

The Steel Construction Institute

CIMsteel Integration Standards Release 2: Second Edition

Volume 1 - Overview

CIS/2.1



The Steel Construction Institute

The Steel Construction Institute is an independent, technical, member-based organization, dedicated to the development and promotion of the effective use of steel in construction. Founded in 1986, the SCI now has over 600 corporate members in 37 countries. The SCI's research and development activities cover many aspects of steel construction including multi-storey construction, industrial buildings, light gauge steel framing systems, stainless steel, fire engineering, bridge and civil engineering, offshore and hazard engineering, structural analysis systems, environmental engineering and information technology. The results of these projects are fed back to SCI members via a comprehensive package of benefits. Membership is open to all organizations and individuals that are concerned with the use of steel in construction. Members include designers, contractors, suppliers, fabricators, academics and government departments in the United Kingdom, elsewhere in Europe and in countries around the world. The SCI is financed by subscriptions from its members, revenue from research contracts and consultancy services, publication sales and course fees.

A Membership Information Pack is available free on request from:

The Membership and Development Manager, The Steel Construction Institute, Silwood Park, Ascot, Berkshire, SL5 7QN, UK.

Telephone: +44 (0)1344 623345, Fax: +44 (0)1344 622944.



School of Civil Engineering

The School of Civil Engineering in the University of Leeds is the largest in the United Kingdom, and has excellent contacts with the construction industry and professional institutions. Its staff members have a wide range of experience in the practice of engineering, planning, architecture, design, construction and management. The School is committed to Quality Assurance of its teaching. Contacts with industry include an Advisory Committee of senior engineers and architects in practice, an industrial tutor scheme for undergraduates involvement of practising specialists in student's projects, and lectures by eminent engineers, architects and landscape architects. Industry also participates by giving sponsorships, prizes, providing vacation work and welcoming its graduates on completion of their degree. The emphasis on multi-discipline work, design projects, communication skills and teamwork seems to make Leeds graduates particularly useful.

The Computer Aided Engineering (CAE) group is a multi-disciplinary research group concerned with the application of Information Technologies to the overall engineering process. The group gained an international reputation through its innovative work applying the concepts of product modelling to achieve a more efficient information flow in the construction sector.

For details of current research or a prospectus, please contact:

The School Secretary, School of Civil Engineering, University of Leeds, Leeds, LS2 9JT, UK. Telephone: +44 (0)113 343 2248, Fax: +44 (0)113 343 2265.

CIMsteel Integration Standards

Release 2: Second Edition

Volume 1 – Overview

A J Crowley BEng, PhD

A S Watson BTech, PhD, CEng, MICE

Published by:

The Steel Construction Institute
Silwood Park
Ascot
Berkshire
SL5 7QN

Tel: 01344 623345
Fax: 01344 622944
URL: <http://www.steel-sci.org/>

In association with:

Computer Aided Engineering Group
School of Civil Engineering
The University of Leeds
Leeds
LS2 9JT

Tel: 0113 343 2282
Fax: 0113 343 2265
URL: <http://www.leeds.ac.uk/civil/>

This publication is derived from the deliverables of the CIMsteel Project

© 2000, 2003 The University of Leeds

© 2000, 2003 The Steel Construction Institute

The University of Leeds and the Steel Construction Institute wish to acknowledge the valuable contribution from other CIMsteel Collaborators, and from other parties, in the generation of this material.

Apart from any fair dealing for the purposes of research or private study or criticism or review, as permitted under the Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the UK Copyright Licensing Agency, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organisation outside the UK.

Enquiries concerning reproduction outside the terms stated here should be sent to the publishers, The Steel Construction Institute, at the address given on the title page.

Although care has been taken to ensure, to the best of our knowledge, that all data and information contained herein are accurate to the extent that they relate to either matters of fact or accepted practice or matters of opinion at the time of publication, The Steel Construction Institute, The University of Leeds, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data and/or information or any loss or damage arising from or related to their use.

Publications supplied to the Members of the Institute at a discount are not for resale by them.

Publication Number: SCI-P-265

ISBN 1 85942 099 0 (1st Edition)

British Library Cataloguing-in-Publication Data. (1st Edition)

A catalogue record for this book is available from the British Library. (1st Edition)

Front cover images have been reproduced by kind permission of Whitby Bird & Partners, Rowen Structures Ltd, Taywood Engineering Ltd, and Philip Quantrill (Structural Engineers) Ltd.

This publication is supported by a companion web site at <http://www.cis2.org/>

 [®] is a Registered Trademark

FOREWORD

2nd Edition

This publication introduces the Second Edition of the Second Release of the CIMsteel Integration Standards (CIS/2.1), a set of formal computing specifications that allow software vendors to make their engineering applications mutually compatible. It is the first of a series of SCI publications that make up the CIS/2 documentation. The CIS/2 documentation specifies the information that may be transferred between software applications, and how that information must be structured in a repository or data exchange file. This publication introduces the CIS, declares its scope, describes its origins in the CIMsteel Project, and gives an overview of CIS/2 documentation. It also describes some of the theory of 'Product Modelling' and the underpinning of CIS/2 by STEP (ISO 10303).

This Second Edition should be seen as a 'point release', rather than a whole 'new' release of the CIMsteel Integration Standards. Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

In the 3 years since the First Edition of CIS/2 was published, CIS/2 users have raised 103 issues. Most of these have been trivial, but the 25 'Major Technical' issues warranted the development of a second edition of CIS/2. Further, second editions of the STEP Generic Resources^[44, 45, 46, 47] used in LPM/5 were published in 2000; the revised constructs contained therein are incorporated into LPM/6.

The development of the second edition was coordinated by the CIS/2 International Technical Committee (ITC), the body responsible for improving and maintaining CIS/2 as an open standard. Chaired by Chuck Eastman, the ITC includes The Steel Construction Institute and Leeds University (the joint custodians of CIS/2), together with The American Institute of Steel Construction (AISC), Georgia Institute of Technology and the National Institute of Standards and Technology (NIST). The successful deployment of CIS/2 technology over the past years has been due to the efforts of software vendors who have developed commercially viable CIS/2 translators. Consequently, several of these companies are represented on the ITC, including: Bentley, CSC (UK) Ltd, CSI, Design Data, Fabtrol, Intergraph, RAM International, and Tekla.

The development of the Second Edition of CIS/2 was sponsored by the American Institute of Steel Construction (AISC).

Previous Releases of CIS

The first release of the CIMsteel Integration Standards (CIS/1) and the draft release of CIS/2 were deliverables of the Pan-European Eureka EU130 CIMsteel Project. Over a ten-year period, this extensive research and development project involved seventy organizations in ten countries, representing a wide cross section of the constructional steelwork industry, including designers, fabricators, software vendors, universities, research and trade organizations. In addition to the contributions of the individual collaborators, the CIMsteel project also received financial support from:

- Forschungsförderungsfonds für die gewerbliche Wirtschaft (FFF) in Austria
- Erhvervsfremme Styrelsen, Industrie-ministeriet in Denmark
- Ministère de l'Industrie, Département Communication et Commerce Extérieur in France
- Istituto Mobiliare Italiano in Italy
- Senter Den Haag in The Netherlands
- Department of Trade and Industry in the United Kingdom

The principal authors of this publication are Andrew Crowley of the SCI and Alastair Watson of the University of Leeds. The development of this publication was funded by the SCI, the University of Leeds and Corus Group plc (formerly British Steel plc). The authors would like to thank their (former) colleagues in the Computer Aided Engineering (CAE) Group within the School of Civil Engineering at the University of Leeds for their work on the CIMsteel Project; in particular Steven Bennett, Dimitris Christodoulakis, Paul Hirst, Nashett El-Kaddah, George Kamparis, Gareth Knowles, Peter Riley, Alan Smith, and Mike Ward. The authors would also like to acknowledge the valuable outputs of the CIMsteel 'Overall Design and Analysis Working Group'. This group was led by Richard Greiner of the Technical University of Graz, Austria, and included representatives from Leeds and Nottingham Universities, QSE, CSC, SCI, Ove Arup, Taywood from the UK, Ramboll (Denmark), TDV (Austria), CTICM (France), Sidercad (Italy) and FCSA (Finland).

The CIS are the result of considerable efforts by many persons representing CIMsteel collaborators, associate collaborators and other interested parties. The principal developers of the CIS were Leeds University & SCI (UK), CTICM & LGCH (France), TNO (Netherlands), Ramboll (Denmark), Italsiel & Sidercad (Italy). Inputs from beyond Europe, particularly from the USA and Japan are acknowledged. The issues raised by the international review team of AP230 have been invaluable in the development of CIS/2.

AISC endorsement of CIS/2

CIS/2 has been endorsed by the American Institute of Steel Construction (AISC) as the standard for the electronic exchange of structural steel project information for the North American structural steel design and construction industry^[26, 27].

During 1998, the Electronic Data Interchange (EDI) Review Team of the AISC evaluated data transfer standards with a view to adopting one. On December 7th 1998, their recommendation of CIS/2 was approved by the AISC Board of Directors as part of the AISC Business Plan for Standardizing the Electronic Exchange of Structural Steel Project Information. Phase I of the AISC Business Plan (now completed) included the public endorsement of CIS/2 as the standard for the electronic exchange of structural steel project information for the entire U.S. structural steel design and construction industry, as well as the recommendation that it be adopted as an international standard. Phase II of the AISC Business Plan includes several activities that will promote and support CIS/2 and its implementation.

AISC, headquartered in Chicago, is a not-for-profit organization established in 1921 to serve the structural steel industry in the United States. The organization's mission is to promote the use of structural steel through research activities, market development, education, codes and specifications, technical assistance, quality certification and standardization. AISC maintains the specification for the design of structural steel framing in the U.S. It has a long tradition of more than 75 years of providing assurance and service to the steel construction industry by providing reliable information.

For further details, please contact:

Director of Information Technology,
American Institute of Steel Construction,
One East Wacker Drive, Suite 3100, Chicago, IL 60601-2001, USA.
Telephone: +1 312 670 5413, Fax: +1 312 670 5403

CONTENTS

	Page No.
FOREWORD	III
2 nd Edition	iii
Previous Releases of CIS	iv
AISC endorsement of CIS/2	v
CONTENTS	VII
SUMMARY	IX
Résumé	x
Resumé	xi
Tiivistelmä	xii
Zusammenfassung	xiii
Sommario	xiv
1 INTRODUCTION	1
1.1 What are the CIMsteel Integration Standards?	1
1.2 What CIS/2 is not	2
1.3 The potential of CIS/2	2
1.4 Consequences for the software market	4
1.5 Visualizing CIS Data	5
1.6 The CIS/2 documentation	7
2 BACKGROUND TO THE STANDARDS	9
2.1 The evolution of data exchange	9
2.2 The CIMsteel Project	11
2.3 STEP	14
2.4 The IAI & the IFCs	16
2.5 XML	17
2.6 The evolution of CIS/2	19
2.7 The future of CIS/2	21
3 PRODUCT MODELLING	24
3.1 The Logical Product Model (LPM)	24
3.2 What is a Product Model?	24
3.3 Product Modelling Theory	25
4 THE SCOPE OF CIS/2	26
5 A CONCEPTUAL OVERVIEW OF LPM/6	27
5.1 The analysis model	27

5.2	The design model	28
5.3	The manufacturing (or physical) model	29
5.4	The Second Edition	30
6	THE DEVELOPMENT OF CIS/2	31
6.1	Drivers for the development of CIS/2 & LPM/5	31
6.2	Extending the scope & functionality of CIS/1	31
6.3	CIS/2 development as an Application Protocol	35
6.4	Drivers for the Development of the Second Edition	41
7	GUIDE TO THE CIS/2 DOCUMENTATION	42
7.1	Overview	42
7.2	Implementation Guide	42
7.3	The Information Requirements	42
7.4	The Logical Product Model (LPM/6)	44
7.5	Conformance Requirements	44
7.6	Worked Examples	45
8	REFERENCE SECTION	46
8.1	Bibliography	46
8.2	Abbreviations	52
8.3	Glossary	53

SUMMARY

This publication introduces the Second Edition of the Second Release of the CIMsteel Integration Standards (CIS), a set of formal computing specifications that allow software vendors to make their engineering applications mutually compatible. It is the first of a series of six SCI publications that make up the CIS/2 documentation.

The CIS/2 documentation specifies the information that may be transferred between software applications, and how that information must be structured in a repository or data exchange file. This publication introduces the CIS, declares its scope, describes its origins in the CIMsteel Project, and gives an overview of the CIS/2 documentation. It also describes some of the theory of 'Product Modelling' and the underpinning of CIS/2 by STEP (ISO 10303).

CIS/2 has been developed to facilitate a more integrated method of working through the sharing and management of information within and between companies involved in the planning, design, analysis and construction of steel framed buildings and similar structures. As this new method of working is likely to evolve gradually, the standards may be implemented in various degrees of complexity, from basic file exchange through to advanced implementation in a Database Management System (DBMS). The constructional steelwork industry is now in an enviable position to take advantage of the ways of working enabled by CIS/2.

CIS/2 substantially extended the engineering scope of CIS/1, and introduced advanced data management capabilities to enable data sharing. At its simplest, the CIS provides specifications and guidelines for the development and implementation of translators that enable the users of engineering software to export data from one application into another. However, CIS/2 also allows software vendors to support concurrent engineering via more direct mechanisms for information sharing and management. Thus, CIS/2 provides specifications and guidelines for the development and implementation of DBMSs built around the CIS and its related technology. Such a DBMS is known as a Product Model Repository (PMR).

Other documents in this series include:

- *Volume 2 - Implementation Guide*^[15]
- *Volume 3 - The Information Requirements*^[17]
- *Volume 4 - The Logical Product Model (LPM/6)*^[18]
- *Volume 5 - Conformance Requirements*^[19]
- *Volume 6 - Worked Examples*^[20]

1 INTRODUCTION

1.1 What are the CIMsteel Integration Standards?

The CIMsteel Integration Standards (CIS) are a set of formal computing specifications that allow software vendors to make their engineering applications mutually compatible. Potentially, they are applicable to any software application that involves the creation or use of engineering information concerning steel-framed buildings or similar structures. The Standards are intended to be used by software vendors to develop and implement translators for their applications.

At their simplest, CIS translators enable the users of the software to export engineering data in the form of a data exchange file from one application into another (Figure 1.1). Within data exchange file, the engineering information is structured in a logical and standardized way, irrespective of which vendor's software created it. Other CIS-compatible applications are able to read such a file and extract the information that is relevant to them.

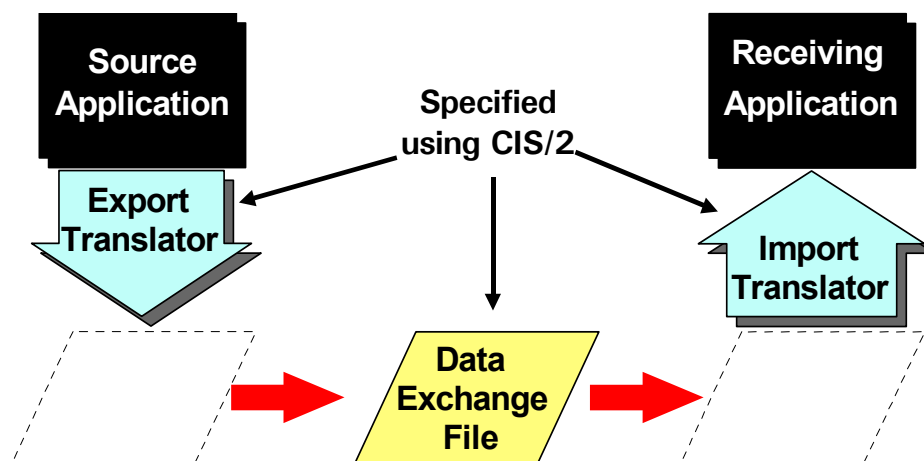


Figure 1.1 *Data exchange through CIS translators*

The first release of the CIS was published in September 1995, and was implemented by a number of software vendors in many countries, for a range of application types. This industrial use proved invaluable during the development of the second release (CIS/2), which was published in April 2000. Since then, the SCI and AISC have encouraged the implementation of CIS/2 through a series of biannual workshops and demonstrations and national conferences. As a consequence, many common software programs in Europe and the USA, now have commercially marketed CIS/2 translators.

This publication introduces the Second Edition of CIS/2, an extended and enhanced second-generation release of the CIS. CIS/2 has been developed to facilitate a more integrated method of working through the sharing and management of information within and between companies involved in the planning, design, analysis and construction of steel framed buildings and similar structures. As this new method of working is likely to evolve gradually, CIS/2 may be implemented in various degrees of complexity, from basic file exchange through to advanced implementation via Database Management Systems (DBMS). For further explanation, see the Glossary in Section 8.3.

1.2 What CIS/2 is not

CIS/2 is **not** a software application, or a computer program, or an IT system. CIS/2 cannot be installed or run on a computer. It is simply a specification that is documented as an International Standard.

CIMsteel does not provide a ‘magic bullet’. It will not automate the process of information creation, sharing, or management. CIS/2 is a ‘facilitator’, providing the standard against which software vendors can create the mechanisms for information sharing.

