

The lexical syntax of terms in the object identifier is similar to that of EXPRESS, except that occurrences of underscore (_) shall be replaced by hyphen(-).

In the annexes, ISO/IEC 8824-1 defines the four topmost levels of all object identifiers. In particular, it defines the form

```
{ 1 0 n nn }
```

to be an object identifier that identifies an ISO standard number "n", part "nn". It then provides for the committee or subcommittee that wrote the standard to assign other nodes beneath this for identifying information objects related to this standard. Note that this identifier can also be written

```
{ iso standard n part(nn) }
```

which is closer to the form we normally use. In this notation, the defined symbol "iso" has the value 1, and the defined symbol "standard" has the value 0.

To repeat, ISO (in ISO/IEC 8824-1) defines the interpretation of the nodes at the first four positions of these object identifiers. The subcommittee writing the standard (in this case, SC 4) controls the semantics of nodes at lower positions.

SC 4 has decided to associate the fifth node with the version of the information object being identified. This decision means that a standard form of the object identifier for the (part of the) standard considered as a whole is

```
{ iso standard 10303 part(nn) version(v) }
```

SC 4 has adopted the convention that the sixth node identifies an object type, and the succeeding node or nodes identify a specific instance of that object type. The initial release defined only a single value for the object type; object(1) indicates that the object being identified is a schema. In the future, SC 4 may define other values for this node to cover other information objects such as entities, defined types, conformance classes, or parts libraries. Object values of 2 and greater are available for this purpose when the need arises. As of today, however, the only valid value of the sixth node is 1.

As corrigenda or amendments to the standards or new editions of the standards are published, the version number of the total content of the standard shall be increased by 1 to reflect the new content. It may be that in adopting a new edition of some standard, some information objects (schemas) within the standard will be unchanged from the previous versions. In this case the object identifier that identifies that information object (schema) should be the same as in the previous version of the standard, indicating that that particular item (schema) is unchanged.

Annex B
(informative)

How to use to the required text

For the ISO Simple template users and the developers of supporting tools the required text for each series is available under the GIT ISO repository.

SC 4 requires editors always to use boilerplate wording available at https://sd.iso.org/bitbucket-pilot/projects/ISOTC184SC4/repos/boilerplate

Please follow the readme.txt file available at https://sd.iso.org/bitbucket-pilot/projects/ISOTC184SC4/repos/boilerplate/browse/readme.txt

Any issues or questions regarding the boilerplate shall be reported to the SC 4 Quality Committee.

NOTE For the STEPmod and STEPlib development teams the required text is available within the tools.

Annex C
(informative)

Change History

The Table C. 1 below gives in the left column the section of the previous edition and in the right column the changes that have been made.

Table C. 1 — Changes between N2412 and N3485

N2412	N3485
Foreword	The references to 10303 has been removed.
Introduction	Modified as appropriate, the references to 10303 has been removed. Table 1 revision history has been moved to the inside cover. Main changes updated and moved to the Foreword. Target audience has been moved to the cover. Characteristic of a standard has been moved to the Clause 4 and renamed: Characteristics of an SC 4 standard Conventions has been removed. The convention for to delineate example of boilerplate text is given in 5.3 Required text.
1 Scope	Updated.
2 Normative References	Updated
3 Terms, definitions and abbreviations	
	3.1 Terms defined in ISO/IEC Directives, Part 2 Terms added
3.1 Terms defined in ISO 10303-1	3.2 Terms defined in ISO/DIS 10303-1:2020 Terms has been modified to follow the changes in the document.
3.2 Terms defined in ISO 10303-31	Removed as the terms are now defined in 10303-1
3 Terms defined in ISO 10303-202	Removed as the terms are now defined in 10303-1
3.4 Terms defined in ISO/IEC Directives, Part 2	Moved to 3.1
3.5 Other terms and definitions	Terms has been modified to follow the changes in the document.
3.6 Abbreviations	Abbreviations has been modified to follow the changes in the document.