1.3 The potential of CIS/2

CIS/2 provides specifications and guidelines for the development and implementation of both basic data exchange translators and application interfaces for a Product Model Repository (PMR). A PMR is a DBMS built around the CIS and its related technology.

CIS/2 enables construction industry professionals to modify their working practices initially by improving internal communication of information inside a company, then by improving communication between participants in the building team. Ultimately, communication will grow to the ideal where computer applications can continuously communicate via physical files, databases, or direct procedure calls. The physical links will be provided by a combination of LANs, WANs, Extranets and the Internet. The information exchanged should be structured according to a standard specification controlled by the umbrella organizations of the participants involved. Project participants can then establish direct network communications, so that they can share information any time they want. An appropriate information management system should assure protection and consistency of information. By this stage, electronic information sharing will be the rule rather than the exception. This implies that the communication infrastructure of the construction industry will be adapted to electronic communication. New software will have to have interfaces based on appropriate standard specifications.

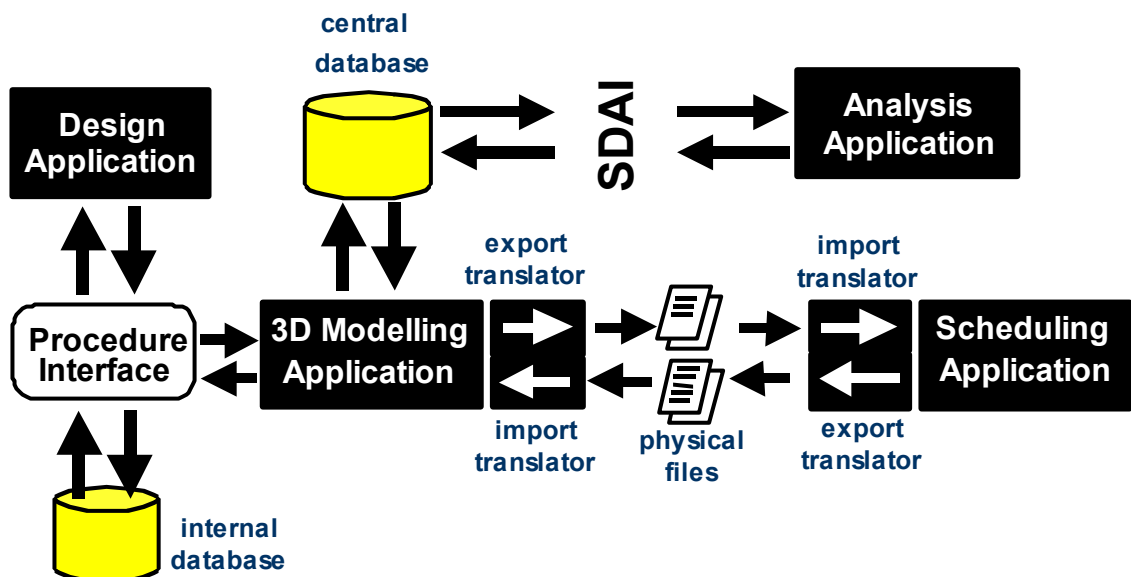


Figure 1.2 *Concurrent use of three forms of electronic communication*

A possible scenario is shown in Figure 1.2. Here, three forms of electronic communication are used concurrently, based around a 3D CAD / modelling system. The database of the modelling system is freely accessible, and thus acts as a central database. The system exports and imports physical files, and calls procedures. Upon request from the user, the procedural interface calls procedures that activate applications to manipulate information that has been captured by other CIS/2-compatible applications.

The details of how CIS/2 is implemented in the systems shown in Figure 1.2 are discussed in the *Implementation Guide*^[15].

A simplified view of the potential information flows facilitated by CIS/2 is shown in Figure 1.3.

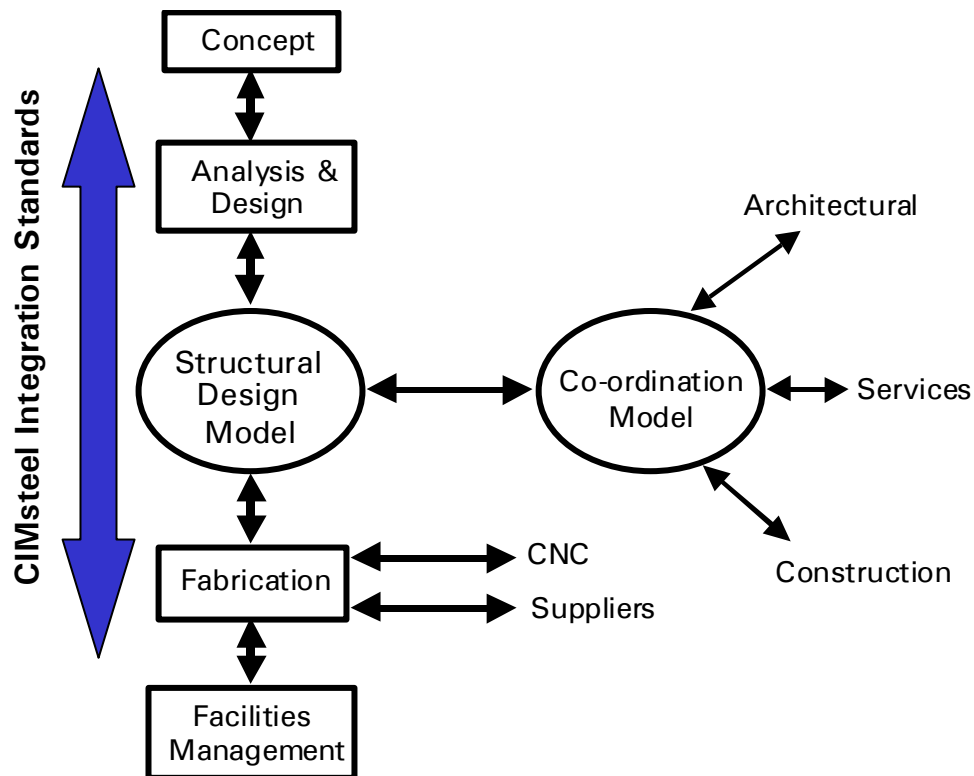


Figure 1.3 *Potential information flows*

It can be seen that the scope of CIS/2 lies within the steel supply chain. Thus, engineering end users of CIS/2-conformant systems will be able to share data among other users engaged in the following aspects of the steel supply chain:

- design
- detailing
- scheduling
- tendering
- ordering
- purchasing
- payment

This mechanism for electronic business enables improvements to be made in the efficiency of the design and construction of steel-framed buildings. Already, improvements in the flow of information between consulting engineers and fabricators have seen tender return periods in hours rather than weeks. Indeed, one firm of consulting engineers has shown a significant increase in their productivity^[62], while another demonstrated a 75% saving in man-hours by eliminating drawings.

The 'right first time' approach facilitated by electronic business adds value to the services provided by designers and it allows them to resist reductions in fees by providing a demonstrably better service without increasing their overall costs. The reduction in risk to all participants in the supply chain also gives better value to clients.

Case studies in the US^[14] have shown proven time savings of almost 50% - from preliminary design to the start of steel erection - by using 3D modelling and extensive EDI on a design-build project when compared to a traditional design-bid-build job based on 2-D drawings.

While practice varies across Europe, the majority of UK fabricators are still way ahead of their US counterparts when it comes to the exploitation of information and communications technology. Moreover, in our recent survey of UK practice, 3D modelling and data exchange appear to be the norm for UK fabricators, although they are still using predominately proprietary technology. Unfortunately, the same cannot be said of UK consulting engineers. This is one of the barriers to further exploitation of CIS/2 within and beyond the steel supply chain.

Although firms in the process plant sector have been using 3D models for years, the vast majority of practitioners in the building sector are still only scratching the surface of the power that modern 3D modelling tools possess. In the commercial buildings sector, the '3D modelling approach' requires a radical shift in workflow; not only between the fabricator and the design team, but also between the architect and the structural engineer.

With the transfer of 3-D models, communication is improved throughout the whole supply chain. Some estimates suggest that 50% of a project's time is spent on questions about incomplete information. Electronic sharing of 3-D models reduces requests for information, as well as the time and expense involved in tracking them and waiting for answers.

The substantial part of the work to be done in achieving a high level of electronic integration now concerns the resolution of the commercial and legal issues involved utilizing electronic links. The real 'paradigm shift' will only occur when the industry's professional practitioners are more willing to work in an integrated manner. However, the electronic sharing and management of information – as offered by CIS/2 – brings immediate benefits.

1.4 Consequences for the software market

One of the longer-term consequences of the industrial deployment of CIS/2 is that a more competitive, open and wider market for specialist software will emerge, which will drive down software prices, while the quality of the software is both measurable and testable. The more people that demand CIS translators for their applications, the more benefit will accrue to the wider industry. The engineering end-user will benefit from cheaper and better quality software, while the vendor benefits from a much larger market. Their

encouragement should also hasten the implementation of the more advanced CIS/2-conformant systems by the software vendors. This would enable improvements to be made in the business processes of the construction industry. The tools are there, the next step is to convince the members of the building team to use them.

1.5 Visualizing CIS Data

Until recently, CIS/2 data could only be visualized within proprietary software packages that have implemented the standard. It was felt that collaboration would be greatly improved if a software-independent method of viewing CIS/2 files were available, so that all project participants could visualize and verify the structural steel information for a project. This would provide CIS/2 users with capabilities similar to those available for many CAD data formats, such as DXF, DWF, and IGES, which have commercial and freeware software solutions to visualize the information in those files, independent of the software package that generated them.

1.5.1 VRML

The Virtual Reality Modeling Language^[56] (VRML) is a 3D scene description language and file format that is used for visualizing 3D content on the Internet. Research at the National Institute of Standards and Technology has shown that VRML can be effectively used as a method for viewing CIS/2 files.

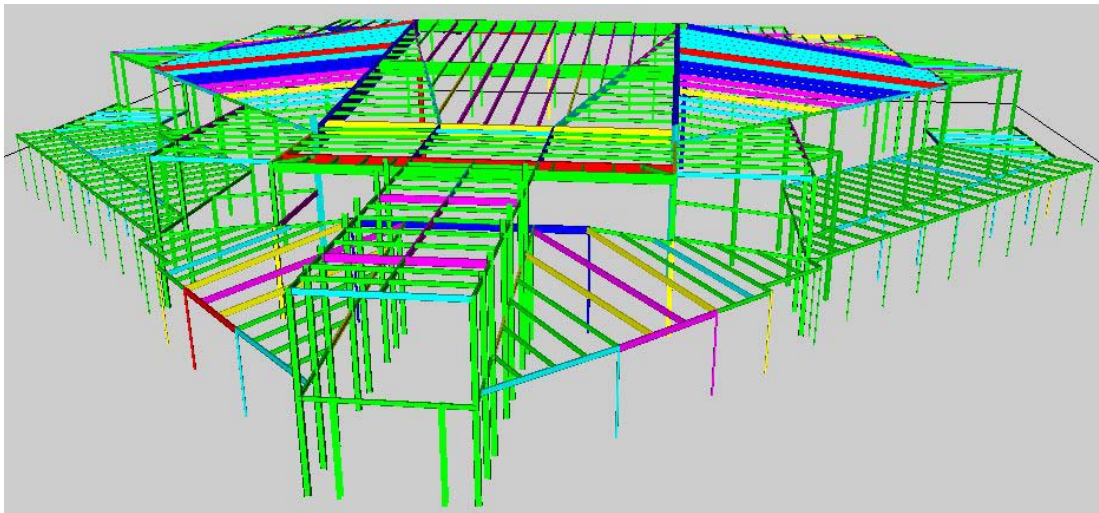


Figure 1.4: *VRML model generated from a CIS/2 file*

VRML files can be viewed in Internet Explorer or Netscape with freely available plugins, such as Cosmo Player, Cortona, and Blaxxun Contact. A VRML file describes the scene graph of the 3D information to be displayed. Nodes in the scene graph include geometric data, appearance, transformations for position and orientation, lighting, sensors, and many others. Geometric data includes primitives such as spheres, boxes, cones, cylinders, and generalized sets of polygon.

The real power of VRML is its extensibility. The prototype mechanism (protos) and scripting capabilities facilitate the creation of new application-specific nodes. VRML protos have been created at NIST that map directly to many of the CIS/2 entities necessary to describe the visual appearance of a steel structure. Those entities include: elements, parts, section profiles, holes, joint systems (bolts and welds), and features

(notches, chamfers, skewed ends, and cutting planes). A simple translator program, developed at NIST, reads a CIS/2 file and generates a complete VRML file. Figure 1.4 and Figure 1.5 show the resulting VRML models produced from CIS/2 files.

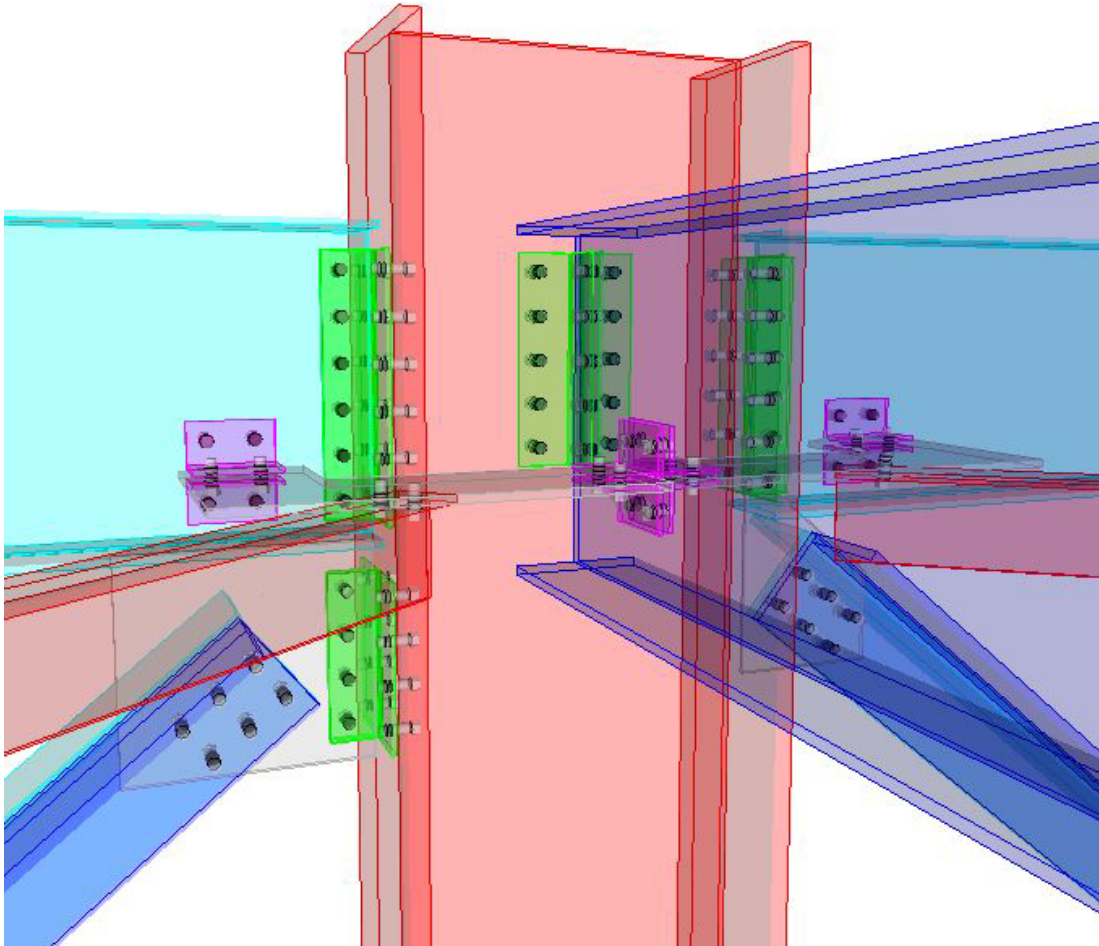


Figure 1.5: *Part of a VRML model generated from CIS/2 file*

1.5.2 Benefits of the VRML Viewer

By generating VRML models, software vendors have been able to verify their CIS/2 export capabilities. The VRML models provide a quick visual check that a steel structure looks like what is expected. Indeed, several problems with some CIS/2 implementations were readily identified, and reported to the software vendors.

Being able to display CIS/2 files as VRML models requires nothing more than a web browser and freely available VRML plugin. A subcontractor who does not have access to proprietary software used in a steel construction project can now access the same 3D model information as the other project members.

Displaying VRML files does not have to be limited to desktop computers. Wearable computers and even handheld PocketPCs that can be brought to a construction site can show VRML files. With increasing broadband Internet access speeds, wireless connections, and faster graphics processing, VRML models can be displayed whenever and wherever they are needed.

The VRML display of a CIS/2 file could also serve as a 3D interface to a project management information system. In addition to a traditional document-driven text-based 2D interface, an intuitive, easily navigated 3D visual interface can be made an integral part of the information flow for steel structure construction projects. Recent work has added facilities that allow users to navigate between assembly drawings, piece drawings, CNC data, and materials lists via a 3D virtual model; integrating several aspects of project information via CIS/2.

1.6 The CIS/2 documentation

The documentation for CIS/2 is published in six volumes as shown in Table 1.1. Section 7 of this publication provides a short guide to the six volumes of the documentation for CIS/2.

The role of the CIS/2 documents is to specify the type of information that may be transferred between applications programs, and how that information must be structured in a repository or data exchange file.

The structure of the CIS/2 documentation reflects the way in which CIS/2 was developed. As explained later in Section 2.3, the development path of CIS/2 parallels the development of a STEP Application Protocol (AP230).

Table 1.1: The CIS/2 documentation

Publication Reference	Volume	Title
SCI-P-265	Volume 1	Overview <i>This document</i> A concise introduction to CIS/2 its scope, its origins in the CIMsteel Project, and an overview of each volume of the CIS/2 documentation. It also describes some of the theory of 'Product Modelling' and the underpinning of CIS/2 by STEP (ISO 10303).
SCI-P-266 ^[15]	Volume 2	Implementation Guide A technical guide for software vendors wishing to implement CIS/2, covering some of the technology used in the development and implementation of CIS/2
SCI-P-267 ^[17]	Volume 3	The Information Requirements The information requirements are specified as a set of <i>Units of Functionality, Application Objects, and Application Assertions</i> , and are graphically represented in the <i>Application Reference Model (ARM)</i> . The information requirements correspond to activities that have been identified as being within the scope of CIS/2 as defined with the <i>Application Activity Model (AAM)</i> . The AAM shows the activities, and the data flows between these activities, during the production of a steel framed building complex.
SCI-P-268 ^[18]	Volume 4	The Logical Product Model (LPM/6) A short form, a long form, and a graphical presentation (in EXPRESS-G) of the CIS/2 implementation schema. The 'short form' provides illustrated textual descriptions of the entities, with their attributes and formal propositions (rules & constraints), together with references to STEP resources. It does not repeat the definitions of those STEP entities that have been used from the Generic Resources. The 'long form' presents LPM/6 as a single EXPRESS schema with all the inter-schema references resolved.
SCI-P-269 ^[19]	Volume 5	Conformance Requirements Specifies what software vendors are required to do to make their applications 'CIS/2-compatible'. It tabulates the <i>Conformance Classes</i> and details the requirements of the formal <i>Conformance Testing</i> procedures, which test whether an application conforms to the CIS/2 specifications. Volume 5 includes a textual description of how CIS/2 deals with national conventions and variations in the referencing of standard and manufacturers' product items and unit systems. It also includes a number of abstract and specific test cases that will be used for testing implementations claiming conformance to the CIS/2 specifications.
SCI-P-270 ^[20]	Volume 6	Worked Examples Illustrated descriptions and usage scenarios for examples of CIS/2-compatible applications

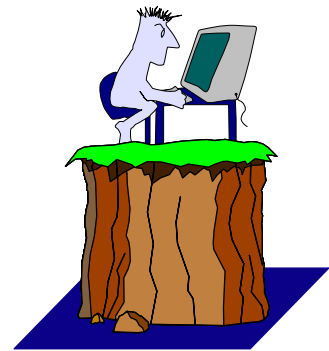
2 BACKGROUND TO THE STANDARDS

2.1 The evolution of data exchange

The construction industry has been slow to exploit the full potential of information technology. Many of the service industries, such as banking and retail sales, have evolved business and operational practices to the point where information technology is now central to their success. In such industries, the importance of representing information in digital form was recognised many years ago. However, the structure of that data is relatively simple, when compared to the information requirements of the construction industry. The focus of the construction industry is on very complex one-off projects, which often result in a unique set of project partners. This coupled with the historic divide between design, construction and maintenance, results in repetition of information at many sites, and the continual requirement to exchange complex information between numerous disciplines.

2.1.1 Islands of automation

Computer applications have been used throughout the construction industry for a number of years. However, until recently, those applications have been used to support existing processes. That is, they were focused on the generation of the same communication media as before, such as drawings, schedules, and specifications. Thus, even though engineering information is being held in digital form, it is (typically) being conveyed using traditional (paper) representations, which have to be interpreted by humans before the information could be used in other applications, thereby creating 'islands of automation'.



It can also be seen that poorly implemented IT strategies are duplicating paperwork, rather than reducing or eliminating it. This is mainly because the systems used throughout the construction industry evolved as a result of the technology of the day (i.e. pre-IT).

2.1.2 The Need for Information Exchange

There has never been a single application, or even a suite of software applications, capable of performing every task required during the design and construction of building projects. It is also inconceivable that a software vendor will ever provide all parties with a system of such complexity and flexibility. Even if they could 'please all of the people all of the time', it is unlikely that they would make a profit doing so. With this in mind, a modular approach to software development is emerging, such that each vendor specializes in what they perceive to be their greatest strengths, and their strongest markets. There is, therefore, a perceived need to exchange information between each of these activities and the supporting software modules.

When using *native* links, each application requires a translator for each and every application involved in the communication, as shown in Figure 2.1. Where more than a handful of modules are involved, one-to-one translators are expensive to maintain in the long term. Thus, there is a clear demand for an open-systems approach, via *neutral* files in accordance with an agreed international standard.

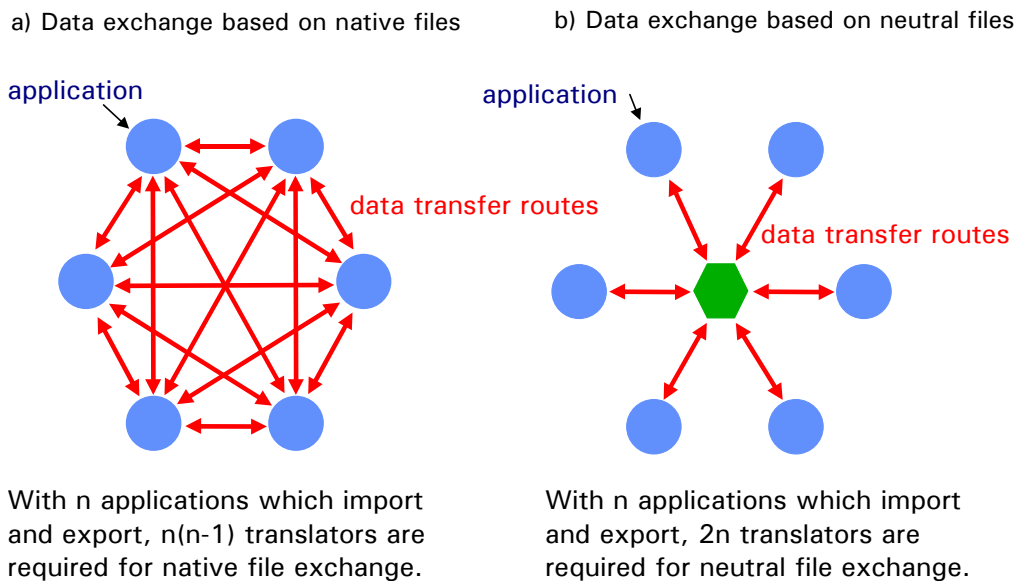


Figure 2.1 Data exchange using native and neutral file approaches

2.1.3 Native Data Exchange File Formats

The ‘islands of automation’ were first bridged by proprietary data exchange files using a native format. In the seventies and eighties, Computer Aided Draughting (CAD) systems were considered to be the central points for integration. Indeed, CAD was adopted by a number of professional, contracting, and supply organizations to improve the accuracy, quality and cost effectiveness of drawing production.

Models for the exchange of shape information grew more and more sophisticated, from wire frames and surface models to solid models (e.g. CSG and B-rep). Typically, these systems used a native (proprietary) file format. Software vendors then provided interfaces that allowed data to be exchanged directly between specific applications. The most widely used example in construction was the data exchange file format (DXF) of AutoCAD^[2]. This allowed drawings and related data to be passed in digital form between applications with appropriate DXF interfaces. Compared with traditional paper documents, many advantages flowed from such digital data exchange, the major one being the reduction of errors in interpreting the supplied representation.

DXF came to be regarded as the *de facto* drawing exchange standard, reflecting AutoCAD’s dominance of the CAD market in construction. One factor in the commercial success of AutoCAD was the number of applications that were developed using AutoCAD as a base. These ‘add-ons’ usually incorporated DXF translators into their systems allowing them to exchange data easily with similar CAD systems.

2.1.4 Neutral File Formats

Native file formats - such as DXF - were established by particular CAD vendors and remained under their control. Another drawback was the fact that as more applications became involved in data exchange, the more specialist translators were needed. One way of reducing the number of translators required is to use a ‘neutral’ format. Neutral file formats standardize the exchange of data in a non-proprietary way, aiming for universal application. For neutral file exchange, the means of communication is not fixed; i.e. the

file may exist on any storage medium (disk, tape, CD) and may be exchanged either via a network (local, wide, or global) or by simpler means (e.g. a floppy disk sent by post).

The most significant of these 'neutral standards' was the Initial Graphics Exchange Specification (IGES)^[68]. This was developed to support the data exchange requirements that users of CAD/CAM systems had identified in the 1970s. Version 1.0 of IGES was published in 1980, and covered graphical information with some annotation. Enhanced and more complex versions soon emerged to include finite element modelling.

The major problem with IGES was that CAD vendors were selective in their implementation of the standard, with the result that many translators did not support all the entities defined in a particular version of the standard^[75]. This, combined with the technical difficulty of writing good translators, led to incompatibilities between translators.

The neutral file concept established by IGES led to the evolution of several other data exchange standards; each targeted on the needs of a specific group of CAD/CAM users. In each case, the objective was to make the exchange process more efficient and reliable, and to maximize the ability of the developed data exchange file format to represent particular classes of engineering information.

Product Data Definition Interface (PDDI)^[9] was a US Airforce project of the mid-1980s, which built on IGES to demonstrate a computer-based product definition interface between design and manufacturing for airframes. A similar and parallel project by Aerospatiale in France led to the Standard for Exchange and Transfer (SET)^[1]. Concern for the efficient introduction of CAD/CAM into the German automotive industry led to a project in 1982 that created the VDAFS standard^[23] for the transfer of free-form car body surfaces.

While considerable technical progress was made by these various projects, the result was a proliferation of data exchange formats. It was recognized that a solution lay in a single second generation neutral file standard which would provide a unified framework for data exchange by all sectors of engineering^[75].

2.2 The CIMsteel Project

The CIMsteel Integration Standards is an outcome of the Eureka EU130 CIMsteel Project. Over its ten-year life, this extensive research and development project involved seventy organizations in ten countries, representing a wide cross section of the constructional steelwork industry, including designers, fabricators, software vendors, universities, research and trade organizations.

The focus of the CIMsteel Project was on building-type steel-framed structures, and the overall objective was:



“To improve the efficiency and effectiveness of the European Constructional Steelwork Industry both through harmonisation of design codes and specifications, and through the introduction of Computer Integrated Manufacturing techniques for the design, analysis, detailing, scheduling, fabrication, erection and management functions.”

The CIMsteel Project was concerned with facilitating a more integrated pattern of working within the industry through the adoption of information technology. Its objective was to turn a traditionally craft-based industry, made up of many disparate companies, into a state-of-the-art integrated manufacturing industry, by developing methods for ‘Computer Integrated Manufacturing’ (CIM) to streamline the business processes of the constructional steelwork industry.

The Information Technology (IT) aspects, as represented by the CIMsteel Integration Standards (CIS), were only one of many parts of the total project, accounting for approximately 20% of the project’s costs. Work on the CIS was led by Leeds University. The CIMsteel project formally ended in March 1998, at which time administrative responsibility for its deliverables (including the CIS) was vested in The Steel Construction Institute.

2.2.1 Computer Integrated Construction?

The CIM philosophy views manufacturing as an information-processing industry. In order to apply CIM techniques to the construction industry, one must view construction in a similar way. The basic idea of Computer Integrated Construction (CIC) is to facilitate the communication of data, knowledge and design solutions between project participants. Early development efforts focussed primarily on technical issues, such as the data structure of the constructed product and production processes. Later researchers have seen CIC as the key to a wider revolution. Protagonists of this ‘revolution’ saw that every process in the construction life-cycle is influenced by IT. They suggested that as it became more cost-effective to create and manipulate engineering information in digital form, the construction industry would face growing pressure to revise working methods so that they are more suited to the management of digital information and the related computer systems.

2.2.2 The culture change

A more significant legacy of the CIMsteel Project is the way in which it initiated the change in the business practices of the constructional steelwork industry. In part, this was facilitated by the industrial deployment of the CIMsteel Integration Standards (CIS); but the presence of technology, in and of itself, cannot wholly transform an industry. The real ‘paradigm shift’ can only occur if the industry’s professional practitioners are willing to work in a more integrated fashion. It was the adoption of the ‘CIMsteel philosophy’ rather than any individual piece of technology that brought about the significant improvements in productivity and profitability seen by the companies involved in the CIMsteel Case Studies^[10].

The substantial part of the work to be done in achieving a high level of electronic integration concerns the resolution of the commercial and legal issues involved in utilizing electronic links. Indeed, it can be seen that overcoming the barriers to the commercial implementation of integrated systems lies more in the domain of the management scientist, than the computer scientist or software specialist^[22].

However, the electronic sharing and management of information brings immediate benefits and it is important that there should be an agreed open and internationally-visible standard for sharing and management of electronic data.

2.2.3 IT and the business of change

The CIMsteel project saw that the transfer of digital information between different organizations was technically possible, and thus developed and promoted standard product descriptions that could be easily communicated in digital form. It also saw that the transfer of digital 'product models' created a new set of opportunities and challenges, but that changes in business practices would be needed throughout the industry if companies were to take full advantage of the new technology. Different levels of IT exploitation can support or induce different levels of business transformation to enhance the competitive capability of an organization. CIMsteel identified five such levels:

- i. **Localized exploitation:** introducing technology, such as a new software product, into an existing organization.
- ii. **Internal integration:** introducing digital information exchange or communication between different parts of a company, without affecting the existing company organization.
- iii. **Business redefinition:** re-organizing a company and its way of doing business to maximise the advantages offered by new information technology.
- iv. **Redesigned external links:** redefining links with other companies to enable digital transfer of information that was previously exchanged by traditional methods such as postal services.
- v. **Redefinition of the industrial sector:** totally altering the way in which a sector operates influencing the long term and contractual relationships between companies.

The 'open systems' approach that the CIMsteel Project pursued enables the industry to evolve towards information-centred working. The aim of such an approach is to provide flexibility through the adoption of standards for the sharing of engineering information. A central goal of the CIMsteel Project was to establish an internationally applicable open standard for the digital exchange and sharing of engineering information through the whole life cycle of a steel framework. As discussed later, this goal was also pursued through the STEP (Standard for the Exchange of Product data) initiative of the International Standardization Organization (ISO).

2.2.4 The industrial vision

The industrial vision of the CIMsteel Project was for the steelwork sector as a whole to be centred on the use of digital engineering information through the adoption of globally applicable open standards for information management, based on the concept of 'Product Modelling'. The CIS provides a means for this vision to be realized progressively, being an outcome of long-term industrially driven research, proven through demonstrators and industrial deployment.

The CIMsteel Project promoted the industrial deployment of CIS compatible software and introduced the concept of standardized product data exchange into the construction sector. The existence of a single global information standard for steelwork enables steelwork-related data to be transferred in digital form between heterogeneous engineering software, both within and between companies. This strategy is now yielding enhanced industrial efficiency through error reduction and shorter project time-scales.

The First Edition of CIS/2 was an enhanced second-generation specification that substantially extended the engineering scope of CIS/1, and introduced advanced data management capabilities to enable data sharing. The design of CIS/2 allows for the progressive realization of these improvements, providing professionals in the construction industry with open standards that will carry them beyond simple data exchange.

2.3 STEP

From the outset, it was always intended that the CIMsteel Project would make maximum use of internationally recognised standards. Of these, the most relevant has been ISO 10303, STEP^[38]. Through STEP, the International Standardization Organization (ISO) is providing an expanding series of complementary International Standards for sharing information across all sectors of engineering based on product modelling. Release One of STEP was published in 1994, since then it has continued to develop.

2.3.1 What is STEP?

STEP – the Standard for Exchange of Product Model Data – is a series of standards currently under development within the International Standardization Organization^[38]. Launched in 1983, ISO/STEP is a very ambitious long-term project to develop co-ordinated data exchange and information sharing standards that span all sectors of engineering, based on Product Modelling concepts.

The development of STEP is governed by Sub-committee 4 (SC4 - Industrial Data and Global Manufacturing Programming Languages) of ISO Technical Committee 184 (TC184 - Industrial Automation Systems and Integration), and has the designation ISO 10303. The formal title of STEP is ‘Industrial automation systems and integration - Product data representation and exchange’.

2.3.2 The STEP vision

The developers of STEP envisaged that engineering information would be transferred between application software (and thus between the users of that software) by the use of data exchange files and import/export translators. It also envisaged that information would be shared more intimately among groups of applications linked to a STEP compatible database. Whether the applications were to be ‘interfaced’ or ‘integrated’, STEP sought to provide a Product Model based standard, plus vendor-independent technology for implementing the data sharing.

The use of formal Product Model based specifications (and related implementation technology) ensures that computers can communicate using the same syntax. The procedures of STEP ensure that the meanings (or semantics) are also unambiguously defined. STEP Application Protocols provide specifications against which software vendors can implement translators and end users can successfully deploy data exchange. Related STEP standards such as SDAI (Standard Data Access Interface) provide a basis for shared access to STEP data.