Table C.1— Changes between N2412 and N3485 *(continued)*

4 Requirements for the structure and content of parts of SC 4 standards	Moved to 5. Table 2 replace by the table 2 of the ISO/IEC Directives, Part 2, 2018 and adapted to the SC 4 rules. Annexes modified to optional, Index modified to optional.
4.1 Preliminary elements	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Some topics are still described. See 5.4 Focus on some specific topics: Table of content; Patent rights;
4.1.5 Introduction	4.1.5 "hanging paragraph" Removed as the rules are defined in SO/IEC Directives, Part 2, 2018.
4.1.5.1 Wording for the introduction of parts of ISO 10303	The specific wording for Integrated resources series has been moved to 8.2 The specific wording has been removed. It will be managed as described in 5.3
4.1.5.2 Patent rights	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Some topics are still described. See 5.4 Focus on some specific topics: Table of content; Patent rights;
4.1.5.3 Translation issues	Removed. This will be managed as boilerplate. See 5.3
4.2 Normative elements	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Except 4.2.7.2 Some topics are still described. See 5.4 Focus on some specific topics: Scope The specific wording has been removed. It will be manage as described in 5.3
4.2.5.4 Symbols and abbreviations	Removed. ISO/IEC Directives, Part 2, 2018 apply. Some topics are still described. See 5.4 Focus on some specific topics: Symbols and abbreviated terms
4.2.7.2 Information object registration annex	Moved to 5.3.4 Use of ASN.1 Identifiers in SC 4 standards (optional).
4.3 Supplementary Elements	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Except 4.3.3.
4.3.3 Index	Moved to 5.3.3

Table C.1— Changes between N2412 and N3485 *(continued)*

4.4 Figures, tables, notes, examples, and footnote	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018.
4.5 Lists	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018.
4.6 References within the text	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Except 4.6.3
4.6.3 References to subdivisions of the text	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018, 22.4
4.7 Punctuation of words in a series	Moved to 5.4.8
4.8 Acceptable wording	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Except 4.8.6 and 4.8.7
4.8.6 Use of “i.e.”, “e.g.”, and “etc.	Moved to 5.4.9 and updated
4.8.7 Use of quotation marks	Moved to 8.1.1
4.8.8 Spelling	Moved to 5.4.11
4.9 Hyphenation	Moved to 5.4.11
4.10 Words to avoid	The two first paragraphs are defined by the ISO/IEC Directives, Part 2, 2018 then they have been removed. The rest of the subclause has been moved to 5.4.11
4.11 Frequently used words	Moved to 5.4.11
4.12 Representation of dates	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018.
5 Format and layout of SC 4 standards	Removed as the rules are defined in SO/IEC Directives, Part 2, 2018. Some topics are still described. See 5.4 Focus on some specific topics: Landscape;
6 EXPRESS presentation style	6.1 General added to avoid hanging paragraph The clause has been updated with the annexes B3, B4 and B5 of the document “Developer Manual, Express Engine 5.0.20 (Express Engine Team, available at http://exp-engine.sourceforge.net) ». The cross-references have been updated, some within the document and other to ISO/IEC Directives, Part 2, 2018. 6.2 Expressions has been added. 6.3 Statements has been added. The boilerplate text has been updated. It will be