2.3.3 The use of STEP in CIS/2

The key to the success of the CIS is its use of STEP and Product Modelling technology. It is, therefore, not just another proprietary data exchange format. The use of an open systems approach, incorporating Product Models and ISO STEP technology, brings a number of advantages to the users of CIS-compatible applications, as outlined below:

- The underlying logical data models are defined in the EXPRESS language^[39] using the architecture and resources^[44, 45, 46, 47, 48] of STEP. This means that the data models have been through a long and thorough formal development procedure¹.
- Indirect data exchange takes place via physical files whose format is specified by STEP Part 21^[40]. Since the data is held in a plain ASCII text file, it is computer interpretable regardless of platform, as well as being (reasonably) human-readable.
- Direct data exchange takes place via 'STEP physical models' by means of an Interface Definition Language called the Standard Data Access Interface (SDAI) defined in STEP Part 22^[41].

The role of EXPRESS is to define information models that may include constraints (defined by EXPRESS rules) and derived information (defined by EXPRESS functions). Such models are parsed to determine their semantic content, and a corresponding definition of the required STEP implementable form generated. The implementable form may be either a STEP data exchange file conforming to STEP Part 21 or a STEP physical model held in an SDAI repository such as that found in a Database Management System.

Part 21 specifies the format of an exchange structure. Thus, when an EXPRESS schema is interpreted using Part 21, it describes objects for which a corresponding exchange structure exists. Taken on its own, a STEP physical file has no understanding of the EXPRESS rules and functions, as Part 21 does not require that the information held in a STEP physical file complies with the constraints defined by the EXPRESS schema. Thus, it can be seen that a STEP physical file is no more than a carrier of basic data. That data need not satisfy rules defined in the original schema, and any derived information is not available².

In marked contrast, SDAI adds significant value to the EXPRESS schema. SDAI effectively makes EXPRESS a generic object-oriented language, and turns every valid EXPRESS schema into a specification of an object library. Standard functions are provided to access data instances in the repository, to evaluate derived information, and to check the validity of instances.

2.3.4 CIS/2 and Application Protocols

The STEP mechanism for data exchange is described as an Application Protocol (AP), which is based upon the STEP Integrated Resources (IRs). The IRs contain generic data structures which are then specialized (or interpreted) to form data structures applicable to specific domains. Data sharing within a domain takes place on basis of Conformance Classes (CCs), which are implementable and testable subsets of the AP.

In many ways, CIS/2 was developed as if it were a STEP AP. This was due to the concurrent development of STEP AP230, the formal AP for structural steelwork. However, because the scope and functionality of the underlying product model (the LPM)

¹ Second editions of the STEP Generic Resources^[44, 45, 46, 47] used in LPM/5 were published in 2000; this was one of the issues that prompted the development of a second edition of CIS/2. The revised constructs contained in these new resources are incorporated into LPM/6.

² Although the specification provided by ISO 10303-21: 1994^[40] does not require the data held by a STEP Part 21 file to comply with all aspects of the governing EXPRESS schema, CIS/2 requires that STEP Part 21 files created by CIS/2 conformant systems contain valid data in accordance with LPM/6. See the rules defined in the *Logical Product Model*^[18] and the *Conformance Requirements*^[19]

is much greater than that of any existing AP, it is structured differently from these APs. Any future AP development based on CIS/2 is likely to produce a family of APs rather than just one AP. Interoperability would still be possible, as these APs would all belong to the same family of STEP APs and use the same STEP architecture and resources. That is, an application conforming to one of these APs should be able to share data with another application that conforms to another member of the ‘steelwork family’ of STEP APs.

The ‘Working Draft’ of AP230^[55], which was based upon CIS/1, was submitted to ISO in 1996 for review by members of the international STEP community. This contained all the ‘normative’ and ‘informative’ sections that are required for a WD version of an AP. It did not include an Application Interpreted Model (AIM) because it had not been through the ‘interpretation process’ – a process that only starts once the WD is agreed. The purpose of the WD was to identify the scope and the information requirements of the AP.

2.4 The IAI & the IFCs

The International Alliance for Interoperability (IAI) began with a mission to define, promote and publish a specification for sharing data throughout the project life cycle, across disciplines and technical applications within the domains of Architectural, Engineering, Construction (AEC) and Facilities Management (FM). Its member organizations intended to specify how the ‘things’ that could occur in a building (such as doors, windows, and walls) should be represented electronically. Each specification (called a ‘class’) is used to describe a range of things that have common characteristics. The classes defined by the IAI are termed ‘Industry Foundation Classes’ or IFCs.

The IAI developed a common view of the data that can be shared by AEC/FM and software development professionals^[28, 29, 30, 31, 32, 33, 34]. This view is referred to as the ‘IFC Object Model’^[28]. By starting with a very general view of the AEC/FM industry, the IAI has defined an overall model of a building that can be successively worked into a detailed model suitable for creating software applications.

The IAI use the IFC Object Model to represent ‘classes’ and their ‘interfaces’, ‘attributes’ and ‘relationships’, in what they describe as a composite representation. The IAI describe classes as ‘object-oriented programming components’ that are used to define the AEC/FM data objects (which can be physical or abstract). They say that interfaces are used to provide access to the objects and enable software vendors to implement IFC-based objects.

Examination of the IFC Object Model shows that it has been shaped by the STEP standards and their developers³. The information model is defined in EXPRESS and illustrated in EXPRESS-G, while the ‘IFC Core Model’ is a direct descendant of the Building Construction Core Model^[49]. The IFCs have also adopted and adapted the STEP Generic Resources⁴ including those for measurement^[44] and geometry^[45].

³ This is not surprising, since many of the member organizations of the IAI contain a number of people who were, or still are, involved in the development of the STEP standards.

⁴ This is not unreasonable, as these aspects are well defined in STEP.

It should be noted that Release 1.5 of the IFCs concentrate on architectural design information for layout and three-dimensional visualization. The type of information that would be produced by a civil or structural engineer will not be included in the IFCs until Release 3.0.

2.5 XML

CIS/2 was five years in the making, while its predecessor – CIS/1 – spent a similar period in gestation. In that time, many things in the IT world changed; of most significance has been the rise of the Internet and its user-friendly interface, the World Wide Web. As the Internet evolved, so did several important technologies; the most talked about of which is XML^[7] – eXtensible Markup Language; an extensible subset of the Standard Generalized Markup Language SGML^[36]. In theory, this allows a document to be ‘tagged’ with metadata, placing its content into an appropriate context. Because it is perceived as easy, with low-entry costs, XML has been seen as a *de facto* standard transfer syntax; facilitating data exchange, rather than simply document sharing. Indeed, many businesses turned to XML for business-to-consumer (e.g. e-retailing) and business-to-business (e.g. e-marketplace) solutions to reduce transaction costs, open new markets and better serve their customers.

However, it should be noted that XML is not actually a data exchange specification, nor a markup language. It is a specification that allows anybody to write their own markup language. The problem is that many diverse groups of people have written their own markup languages⁵ with little strategy or coordination; repeating the discussions and mistakes first seen in the IGES arena. Because the closest thing the Web has to a governing body is W3C⁶ it follows that XML does not have the status of an International Standard, merely a ‘W3C Recommendation’.

2.5.1 aecXML

One initiative that stood a good chance of success was aecXML⁷. Initially a product of Bentley Systems Inc., aecXML grew to become an international project, and is now within the remit (as a subproject) of the International Alliance for Interoperability (IAI). aecXML is an XML-based language used to represent information in the Architecture, Engineering and Construction (AEC) industry. This information may be resources such as projects, documents, materials, parts, organizations, professionals or activities such as proposals, design, estimating, scheduling and construction. It was intended that aecXML would facilitate communications and e-commerce among the various organizations involved in AEC projects.

2.5.2 STEPml

Since the Web was designed primarily to *present* documents and STEP was designed primarily to *represent* and *exchange* data, it would seem logical to combine aspects of these technologies to create a mechanism that can *represent* and *present* information via the Internet. STEP and XML were developed by different types of people for different

⁵ The number of MLs in development is too long to list here; for a list of ‘official’ developments see - <http://www.xml.org/>

⁶ The World Wide Web Consortium - <http://www.w3c.org/>

⁷ See <http://www.aecXML.org>

purposes, and the emerging technologies are very different for good reasons. XML arose from the need to present documents to a human end user, rather than the more onerous task of representing engineering data for consumption by remote software applications. As a basic markup language is transformed into a data definition language, XML technology is now taking the Web to places it never thought it could go. However, before we can create the 'semantic' Web, applications must be able to understand the content and concepts of business domains.

In XML, W3C has provided a generic technology framework upon which bespoke solutions may be built. Yet there are still several aspects missing from the XML framework and many e-commerce solutions. The first missing aspect is a range of predefined and compatible data models. These data models are needed to capture the information concerning the design, manufacture and supply of products. Most importantly, W3C needs a formal and authoritative testing regime. This is needed if end users are to be assured that an implementation will perform as promised. It is possible that W3C will evolve into something more formal; i.e. a standardization body. Alternatively, ISO could become the governing body of certain XML implementations – those wishing to be internationally acceptable.

One initiative where the two technologies are converging is STEPml⁸, a library of XML specifications - Document Type Definitions (DTDs) and XML Schemas for product data. STEPml's content is based on information models from STEP. It combines the semantically rich, international standard data models from STEP with the widespread infrastructure of XML and the Web. The use of STEPml will enable companies to communicate information about products with their business partners using the Web.

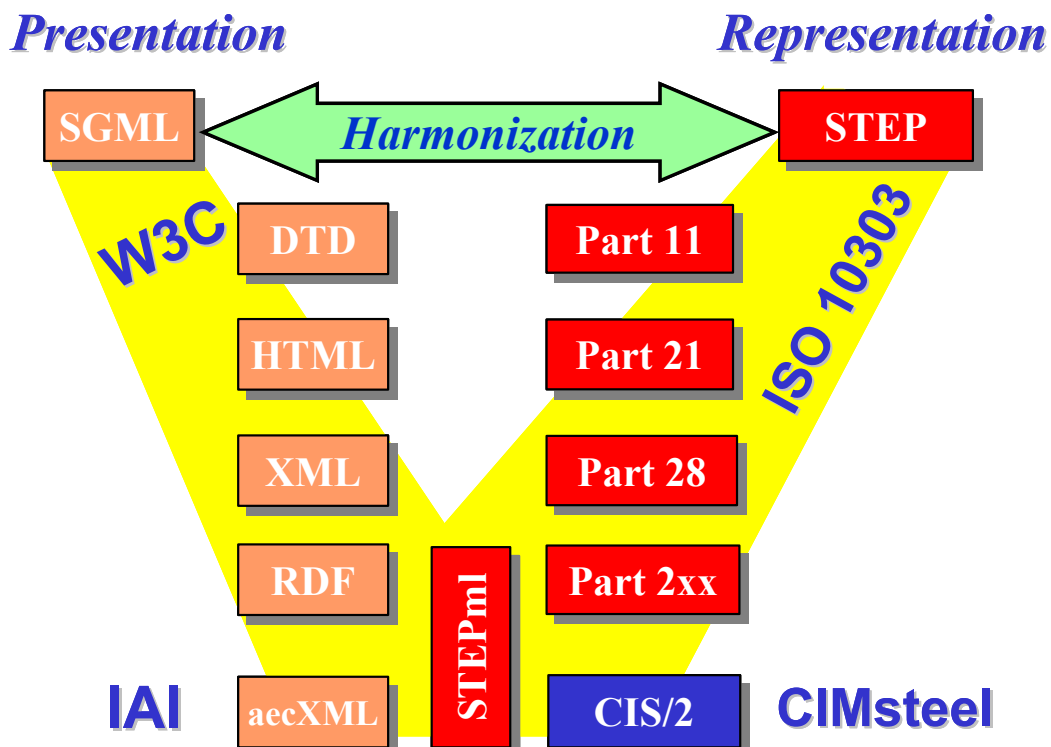


Figure 2.2: The potential convergence of technology

⁸ <http://www.stepml.org/>

2.6 The evolution of CIS/2

2.6.1 From CIS/1 to CIS/2 (1st Edition)

Figure 2.3 illustrates the evolution of CIS/2, and the influence that AP230 had on its development. As mentioned earlier, CIS/1 was launched in 1995 and has since been implemented in several commercial software applications. Shortly after its launch, Leeds University began work on AP230, effectively rewriting CIS/1 in the formal style of ISO STEP. As part of the normal AP development process, the Working Draft (WD) version of AP230^[55] was issued for international review. It was at this stage that the developers of AP230 realized that it would not be accepted as an International Standard in its WD state. However, the feedback received from the international review of the Working Draft of AP230 was invaluable during the development of the second release of the CIMsteel Integration Standards (CIS/2).

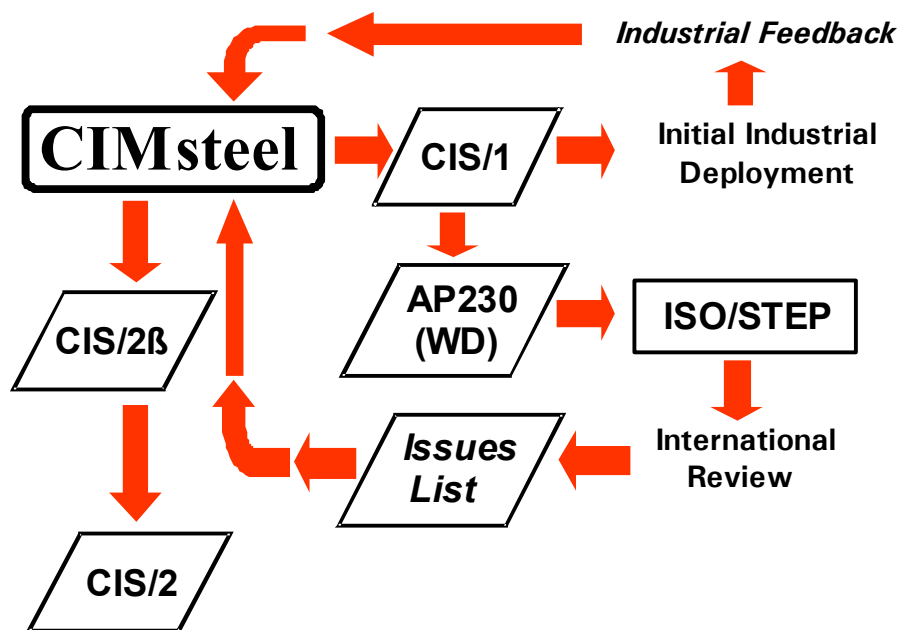


Figure 2.3 *The evolution of CIS/2 (1st Edition)*

CIS/2.0 also benefited from the feedback received from the industrial deployment of CIS/1. Indeed, so many issues were raised (in terms of scope and functionality) by the CIMsteel collaborators and the international review team of AP230, that the CIS had to be completely rewritten to address them all. The result was CIS/2.0 – an enhanced and extended second-generation release of the CIMsteel Integration Standards. CIS/2.0 was the culmination of all that was learned through development of AP230 and the implementation of CIS/1.

The CIMsteel Project was completed early in 1998, at which time Leeds University issued an incomplete Beta version of CIS/2 to former CIMsteel Collaborators and ‘CIS Registered Developers’. Following the end of the CIMsteel project, the Steel Construction Institute were asked by the project manager (Taylor Woodrow), to act as the custodians of the project deliverables, including the CIS, on behalf of the other CIMsteel collaborators. Consequently, CIS/2 is documented in six formal SCI publications, each of which has been subjected to the rigorous quality control regime established by previous SCI publications.

2.6.2 The Second Edition (CIS/2.1)

In the 3 years since the first edition of CIS/2 was published, the SCI and AISC have encouraged the implementation of CIS/2 through a series of biannual workshops and demonstrations and national conferences. As a consequence, many common software programs in Europe and the USA, now have commercially marketed CIS/2 translators. These CIS/2 users have raised 103 issues. Most of these have been trivial, but the 25 'Major Technical' issues warranted the development of a second edition of CIS/2. Further, second editions of the STEP Generic Resources^[44, 45, 46, 47] used in LPM/5 were published in 2000; the revised constructs contained therein are incorporated into LPM/6.

In summary, 124 EXPRESS constructs have been added to LPM/6 for the 2nd Edition of CIS/2. A further 112 have been modified. Many of the changes to the Logical Product Model have been made to accommodate the information requirements of MIS systems, including:

- Drawing files
- Cost codes
- CNC files
- Sequences and lots

Other changes were required to represent:

- Design loading
- Cambered assemblies
- Various types of Welds
- Fasteners with position
- Pock marks
- Managed data without data history

This 2nd Edition should be seen as a 'point release', rather than a whole 'new' release of the CIMsteel Integration Standards. One overriding consideration was that LPM/5 would be upwardly compatible with LPM/6, minimising the migration effort required for those developers of CIS/2 translators.

Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

2.6.3 CIS/2 International Technical Committee (ITC)

The development of the second edition was coordinated by the CIS/2 International Technical Committee (ITC); the body responsible for improving and maintaining CIS/2 as an open standard. Chaired by Chuck Eastman, the ITC comprises The Steel Construction Institute and Leeds University (the joint custodians of CIS/2), together with The American Institute of Steel Construction (AISC), Georgia Institute of Technology and the National Institute of Standards and Technology (NIST). The successful deployment of CIS/2 technology over the past years has been due to the efforts of software vendors who have developed commercially viable CIS/2 translators.