	managed as described in 5.3

Table C.1— Changes between N2412 and N3485 *(continued)*

6.1 Layout rules	Updated
6.1.6 Layout of a schema	Updated
6.1.7 Layout of interface statements	Updated
6.1.8 Layout of a constant declaration	Updated
6.1.9 TYPE layout	Updated
6.1.10 Algorithm layout	Updated
6.1.11 Entity data type declaration layout	Updated
6.2 Style rules	Moved to 6.4 and updated
6.3.3 Definition of constraints	Moved to 6.6.3 and updated
6.5 EXPRESS-G diagram style	Moved to 6.7 and updated
7 Documentation of integrated resources series of parts of ISO 10303	Moved to 8.2 and updated
8 Documentation of the application interpreted construct series of parts of ISO 10303	Removed. Application interpreted construct series are no more in development.
9 Documentation of the application protocol series of parts for ISO 10303	Moved to 8.3 and updated
10 Documentation of abstract test suite series of parts of ISO 10303	Removed. Abstract test suite series are no more in development.
11 Documentation of the application module series of parts of ISO 10303	Moved to 8.4 and updated
12 Documentation of the parts of ISO 13584	Moved to 9 and updated
13 Documentation of the parts of ISO 15531	Removed as the series follows the SC 4 requirement, See Clause 5.
14 Documentation of the parts of ISO 15926	Moved to 10 and updated
15 Documentation of the parts of ISO 18629	Removed as the series follows the SC 4 requirement, See Clause 5.
16 Documentation of the parts of ISO 18876	Moved to 11 and updated
17 Documentation of standards that are produced jointly with other committees	Removed as the series follows the SC 4 requirement, See Clause 5.
18 Technical Corrigenda	Removed as the Technical Corrigenda is no more in use. Amendment follows the ISO/IEC Directives, Part 2, 2018.
Annex A (normative) Cover pages	Removed as the topic follows the SC 4 requirement, See Clause 5.
Annex B (normative) Use of ASN.1 Identifiers in SC 4 standards	Moved to 5.3.4 for the paragraphs related SC 4. Moved to Annex A for the "Tutorial"
Annex C (informative) Examples	Examples regarding EXPRESS have been moved to 6.9 and delete the other are they are already addressed by the Directives or by some specific SC

	4 documents.

Table C.1— Changes between N2412 and N3485 *(continued)*

Annex D (informative) Criteria for lexical definitions	Moved to 5.4.5
Annex E (informative) Checklists and approval procedures	Moved to 5.2
Annex F (informative) Extract from Guide to using the ISO technical programme	Removed as this is addressed by the ISO/IEC Directives, Part 1
Annex G (informative) Revision history	Is now Annex C
Bibliography	Updated as appropriate
Index	Removed

Bibliography

10303-41:2019. (n.d.). *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support.*

ISO 7498-2:1989, Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture. (1989). ISO.

STEP (Standard for the Exchange of Product Model Data) Resource Integration: Semantic & Syntactic Rules. (1991). ISO TC 184/SC 4 N80.

2, I. G. (n.d.). *Standardization and related activities — General vocabulary.*

ISO/TC 184/SC 4 N1110, SC 4 Quality manual. (2000).

ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation. (2015). ISO.

ALRED, G., BRUSAW, C., & OLIU, W. (2012). *The Handbook of Technical Writing, 10th edition.* New York: St. Martin's Press.

BARNARD FEENEY, A., & PALMER (eds), M. (1998). *Procedures for application interpretation (draft).* ISO/TC 184/SC 4/QC N079.

Eurostep. (available at stepmod/help/STEPmod_Tutorial_v008.pdf). *STEPmod Tutorial.*

Express Engine Team. (available at http://exp-engine.sourceforge.net). *Developer Manual, Express Engine 5.0.20.*

HURLEY, P. J. (1999). *A Concise Introduction to Logic.* Wadsworth.

ISO 10241-1:2011, Terminological entries in standards — Part 1: General requirements and examples of presentation. (n.d.).

ISO 8000-2:2018, Data quality — Part 2: Vocabulary. (n.d.).

ISO 999:1996. (n.d.). *Information and documentation — Guidelines for the content, organization and presentation of indexes.*

ISO/IEC/IEEE 31320-1:2012, Information technology — Modeling Languages — Part 1: Syntax and Semantics for IDEF0. (n.d.).

ISO/TC 184/SC 4 N1337, Application module development points within the application protocol development process. (n.d.). ISO/TC 184/SC 4.

ISO/TC 184/SC 4 N3263. (2018, available at https://isotc.iso.org/livelink/livelink/properties/20850151). *SC4 Handbook.*

N532:1997, I. 1. (n.d.). *Guidelines for application interpreted model development.*

STEP Extended Architecture Team. (2019, available at http://cvs.boost-lab.net/sphinx/STEPlib_User_Guide/html/). *STEPlib SysML Guidelines.*