2.6.4 NIST & the AISC EDI Initiative

As mentioned earlier, the American Institute of Steel Construction (AISC) made Electronic Data Interchange a major initiative in North America, and adopted CIS/2 as a key enabling technology. At the request of the AISC, the Building and Fire Research Laboratory (BFRL) at NIST embarked on a three-year effort to help the American construction industry understand, evaluate, implement, and extend the CIS/2 standard. In 2002, BFRL established close working relationships with companies participating in the AISC Electronic Data Interchange initiative including:

1. principal steel design and detailing software vendors
2. principal structural analysis software vendors
3. principal steel fabrication management software vendors, and
4. principal end- user organizations

BFRL work assisted these partners in the interpretation of the CIS/2 standard and in the testing of CIS2-based file translators for their software products. The BFRL work has included making a CIS2-to-VRML translator⁹ (discussed earlier), which makes it possible for any implementer or user to visualize and test a CIS representation of a steel structure.

The BFRL work accelerated the implementation and adoption of CIS2 in the US steel construction industry. The benefits of this work include shortening cycle times in steel design, analysis, detailing, fabrication, and erection and reducing project delivery costs, as a result of eliminating non-value added work processes involving the manual interpretation and re-entry of data by each project participant.

2.7 The future of CIS/2

2.7.1 CIS/2 ITC member activities

As mentioned earlier, the body responsible for improving and maintaining CIS/2 as an open standard is the International Technical Committee (ITC). Issues raised via the CIS/2 web site, the EDI email exploders, and AISC conferences are compiled and submitted for action to the ITC. The primary members of the ITC – SCI, AISC, Georgia Tech and NIST – will continue to work with vendors to improve their CIS/2 translators.

In collaboration with the AISC EDI team members, NIST also plan to continue their work in three areas:

1. The definition and development of an *interoperability testbed* to be used in evaluating how individual CIS/2-enabled software products can be used jointly in the project delivery process.
2. The development of a CIS/2-aware *product data repository* to support this interoperability testbed.

⁹ Available on the World Wide Web (through <http://cic.nist.gov/vrml>)

3. The definition of the functional requirements and possible technical solutions for the next edition of the CIS/2.

Work also continues at Georgia Institute of Technology on the implementation of advanced project databases based on CIS/2. Built around Microsoft SQL Server, their work incorporates an object-oriented information model within a relational DBMS; tables and fields are defined from EXPRESS constructs in the LPM. With the inclusion of some basic data management tools, they are taking their first steps towards implementing a Product Model Repository (PMR).

2.7.2 The AISC EDI Initiative: Phase III

Although firms within the process plant sector have been using 3D models for years, the vast majority of practitioners in the building sector are still only scratching the surface of the power that modern 3D modelling tools possess. In the commercial buildings sector, the '3D modelling approach' requires a radical shift in workflow; not only between the fabricator and the design team, but also between the architect and the structural engineer.

During 2003, the AISC will launch its next big push for EDI in the USA. While most of the necessary tools are now in place, the next step is to convince the members of the building team to use them. The AISC is beginning to see the bigger picture – the need to get the owner and architect on board, not just the structural engineer. The institute is planning to reach out to their representative groups through a new task force created to develop an EDI business model for steel. The first step is to develop contracts that allow for the exchange of 3D models as contract deliverables. The language developed will be incorporated into a code of standard practice in the AISC's 2005 Unified Manual of Steel Construction.

2.7.3 The impact of XML on CIS/2

It has been suggested that XML will render data exchange specifications, such as CIS/2, redundant. However, this position ignores the complex requirements of data exchange. Far from making CIS/2 redundant, XML implementations highlight the need to resolve several issues that have already been addressed in the product modelling world.

The *Implementation Guide*^[16] discusses several different types of CIS implementation, from basic file exchange through to advanced Database Management Systems (DBMSs). The various degrees of complexity allow software vendors to extend the functionality of their CIS implementation incrementally. It should be noted that these 'implementation levels' were those identified within the STEP community and are not the only possible forms of implementation. It is likely that XML will become an accepted alternative to STEP Part 21 as an implementation form of an EXPRESS schema. This may well become a practical reality quite soon. However, until there are agreed standards (and supporting toolkit software) for representing EXPRESS data structures in XML, it would not be advisable to base industrial solutions on XML. Currently, STEP Part 28^[42] is a draft technical specification under ISO/TC184/SC4.

2.7.4 The long-term strategy

Data exchange standards will go through an evolutionary development and deployment process. Because of the balloting and review cycles involved in developing a STEP AP, it takes several years before an AP achieves International Standard (IS) status. This time scale is simply not acceptable to engineering end users in the construction industry, and so CIS/2 provides the pragmatic solution that will meet the needs of these users for the

interim period. One possible strategy for the evolution of data exchange standards for structural steelwork is as follows:

- CIS/2 provides the short-term solution, facilitating data exchange and information management within the structural discipline along the 'steel supply chain' (e.g. between structural engineers and fabricators), for the foreseeable future
- VRML will continue to be a valuable tool for viewing CIS/2 data and testing CIS/2 implementations. It may also emerge as the focus of integration; allowing web-based collaboration between project participants
- It is likely that XML will become an accepted alternative to STEP Part 21 as an implementation form of an EXPRESS schema, although the verbosity of XML files will reduce their attraction as a mechanism for sharing engineering data. Moreover, by that stage, database implementations should be more common, rendering physical file transfer less important.
- It can be seen that IAI IFC Release 3.0 could provide the medium term solution, facilitating data exchange and information management across disciplines with the construction industry (e.g. between structural engineers and architects)
- It is envisaged that STEP APs will provide the long-term solution, facilitating data exchange and information management across all industries (e.g. between steel producer and all their suppliers and customers)

Both IAI and CIS rely extensively on the use of the STEP standards and the associated technology. In both cases, the format of the data exchange file is specified by STEP Part 21, and the meaning of the data contained within the file is derived from a logical schema written in accordance with STEP Part 11. Both IAI and CIS also use constructs contained in the STEP Generic Resources, in particular, those for measurement and geometry.

Thus, CIS, IAI and STEP can all be seen as 'stepping stones' that allow information to be shared between users on their 'islands of automation'; the scope and visibility of that information becoming greater with each leap.

This strategy is not unique to the construction industry. Indeed, the APs developed for the process industries and those for oil and gas are all based on *de facto* standards arising from initiative such as POSC/CAESAR, Process Base, etc^[61, 70, 76]. Commercial organizations in these industries see that the best standards for data exchange evolve through practical use rather than through academic exercise. The resulting APs are much stronger for these 'pre-development' stages, and are much more likely to be accepted by industry, than if they had followed the 'purest' route.

3 PRODUCT MODELLING

3.1 The Logical Product Model (LPM)

The CIS are based upon a formal ‘Product Model’ known as the ‘Logical Product Model’ (LPM). Product modelling is mainly concerned with flows of information. A Product Model standardizes the way engineering information is held, to facilitate the unambiguous transfer of information between computer systems. The scope of the Logical Product Model determines the maximum scope of the information that may be shared using CIS-compatible software. The version of the LPM that forms part of the CIS/2.1 specifications is known as LPM/6.

The LPM itself is simply a formal specification; effectively it is a ‘pattern’ to make a ‘container’ to hold product data. The product data is the significant engineering information relating to the design, construction and maintenance of a specific structural steel framework.

3.2 What is a Product Model?

The concept of the Product Model emerged with the development of ‘next generation’ Database Management Systems and data exchange standards (such as STEP). The use of the word ‘product’ underlines the fact that the ‘new generation’ of data exchange standards were intended to enable software to share engineering information, rather than simply graphical representations. The CIS defines a Product Model in the following way:

“A Product Model is a formal description of types of ideas and facts, with an explicit set of explanatory rules, which together form a simplified yet complete, accurate, coherent and unambiguous (computer sensible) representation of a portion of interest of physically realizable artefacts made by manufacturing processes.”^[12]

A Product Model is ‘simplified’ because there is no need to represent the full complexity of the real world; yet it is ‘complete’ for the purposes of communicating information between the human end users of Computer Aided Engineering (CAE) systems. It must be ‘computer sensible’, as there would be little point in developing a Product Model that cannot be used within CAE systems. The reference to ‘types of ideas and facts’ emphasises the role the Product Model plays as a ‘pattern’ or ‘template’ model as opposed to a populated data model.

A Product Model defines a logical structure for data in terms of entities (or things), attributes (or characteristics), and relationships between entities. This structure is independent of specific software. However, it allows the vendor of a particular application to map information to and from that application’s internal representation of the data to and from that of the Product Model. By this mechanism, it becomes possible to transfer engineering information unambiguously between different vendor’s application software.

3.3 Product Modelling Theory

The theory of Product Modelling is founded on the theory of conceptual modelling^[5, 6, 8, 37] which is typically the first stage in the process of developing database applications. All computer databases depend on some underlying 'data model'. This is an abstraction that allows the meaning of the data to be captured by placing restrictions on the relationships that can exist between data items, and the values they are allowed to take^[5]. The data model is a snapshot that captures the static features of the world onto allowable combinations of data items in the model. A good data model will be structured in such a way that it is possible to define a comprehensive set of operations which can be applied to the data without causing any inherent integrity constraints to be violated^[57].

An understanding of how information can be represented is one of the keys to communication^[12]. Both humans and computers use *models* to represent aspects of the 'real world', and the gap between the real world and that of the computer parallels the gap between *knowledge* and *data*. Mapping between a human view of knowledge and a computer view of data involves mapping between different types of inter-related model, which all use different languages. From a human viewpoint, the 'real world' is often represented using drawings and documents. While documents use a natural (textual) language composed of sentences containing nouns, verbs, adjectives and adverbs, drawings use a symbolic language composed of graphical objects such as points, lines, and curves. From the computer's viewpoint, these natural languages are incomprehensible and are as unnatural to it as the computer's binary data is to the human user.

The 'real world' of structural engineering uses a (predefined) lexicon of terms and symbols that act as substitutes for the engineering concepts of their domain, which are only relevant within a given *context*. The 'product modeller' captures these concepts in a formalized and structured manner. In the case of the CIS, the formal languages used at the 'conceptual level' include IDEF0^[63], IDEF1X^[21, 64], as well as 'structured English'. The resulting 'conceptual model' is then interpreted for implementation in a computer system. The result of the 'interpretation process' is a 'logical model', which uses a formal data definition language (in the case of the CIS, the EXPRESS Language^[39]) composed of a schema containing entities, attributes and constraints.

For the CIS, both the conceptual level and the logical level are important. The conceptual model represents *information* about some aspect of the real world. The logical model remodels (or interprets) the conceptual model to produce something that can be implemented in computer systems that hold *data constructs* to represent this information. When implemented as a mechanism for data exchange, the logical model gives meaning to the data held in a (physical) data exchange file.

4 THE SCOPE OF CIS/2

The scope provided by CIS/2 strikes a balance between the desire to include all the information that can be associated with any type of steel structure during its full life cycle, and the practical need to limit the size and complexity of the resulting model. The Logical Product Model contained in CIS/2.1 (LPM/6) provides for the planning, design, and construction phases of the life cycle of steel framed structures, as well the future re-engineering of structures. It also provides input into facilities management.

LPM/6 addresses the engineering of low, medium and high rise construction in domestic, commercial and industrial contexts. It covers both the main structural steelwork and the secondary steelwork (such as purlins, siderails, cleats and cladding). The frames can be braced or unbraced. Connections can be considered pinned, rigid, or semi-rigid; the latter two may be full or partial strength. The frames are built from manufacturing assemblies, which in turn, are made up of parts and joint systems. The parts can be prismatic (e.g. rolled sections), two-dimensional (e.g. plates and sheets) or three-dimensional (e.g. castings). The parts may be rolled, welded, cast, or cold-formed. They can also be described as curved, cambered or tapered. The joints can be bolted, welded, pinned or complex.

CIS/2 provides mechanisms that allow standard items, such as bolts, nuts, washers, and rolled sections, to be 'passed by reference' rather than by attribute value. Similar mechanisms for manufacturers' items (such as cold-formed purlins) and library items are also provided.

Although the CIS has been developed primarily to enable the engineering of building frames, it can also be applied to other types of steel frame. For example, industrial structures such as process plant installations, transmission towers, and (to some degree) offshore structures. Given the wide range of possible structural forms, it is difficult to give definitive statements on what proportion of steel structures can be fully represented using LPM/6. In the case of conventional building frames, the structural steelwork of almost 100% of buildings can be fully represented, together with at least 95% of any non-structural steelwork and 90% of the related non-steel items (such as grout). These percentages will be a little less for industrial structures. Other steel structures, such as bridges and offshore structures, have specialist aspects that cannot be captured fully by LPM/6. However, it is likely that LPM/6 could be used to capture a substantial proportion of the relevant information.

CIS/2 makes no explicit provision for sharing information about items such as:

- Concrete, masonry, timber and glass components
- Heating, ventilation and air conditioning
- Architectural components (windows, doors, etc.)
- Control of, and programming for, NC or CNC machinery

5 A CONCEPTUAL OVERVIEW OF LPM/6

This Section provides a concise overview of LPM/6 and the terminology it employs. Words shown in *bold italics* refer to Application Objects contained in the Application Reference Model (ARM) that are used to represent the engineering concepts required by the users of CIS/2-compatible software. Each of the concepts in the ARM has been remodelled in LPM/6 in a more data-centric manner.

A *structure* may be placed in the context of a *project*, a *site*, a *building* or a *building complex*. Each of these may be divided into *zones*. Buildings, sites and structures may all have interrelated *grids*, with *setting out points*, which may be resolved to *geographical locations*. CIS/2 supports *orthogonal*, *radial* and *skewed* grids.

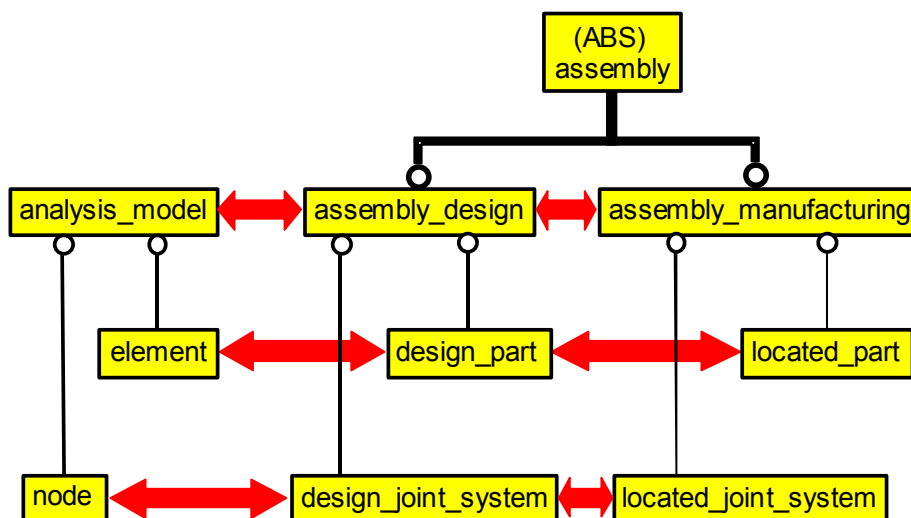


Figure 5.1 The 'three views' in LPM/6

There are three different ways of representing the structure using the constructs contained in CIS/2. These three views are illustrated in simplified EXPRESS-G in Figure 5.1, representing the viewpoints of the analyst, designer and manufacturer. These are sometimes referred to as three models: *analysis model*, *design model* and *manufacturing model*. Any CIS/2 file may contain several aspects of each model. As a project progresses, more information will be created. The three models - complete with references to external documents, drawings and CNC data - may be compiled to form a combined CIS/2 model; one that can be shared (perhaps via a web-based project extranet) with other members of the design and construction teams, accessible in a VRML viewer. The use of these models is illustrated in Figure 5.2.

5.1 The analysis model

For analytical purposes, the structure is represented using *analysis models*. These are made up of *nodes* and *elements*. The elements are categorized by their *dimensionality* as either *point elements* (zero-dimensional), *curve elements* (one-dimensional), *surface elements* (two-dimensional) or *volume elements* (three-dimensional). Elements may be represented using implicit or explicit geometry. The structure - in the form of the analysis model - may be restrained at the nodes. These *boundary conditions* may

represent a pinned or fixed support, or a (linear or non-linear) spring. The elements may be eccentric from their connecting nodes. The ends of the elements may be released (as a pinned end, a linear spring or non-linear spring) or remain rigid. **Loads** may be applied to the nodes and elements. Elements may be subjected to **concentrated**, **distributed** or **thermal loads**. The loading takes the form of **basic load cases**, which result from **physical actions** (e.g. **permanent**, **variable**, **accidental**, or **seismic**). **Applied loads** can be described as **static** or **dynamic**. Static loads are defined in terms of **displacements**, **forces**, or **pressures**, while dynamic loads are defined in terms of **velocities** or **accelerations**, associated with sets of static loads. The basic load cases are combined and factored to form **loading combinations**. A number of different **static** and **dynamic analysis methods** are supported. Results of the analysis may be given at nodes, element ends, or at a point within the element. These **analysis results** can be described in terms of **displacements**, **forces**, **velocities** or **accelerations**, and may be combined as **sets**.

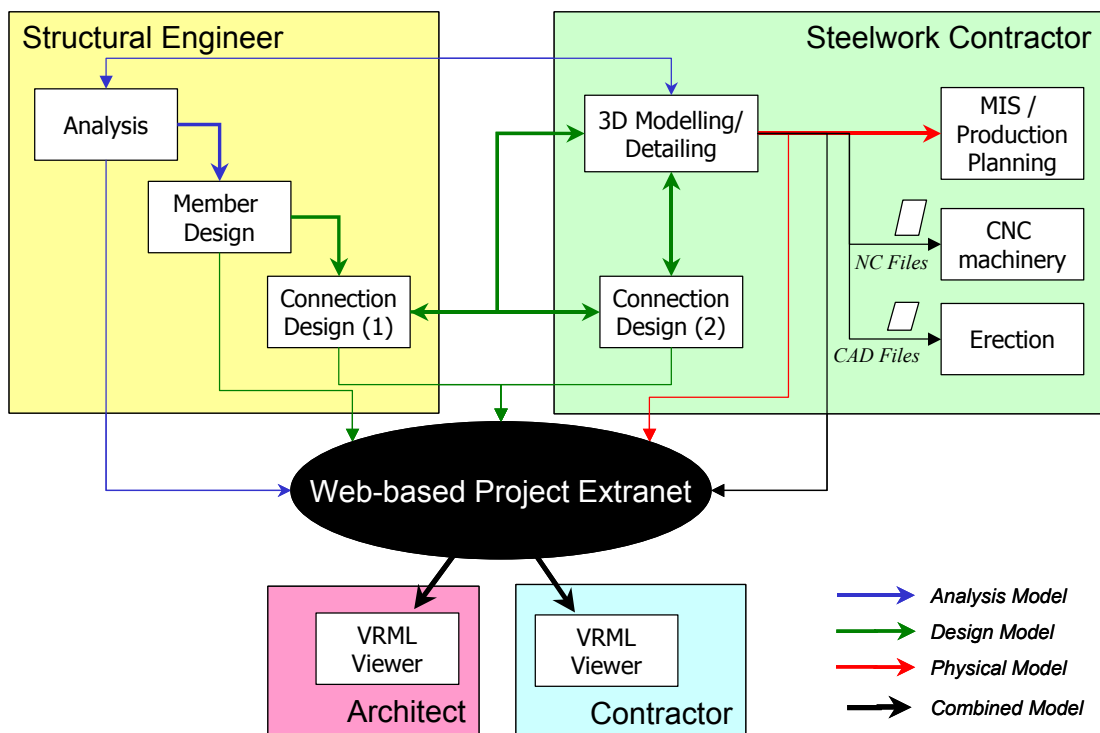


Figure 5.2: Use of the 3 models in CIS/2

5.2 The design model

For the purposes of member and connection design, the structure is represented using **design assemblies**. These may be decomposed into other design assemblies. The design assemblies are categorized by their **functional role** as either **structural members**, **structural connections**, or **structural frames**. The structural members are categorized by their **dimensionality** as either **linear** (one-dimensional), **planar** (two-dimensional) or **cubic** (three-dimensional). The shape of an assembly may be represented using explicit geometry. Design assemblies ultimately comprise **design parts** and **design joint systems**. A design part is a conceptual representation of a basic piece of steel. A design joint system is a conceptual representation of a basic joint system. **Design results** can be described for structural members and structural connections expressed in terms of **elastic** or **plastic resistance**.

5.3 The manufacturing (or physical) model

For the purposes of detailing, production preparation, and manufacturing, the structure is represented using *manufacturing assemblies*. These may be composed of other manufacturing assemblies. Manufacturing assemblies are ultimately composed of *located parts* and *located joint systems*. A located part is a representation of a basic physical piece of steel. A located joint system is a representation of a physical joint system (e.g. a bolt group or a weld).

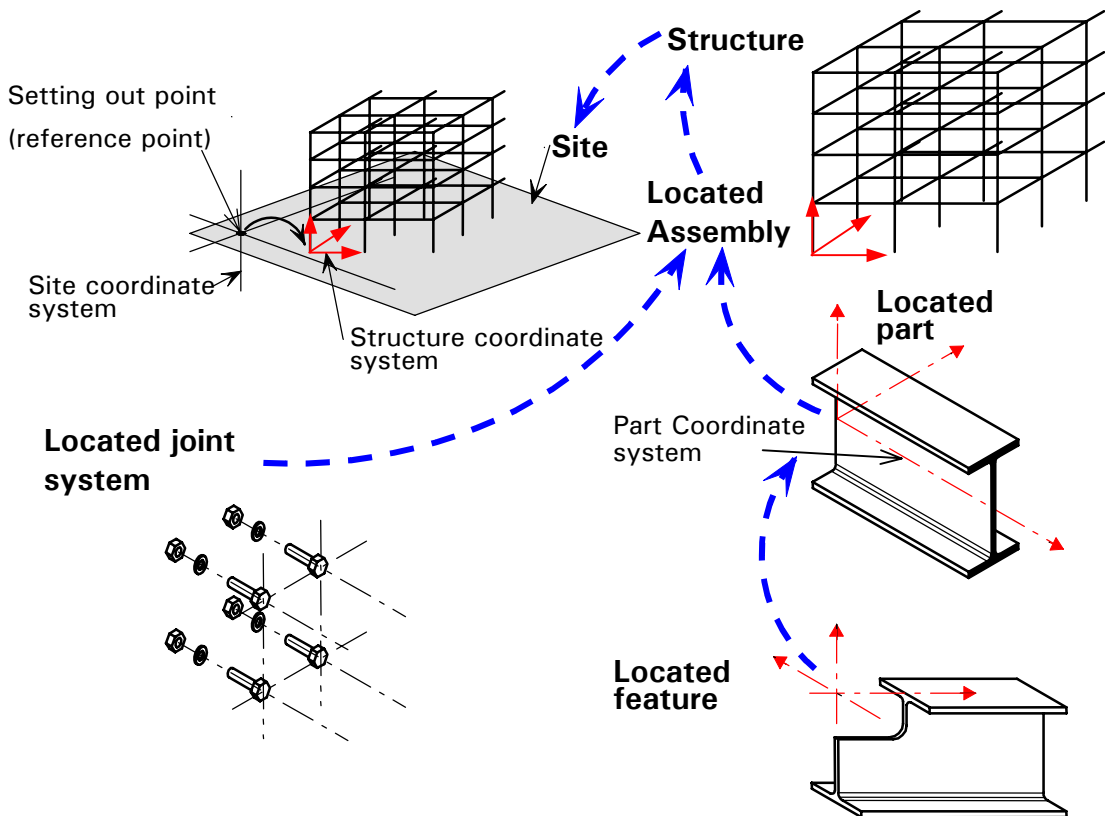


Figure 5.3 Located parts, features and joints in LPM/6

Parts are categorized by their dimensionality and the complexity of their shape definition, e.g. *prismatic*, *sheet* or *complex*. CIS/2 supports a comprehensive range of *section profiles* and their associated *section properties*. *Joint systems* may be *mechanical* (e.g. a bolt group), *chemical*, *welded*, *complex* (e.g. welded shear studs) or *amorphous* (e.g. grout). CIS/2 supports a comprehensive range of *fasteners* such as self-drilling and self-tapping screws. *Features* are used to modify assemblies, parts and joint systems. Features can be applied along an *edge*, to a *surface* or as a *volume*. Volumetric features include *notches*, *chamfers*, and *holes*. Features also include *surface treatment* and *name tags*.

CIS/2 allows generic *products* to be defined and their properties exchanged implicitly or explicitly. To do this, CIS/2 identifies four classes of product item, *Standard*, *Proprietary*, *Library*, *non-standard items*. Standard items are internationally recognised generic product items (such as hot-rolled sections) which comply with formal standards that have international visibility. Proprietary items are manufacturer-specific product items (such as cold rolled sections) which comply with manufacturer's catalogues. This class also includes product items that comply with specifications agreed between a consortium of manufacturers. Library items are product items that are not explicitly

covered in standard or manufacturer's catalogues but have been compiled and placed in a 'library' by third party trade associations (e.g. a 'parts library'). The first three classes of items may be 'passed-by-reference'. The fourth class, non-standard items, covers product items that cannot be passed-by-reference because an agreed reference does not exist. These must be passed explicitly by attribute value.

Analytical elements, parts and fasteners may be defined with or without a *material* or *location*. Materials may be described as *isotropic*, *orthotropic* or *anisotropic*. CIS/2 supports a comprehensive range of *material properties* including *density*, *strength*, *elasticity* and *hardness*. These properties may be described as being dependent on *strain*, *loading* or *temperature*. Locations may be defined by a *coordinate system*, a *grid offset*, a *map reference* or *global location* (longitude, latitude, and altitude). CIS/2 supports *Cartesian*, *cylindrical* and *spherical* co-ordinate systems.

CIS/2 also includes some limited coverage of *contractual organisation*, *project planning*, *production scheduling* and *costing*. CIS/2 allows generic types of *products* to be defined with or without *shape*, *material* or *price*.

CIS/2 places no restrictions on the use of particular *design criteria*, Codes of Practice, Building Regulations, or National Standards. Almost any form of design technique can be supported, including Limit State, load resistance factor, or permissible stress. Although CIS/2 provides guidance on the usage of *unit systems*, and requires the support of (at least) one of the predefined sets of units, LPM/6 enables applications to define additional units as required using the constructs from STEP.

5.4 The Second Edition

CIS/2.1 now supports explicit positioning of fasteners – such as bolts, nut and washers. It also includes comprehensive representation of welds. CIS/2.1 adds support for MIS systems, including: *drawing files*, *cost codes*, *CNC files*, *sequences* and *lots*.

CIS/2.1 now supports a comprehensive range of *welds* such as butt, fillet, spot, plug, tee, corner, lap, edge, cruciform, stud, surfacing, flared, bevelled, square, scarf, seam, and groove welds. The shape of the weld may be represented implicitly or explicitly.

Fasteners – such as bolts, nuts and washers – may be positioned explicitly within a fastener mechanism relative to the joined parts.

Volumetric cutting features may be assigned a limit to their depth of penetration.

6 THE DEVELOPMENT OF CIS/2

6.1 Drivers for the development of CIS/2 & LPM/5

The first edition of CIS/2 was not merely a simple evolution of the previous release of the CIMsteel Integration Standards but a radical departure from it. LPM/5 was a major redevelopment of LPM4CIS, the version of the LPM contained in CIS/1. This arose for a number of reasons^[12], some of which are explained below. The main drivers for the development of CIS/2 may be identified as:

- Feedback from vendors and end-users on their experiences of using CIS/1, including their demands for additional scope and functionality.
- The work of other parts of the CIMsteel project (particularly that of the Overall Design and Analysis Working Group).
- The availability of improved software tools for modelling, testing and implementation.
- The requirements of data management, incremental updating and unique identifiers.
- Feedback from vendors on their experiences of conformance testing under CIS/1, including demands for more rigorous and detailed conformance testing.
- The development of other product models in other countries.
- Demands from some vendors and end-users for greater STEP-compatibility (e.g. to incorporate constructs from the STEP Generic Resources into the LPM).
- The development of AP230.
- The ease of implementation.

6.2 Extending the scope & functionality of CIS/1

This Section goes into greater technical detail than the earlier Sections, and assumes some familiarity with the terminology. It is really intended for readers who are already familiar with CIS/1 and wish to understand better the differences between CIS/1 and CIS/2.

6.2.1 Omissions from CIS/1 included in CIS/2

LPM/6 builds upon the scope of LPM/5, which included coverage of the following items that were excluded from CIS/1:

Project definition issues

- Geographical locations (by map reference or longitude and latitude) related to grids and setting out points.
- Buildings, building complexes and their relationship to the structure.
- Zoning of projects, building, structures and sites.
- Contractual arrangements and relationship within and between companies.

Design issues

- Structural members classified by type, dimensionality, primacy and use.

- Structural members described as springs, cables, wires, ties, pipes, etc.
- Stairs, ramps, floors, etc.

Analysis issues

- 2nd order elastic analysis (P- δ).
- Dynamic analysis.
- 3D solid modelling and detailed FEA methodology.
- Dynamic / cyclic loading - earthquake, explosions, vibration.
- Spherical and cylindrical co-ordinate systems.

Detailing issues – parts

- Complex parts – defined using explicit geometry.
- Castellated and ‘Cellform’ beams (with regular features).
- Cambered beams (with a relative or absolute offset within the span).
- Curved beams (with a non-linear longitudinal axis).
- Compound parts made up from 2 or more standard parts (e.g. a plate welded to an I-section, crane rails, plate girders).
- Derived parts (cut from other parts).
- Tapered parts (with an absolute or relative taper).
- General polygonal section profiles (defined by a series of points, or defined by their edges).
- Orthotropic and isotropic materials.
- Dimensions with tolerances associated with them.
- Library items.
- Generic product definition.

Detailing issues – features

- Complex features (non-standard features whose geometry is defined explicitly in 3D space).
- Non-standard threads and threaded holes.
- Name tags.
- Surface treatments and preparation (chemical wash, blast clean, etc.).
- Coatings (corrosion, fire, etc.).

Detailing issues – joints

- Holding down bolts.
- Studs, threaded rods, pins, etc. used in joint systems.
- Complex fasteners (e.g. self-drilling, self-tapping screws).
- Complex joint systems (e.g. combined chemical and mechanical systems).
- Amorphous joint systems.
- Non-linear weld paths.

- Shear connectors (for composite construction).
- Pinned joint systems.
- Grout, filler, adhesives, etc.

Data management issues

- Data sharing / management.
- Version control / database management / data security etc.

Other issues

LPM/5 also provided limited coverage of the following items:

- Contractual arrangements and relationship within and between companies.
- Offshore structures (although there is no special provision).
- Bridges (although there is no special provision).
- Transmission towers (although there is no special provision).
- Composite construction (the steel elements, decking, shear studs, etc.).
- Quality issues & certification.

In addition to all of the above, LPM/6 now provides coverage of the following areas:

- Cost issues.
- Project planning & scheduling.

6.2.2 Presentation of the LPM as a single schema

While LPM4CIS (the version of the Logical Product Model underpinning CIS/1) was developed and presented as 13 separate schemas, LPM/5 was developed as one single schema. This means that the implementation form of LPM/6 – as with LPM/5 – (the Long Form EXPRESS Schema^[18]) is a continuous list, with its entities ordered alphabetically. The more detailed explanation of those entities is, however, presented under 17 subject areas, some of which relate to the 13 schemas of LPM4CIS. This was done so that those users familiar with CIS/1 could see the evolutionary development of the CIS. The relation between the CIS/2 subject areas and the CIS/1 schemas is shown in Table 6.1.

It should be noted that the subject areas in LPM/5 and LPM/6 do not have ‘hard boundaries’; there are many relationships between the different concepts in the separate subject areas. While many of the subject areas have extended the scope and functionality of the schemas in CIS/1, it can be seen from Table 6.1 that there are four new subject areas in CIS/2. These add considerable functionality to the CIS. The most important of these is probably that of data management, which includes the concept of ‘incremental updating’.

The subject areas for LPM/6 are the same as those used in LPM/5. No new subject areas have been created, although several subject areas have been expanded.

Table 6.1 *Relating CIS/2 Subject Areas with CIS/1 Schemata*

LPM/5 & LPM/6 Subject Area	LPM4CIS Schema
Decomposition	LPM_Decomposition_schema
Project Definition	LPM_Configuration_schema
Structural Modelling	LPM_Structural_modelling_schema
Loading	LPM_Loadings_schema
Structural Response	LPM_Structural_response_schema
Parts	LPM_Parts_schema
Features	LPM_Features_schema
Joint Systems	LPM_Joints_systems_schema
Materials	LPM_Materials_schema
Location	LPM_Location_schema
Grouping	LPM_Grouping_schema
Geometry	LPM_Geometry_schema
Units & Measures	Units_for_LPM_schema
Item References	<i>New for CIS/2</i>
Product Definition	<i>New for CIS/2</i>
Process Definition	<i>New for CIS/2</i>
Data Management	<i>New for CIS/2</i>

6.2.3 Introducing data management

Providing an adequate and efficient mechanism for data management within the LPM was seen as essential to the long-term success of the CIS. A further constraint was that ‘Data Management Conformant’ (DMC) and non-DMC applications have to be able to share data with each other. The key to data management is the use of unique identifiers for all items of data that are to be managed. CIS/2 provides this data management functionality in the LPM by separating the data from the ‘meta-data’, and then allowing those applications that can associate the ‘meta-data’ with the data to be shared (i.e. DMC applications) to do so. This ‘meta-data’ can include some (or all) of the following:

- The unique identifier for the item.
- The person who created the data.
- The date when it was created.
- Whether the data was new or old data that had been modified.
- The date when it had been modified.
- Whether the data has been approved or not.

In CIS/2, data management is considered to be the process of assigning meta-data to data. For example, a DMC application would be used to assign a unique identifier to an instance of ‘assembly’ and to place that ‘assembly’ in a set that stated who created the instance, when it was created, and whether it had been modified. This is discussed in more detail in other volumes of the CIS/2 documentation^[15, 18, 19].

6.2.4 Incremental updating

In simple terms, ‘incremental updating’ is the ability of a software application to add new information to an existing information set while maintaining the integrity of the information set. This is an essential requirement if software applications are to be fully integrated. Incremental updating is a simple form of data management that allows information flow between applications, regardless of the application’s scope, functionality, or position in the production process. In order for heterogeneous systems to support incremental updating they have to be able to maintain an identification system for each item of data. LPM/6 provides a mechanism that allows each data item to be uniquely identified. By comparing identifiers, CIS applications are able to control and track revisions to project data.

In the Second Edition of CIS/2, the basic requirements of DMC translators are less onerous than those specified in the First Edition, allowing CIS/2 conformant applications to maintain a means of identifying data without having to maintain the history of that data.

6.3 CIS/2 development as an Application Protocol

As mentioned earlier, the STEP mechanism for data exchange is described as an Application Protocol (AP). CIS/2.0 was developed as if it were a STEP AP, due to the concurrent development of AP230^[55]. The reason that CIS/2 has emerged before the formal AP is that the end users of engineering software demanded that a standard be made available before it could complete the long processes of interpretation, qualification, and integration required by ISO. The CIS was always seen as a forerunner to a formal AP. Moreover, the industrial deployment of CIS/2 will encourage the use of STEP technology in the construction industry and strengthen any future APs.

The structure of the CIS/2 documentation reflects the main components of a STEP Application Protocol, which are:

- A Scoping Statement
- The Information Requirements - which contain
 - The Units of Functionality
 - The Application Objects and Assertions
- Application Interpreted Model (AIM) - which includes
 - An ARM to AIM mapping table
- Conformance Requirements which includes
 - The Conformance Classes
- Application Activity Model (AAM), and
- Application Reference Model (ARM)

A mapping of the clauses of AP230 onto the documents of CIS/2 is shown in Table 6.2. For the CIS, both the ARM and the AIM are viewed as Product Models. That is, the ARM is viewed as a conceptual Product Model and the AIM as the logical Product Model. Using the STEP methodology, only the AIM is implemented; the other models merely provide the development route to the AIM. In STEP terminology, the AIM is normative while the AAM and the ARM are informative. Having said that, the style of

the AIM is influenced by the Conformance Classes; i.e. the development of the AIM is influenced by (if not dependent upon) implementation considerations.

Table 6.2 *Mapping the AP230 clauses onto CIS/2 documentation*

AP230 Clause		CIS/2 Part
1	Scope	Contained in Vol 1: Overview
2	Normative references	Contained in Vol 5: Conformance Testing And Vol 4: The Logical Product Model
3	Definitions and abbreviations	Contained in Vol 1: Overview
4	Information requirements	Vol 3: The Information Requirements
4.1	Units of functionality	Contained in Vol 3: Units of Functionality
4.2	Application objects	Contained in Vol 3: Application Objects
4.3	Application assertions	Contained in Vol 3: Application Assertions
5	Application interpreted model	Contained in Vol 4: LPM/6 - EXPRESS Schema (Short Form)
6	Conformance requirements	Vol 5: Conformance Requirements
Annex A:	AIM EXPRESS expanded listing	Vol 4 Appendix A: LPM/6 - EXPRESS Schema (Long Form)
Annex B:	AIM short names of entities	<i>Not available</i>
Annex C:	Implementation method specific requirements	Contained in Vol 5: Conformance Requirements
Annex D:	Protocol Information Conformance Statement (PICS) proforma	Contained in Vol 5: Conformance Requirements
Annex E:	Information object registration	Contained in Vol 5: Conformance Requirements
Annex F:	Application activity model	Contained in Vol 3: Application Activity Model
Annex G:	Application Reference Model	Contained in Vol 3: Application Reference Model
Annex H:	AIM EXPRESS-G	Contained in Vol 4: Appendix C: LPM/6 EXPRESS-G Diagrams

6.3.1 Defining the scope of the Product Model

The first task when developing a Product Model is to define its scope and the context in which data is to be exchanged. The context of a Product Model is the range in which it is valid whilst the scope of the model is what it contains. Models that are developed in different contexts will be incompatible (even if they have the same scope) while models that share the same context will fit together. The context of an ARM is particular, while the context of the AIM is universal. This is because although STEP was initially developed for manufacturing industry, manufacturing has been taken to include not simply automotive and aerospace products, but also those of Architecture, Engineering and Construction.

Thus, the development of LPM/5 began with an identification of the scope and context of the information to be exchanged. Much of this had already been determined during the development of the earlier versions of the LPM. As a general statement, the scope of the LPM will ultimately be all the information that could be associated with any type of steel structure, while the context could be stated as the entire life cycle of the steel structure. The LPM could eventually contain all the information required to plan, design, construct, use and demolish a steel structure. This, of course, would be a considerable quantity of information, requiring a Product Model of enormous size and complexity.

More realistically, it was decided that LPM/5 should focus on the planning, design, and construction phases of the life cycle. It was also thought that the scope should be limited to building-type structures. This would not prevent its use in the facilities management of a structure, nor would it be limited to buildings. Engineering end users in the process plant industry were seen as the most likely to benefit from the early implementation of CIS/2.

6.3.2 Defining the Information Requirements

The main source of the information requirements for the LPM was the document entitled *A Structured Approach to Design, Analysis and Code Check of Constructional Steelwork: Engineering Part*^[23], produced by the Overall Design and Analysis Working Group of the CIMsteel Project. It also provided the hierarchical breakdown of the information requirements, from which, a new activity model could be produced.

6.3.3 Activity Modelling using IDEF0

Once the scope of the Product Model has been agreed, the AAM is used to represent the activities (or processes) that are involved in creating or using the information within the subject areas. The AAM is usually represented using IDEF0^[63]. IDEF0 is a diagramming method used to describe systems. It is a top-down (from general to specific) method. It starts with a single diagram representing the entire system and decomposes this into ever more detailed diagrams explaining subsystems. Within the AAM, information flow is shown as a number of ICOM arrows. These ICOM arrows may be Inputs, Controls, Outputs, or Mechanisms for the activities, as shown in Figure 6.1.

The fundamental principle of the IDEF0 methodology is that an Input is converted into an Output by an Activity. A Control governs when and how an activity occurs, while a Mechanism is something that is used (but not consumed) during an activity. A component manufacturer could view this as a way of representing the system whereby raw materials (the Input) are processed (the Activity) to produce a product (the Output) using some manufacturing plant (the Mechanism) which is controlled by a production schedule (the Control).

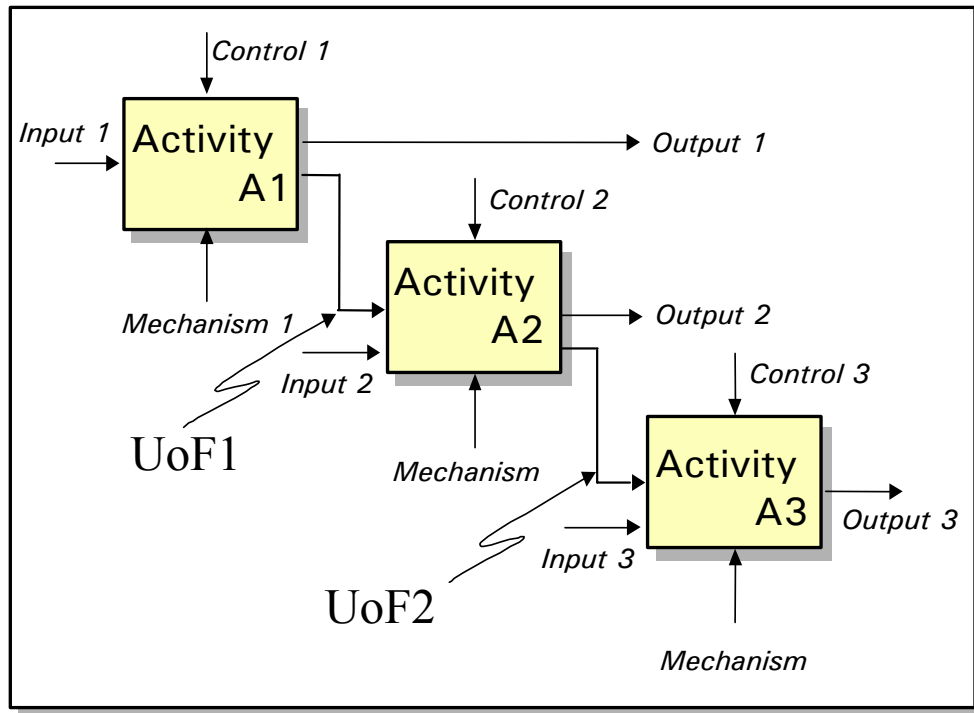


Figure 6.1 *The Application Activity Model (AAM)*

Of particular interest in the AAM are the Inputs and Outputs that represent flows of information between the activities. In the STEP AP, these inputs and outputs are known as Units of Functionality (UoF) and form the basis of the next model - the Application Reference Model (ARM).

A simplified view of the potential information flows facilitated by CIS/2 was shown in Figure 1.3. The CIS/2 AAM goes into much greater detail, decomposing all of the activities and information flows of interest, marking them as either in or out of scope of the CIS/2 Product Model.

The CIS/2 AAM is documented in the *Information Requirements*^[17]. The top-level diagram for the CIS/2 AAM is shown in Figure 6.2, while the first decomposition of the CIS/2 Activity Model is shown in Figure 6.3.

It can be seen that the scope of the CIS/2 AAM is the planning, analysis, design and manufacturing of steel framed buildings. Its purpose is to identify the creation, use and sharing of information within the domain of a Structural Engineer. The AAM is drawn from the viewpoint of a Software Engineer specifying the functional requirements of various pieces of specialist application software.

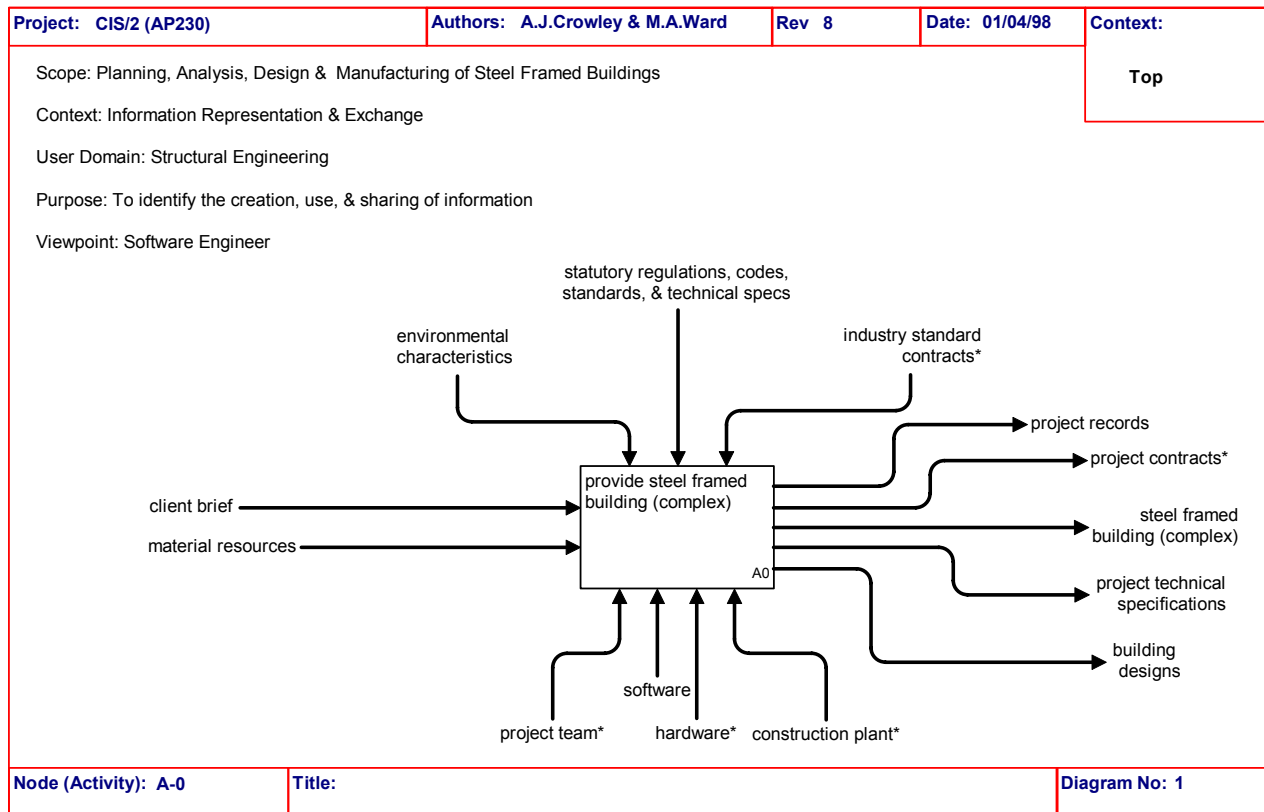


Figure 6.2 The top level diagram for the CIS/2 Activity Model

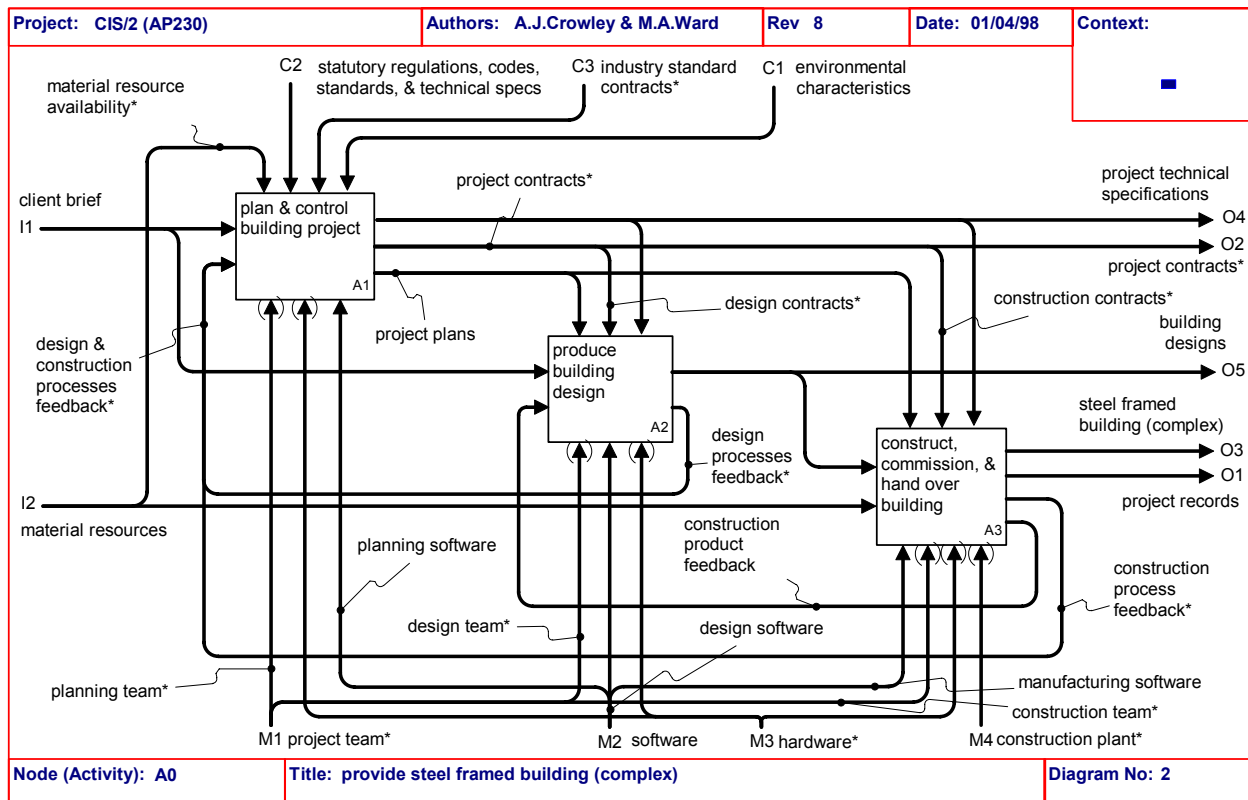


Figure 6.3 The first decomposition of the CIS/2 Activity Model

6.3.4 Conceptual Modelling: The ARM

The Application Reference Model (ARM) represents the information contained in a number of Units of Functionality (UoF) as Application Objects and Application Assertions. The ARM represents information in terms of the real world, from an application user's perspective. The Application Objects and Application Assertions are merely text statements but the ARM brings collections of these statements together and represents them graphically using IDEF1X^[21, 64] as shown in Figure 6.4.

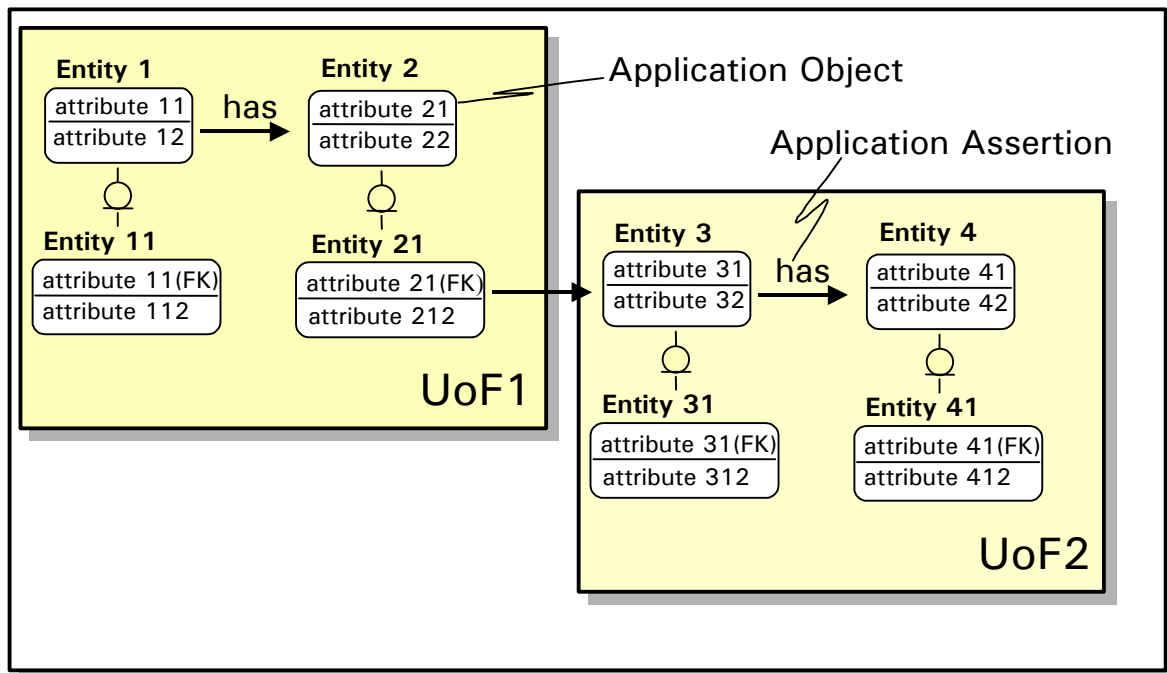


Figure 6.4 *The Application Reference Model (ARM)*

Each Application Object is an atomic element that embodies a unique (engineering) concept and contains attributes specifying the data associated with the Object. Each Application Assertion specifies the relationship between two Application Objects, the cardinality of the relationship, and the rules required for the integrity and validity of the Application Objects.

6.3.5 Modelling at the Logical Level: The AIM

It is a requirement of STEP that the user-oriented ARM is 'interpreted' using the STEP Integrated Resources (IRs) to produce a user-independent Application Interpreted Model (AIM). While the ARM is defined from a real world viewpoint using engineering-oriented terms, the AIM is defined in systems-oriented terms from a data-centric viewpoint. While CIS/2 followed the STEP methodology for developing the AAM and the ARM, the formal 'interpretation process' required by STEP for an AP was deemed to be inappropriate for the development of the Logical Product Model (LPM/5). Thus, LPM/5 was derived from the ARM, but was not strictly interpreted from it. This had the advantage of producing an implementation model that was much more compact than the equivalent AIM. It should also be noted that LPM/5 contained in a single model the scope and functionality that would have to be placed in the AIM of several APs.

6.4 Drivers for the Development of the Second Edition

The body responsible for improving and maintaining CIS/2 as an open standard is the International Technical Committee (ITC), comprising:

- The Steel Construction Institute
- Leeds University
- The American Institute of Steel Construction (AISC)
- Georgia Institute of Technology
- National Institute of Standards and Technology (NIST), and
- The primary developers of commercial CIS/2 translators.

In the 3 years since the First Edition of CIS/2 was published, CIS/2 users have raised 103 issues via the CIS/2 web site, the EDI email exploders, and AISC conferences. These were compiled and submitted for action to the ITC. Most of these have been trivial – requiring little or no action, but the 25 ‘Major Technical’ issues warranted the development of a Second Edition of CIS/2 incorporating a new product model: LPM/6. Further, second editions of the STEP Generic Resources^[44, 45, 46, 47] used in LPM/5 were published in 2000; the revised constructs contained therein are incorporated into LPM/6.

25 editions of the proposed LPM/6 EXPRESS schema were developed by the SCI and distributed to selected users – via the ITC – for testing and review. The final edition was agreed by a unanimous vote during a teleconference on the 25th February. Commercial CIS/2.1 conformant translators were demonstrated at the North American Steel Construction Conference (NASCC), on April 2-5, in Baltimore.

In summary, 124 EXPRESS constructs have been added to LPM/6 for the 2nd Edition of CIS/2. A further 112 have been modified. Many of the changes to the Logical Product Model have been made to accommodate the information requirements of MIS systems, including: drawing files, cost codes, CNC files, sequences and lots. Other changes were required to represent: design loading, cambered assemblies, pock marks, complex welds and fasteners

Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

7 GUIDE TO THE CIS/2 DOCUMENTATION

7.1 Overview

Volume 1 - Overview introduces the CIS, its scope, its origins in the CIMsteel Project, and an overview of each volume of the CIS/2 documentation. It should also be seen as an ‘executive guide’ to CIS/2, providing a backward mapping from CIS/2 to CIS/1 and a forward mapping onto AP230. It also describes some of the theory of ‘Product Modelling’ and the underpinning of CIS/2 by STEP (ISO 10303).

7.2 Implementation Guide

Volume 2 - The Implementation Guide^[15] provides a technical guide for software vendors wishing to implement the CIS. It covers some of the technology used in the development and implementation of the CIS, including the IDEF0, IDEF1X, EXPRESS, and EXPRESS-G languages, the physical file format and SDAI. It also provides two example translator programs.

CIS/2 may be implemented in various degrees of complexity, from basic file exchange through to advanced implementation in a Database Management System (DBMS). Volume 2 explains how software vendors may implement CIS/2 at each of five levels of increasing complexity. Thus, Volume 2 addresses:

1. Implementation of a CIS translator that supports CIS data exchange file operations (known as a ‘Basic Translator’).
2. Implementation of a CIS translator that supports data sharing and management that is Data Management Conformant (known as a ‘DMC-Translator’).
3. Implementation of a CIS translator that also provides for incremental data import and incremental updating (known as an ‘IDI Translator’).
4. Implementation of a CIS translator that supports data sharing and management through a CIS Product Model Repository (known as a ‘PMR-enabled Translator’).
5. Implementation of a CIS Product Model Repository (PMR).

Each of these levels is upwardly compatible such that a ‘DMC-Translator’ is able to accept all of the data produced by a ‘Basic Translator’. A certain amount of ‘downward compatibility’ is also possible such that a ‘Basic Translator’ is able to accept some of the data produced by the more complex implementations. (The meta-data used for data management by higher-level implementations is simply ignored by a ‘Basic Translator’.)

7.3 The Information Requirements

Volume 3 - The Information Requirements^[17] documents the background to CIS/2 and shows how the Logical Product Model (LPM) was derived. Volume 3 includes the ‘Application Activity Model’ and the ‘Application Reference Model’, which are required components of a STEP Application Protocol. These represent the world of structural engineering expressed in the language and terminology of Structural Engineers. The LPM, on the other hand, represents the structural engineering information in the language and terminology of the software engineer. The information requirements are

specified as a set of ‘Units of Functionality’, ‘Application Objects’, and ‘Application Assertions’, and are graphically represented in the ‘Application Reference Model’. The information requirements correspond to activities that have been identified as being within the scope of CIS/2 as defined with the Application Activity Model.

7.3.1 Application Activity Model (AAM)

The Application Activity Model has two main components:

1. IDEF0 Diagrams showing the activities, and the data flows between these activities, during the production of a ‘steel framed building complex’, and
2. A Glossary of the Activities and the ICOM arrows.

The CIS/2 Application Activity Model¹⁰ is defined in IDEF0 and captures the CIMsteel view of the processes involved in the production of a steel structure. The processes are described in terms of activities with data Inputs, Controls, Outputs, and Mechanisms (ICOM). Each stage in each of the processes is given a title and is broken down into sub-processes of increasing detail (see Figure 6.1).

The Application Activity Model is used to identify the information flows between the different types of applications commonly involved in the development cycle of a steel structure. In the Application Activity Model, the flows of information are represented by the arrows between the boxes on the IDEF0 diagrams, as shown in Figure 6.1. In the CIS, the flows of information between each stage of the process are called Units of Functionality¹¹ (UoF).

7.3.2 Units of Functionality (UoF)

The Units of Functionality present a textual description of the information requirements as derived from the ICOM arrows of the AAM, identifying Application Objects within the Application Reference Model. For example, the ‘Structural Scheme’ UoF represents a set of indicative descriptions and specifications of the structural topology and basic geometry; the structural layout; the structural behaviour of the structural elements; and some structural member sizes; for an (idealized) building (or building complex). The ‘Structural Scheme’ UoF is an output of the ‘Develop Structural Scheme’ Activity and serves to inform load assessors, structural modellers, detail designers, concept designers, architects, building services designers, and constructors.

7.3.3 Application Objects

The Application Objects present an illustrated textual description of the information requirements as derived from the UoFs, identifying the data associated with each of the Application Objects. Each Application Object is an atomic element that embodies a unique application concept and contains attributes specifying the data elements of the object. An Application Object may capture ‘high level’ concepts such as a ‘Building Complex’, or more specific concepts such as a ‘Bolt’.

¹⁰ Known as the Activity Model in CIS/1.

¹¹ Known as Local Views in CIS/1.

7.3.4 Application Assertions

The Application Assertions present a textual description of the information requirements as derived from the UoFs, identifying the relationships and constraints between Application Objects. Each Application Assertion specifies the relationship between two Application Objects, the cardinality of the relationship, and the rules required for the integrity and validity of the Application Objects. For example, the 'Part to Finite Element' Application Assertion captures the relationship between the Application Object 'Part' and the Application Object 'Finite Element'. It asserts that a 'Part' may be represented (for analysis purposes) by zero, one or many 'Finite Elements' and that a 'Finite Elements' may be used to represent zero, one or many 'Parts'. This means that there is a relationship between the two concepts 'Part' and 'Finite Element', but not a dependent one.

7.3.5 Application Reference Model (ARM)

The Application Reference Model presents a graphical representation (in IDEF1X^[21, 64]) of the Application Objects & Application Assertions.

7.4 The Logical Product Model (LPM/6)

Volume 4 - The Logical Product Model^[18] documents version 6 of the CIMsteel Logical Product Model (LPM/6), i.e. the model that is implemented by software vendors. The logical model remodels (or interprets) the information requirements to produce something that can be implemented in computer systems to represent data that represents information about aspects of the real world.

7.4.1 LPM/6 - EXPRESS Schema (Short Form)

The main body of Volume 4 contains the 'short form' of LPM/6. This provides illustrated textual descriptions of the entities of LPM/6, with their attributes and formal propositions (rules & constraints), together with references to STEP resources. In general, it does not repeat the definitions of those STEP entities that have been used from the Generic Resources. However, brief explanations of some of the generic constructs are provided to aid their understanding and use with LPM/6.

7.4.2 LPM/6 - EXPRESS Schema (Long Form)

Appendix A of Volume 4 presents the complete LPM/6 as an EXPRESS schema in its 'long form'; that is, as one single schema with all the inter-schema references resolved. The 'long form' part contains no definitions or explanations, merely an EXPRESS long form listing of the CIS/2 implementation schema.

7.4.3 LPM/6 - EXPRESS-G Diagrams

Appendix B of Volume 4 provides a graphical presentation (in EXPRESS-G) of the CIS/2 implementation schema.

7.5 Conformance Requirements

Volume 5 - The Conformance Requirements^[19] specifies what software vendors are required to do to make their applications 'CIS/2-conformant'. It specifies the Conformance Classes and details the requirements of the formal Conformance Testing

procedures, which test whether an application conforms to the CIS/2 specifications. Formal Conformance Testing is based on the procedures defined by STEP^[43] and is intended to provide software vendors with an independent ‘seal of approval’ and give the engineering end-users greater confidence when purchasing tested applications.

7.5.1 Conformance Classes (CCs)

The Conformance Classes are presented as short form EXPRESS schemas that are testable subsets of the implementation schema.

Conformance Classes provide a basis for specifying or describing the scope of a particular translator. CCs are also used in the formal Conformance Testing procedures, which test whether a translator and its application are ‘CIS/2-conformant’.

7.5.2 Conformance Requirements

The fifth volume also explains the detailed conformance requirements at each level and form of implementation taking into consideration the increasing levels of complexity and the requirements for operational documentation, error messages, and log files.

7.5.3 Conformance Testing

Volume 5 provides a description of the formal Conformance Testing procedures that demonstrate whether a specific implementation conforms to the CIS/2 specifications. It discusses how a CIS translator should be submitted for conformance testing and how it will be tested for compatibility with CIS/2 and particular Conformance Classes. It also describes how problems with implementation are dealt with, and what degrees of ‘CIS Conformance’ can result.

7.5.4 Industrial Realization

Volume 5 includes a textual description of how CIS/2 deals with national conventions and variations in the referencing of standard and manufacturers’ product items and unit systems.

Standard Identifiers

Since the CIS has to address the needs of the steel construction industry world-wide, lists of ‘standard identifiers’ for various product items that are defined in national standards are included for implementation in CIS-conformant computer aided-engineering systems.

7.5.5 Test Cases

Volume 5 also includes a number of abstract and specific test cases that will be used for testing implementations claiming conformance to the CIS/2 specifications.

7.6 Worked Examples

Volume 6 - Worked Examples^[20] document contains a number of ‘worked examples’ that are intended to aid software vendors in their development of CIS/2-conformant applications. Each worked example is provided with a usage scenario, illustrated descriptions and commented Part 21 files.

8 REFERENCE SECTION

8.1 Bibliography

1. AFNOR Z68-300: 1989
Representation externe des donne definition de produits, SET Version 89-06
Association Française de Normalisation, France, 1989
2. AUTODESK
Drawing and File Formats', pp. 529-566 in *AutoCAD Reference Manual Release 11*
Autodesk Ltd, Guilford, England, 1990
3. AYRES, R.U.
Computer Integrated Manufacturing. Vol. 1: Revolution in Progress
Chapman & Hall, London, 1991
4. BERGSMA, G., MARKOVITZ, M. & HAMBURG, S.
Structural Steel Software: Building the Future, RAM International, 1999
5. BJORK, B.-C.
Requirements and Information Structures for Building Product Data Models
Technical Research Centre of Finland, VTT Publications, Espoo, 1995
6. BOMAN, M., BUBENKO, J. AND JOHANNESSON, P.
Conceptual Modelling (2nd edition)
Dept. of Computer and Systems sciences, Stockholm University, Stockholm, 1991
7. BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., & MALER, E.
Extensible Markup Language (XML) 1.0 (Second Edition)
W3C Recommendation 6 October 2000
<http://www.w3.org/TR/REC-xml>
8. BRODIE, M., MYOPOULOS, J. AND SCHMIDT, J. (EDS)
On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases and
Programming Languages
Springer-Verlag, New York, 1985
9. CAM-I INC
Product Data Definition Interface, Reports DR-84-GM-01 to -05
CAM-I Inc, Arlington, Texas, 1984
10. CIMSTEEL
Computer Integrated Manufacturing for Constructional Steelwork: Delivering the Promise
Deliverable of the Eureka CIMsteel Project
Available from The Steel Construction Institute, UK, 1997
11. COMPUTING SUPPLIERS FEDERATION
CAD/CAM Glossary
CAD/CAM Forum of the Computing Suppliers Federation
Worcester, UK, 1996
12. CROWLEY, A.J.
The Development and Implementation of a Product Model for Constructional Steelwork
PhD Thesis, University of Leeds, UK, 1998
13. CROWLEY, A.J.
CIS/2: US Case Studies, pg. 20 in *New Steel Construction*, November/December 2002,
BCSA/SCI, UK, 2002
14. CROWLEY, A.J.
CIS/2 Progress, pg. 55 in *Steel Construction Yearbook 2000*,

- McMillan-Scott, UK, 1999
15. CROWLEY, A.J.
CIS/2 Futures: The Impact of XML, pp. 63-65 in *Steel Construction Yearbook 2001*,
McMillan-Scott, UK, 2000
 16. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 2 - Implementation Guide
SCI Publication P266, The Steel Construction Institute, UK, 2000
 17. CROWLEY, A.J., WARD, M.A. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 3 - The Information Requirements
SCI Publication P267, The Steel Construction Institute, UK, (in preparation)
 18. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 4 - The Logical Product Model (LPM/6)
SCI Publication P268, The Steel Construction Institute, UK, 2000
 19. CROWLEY, A.J., SMITH, A.M. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 5 - Conformance Requirements
SCI Publication P269, The Steel Construction Institute, UK, 2000
 20. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition, Volume 6 - Worked Examples
SCI Publication P270, The Steel Construction Institute, UK, (in preparation)
 21. DACOM
Information Modelling Manual IDEF1X
D.Appleton Company Ltd, 1985
 22. DAVIS E. H. AND GOEDHART, J. L.
Integrated planning frontiers, pp. 249-276 in Oliff, M (ed); *Intelligent Manufacturing*
Menlo Park, Benjamin/Cummings, 1987
 23. DIN 60301: 1986
Verband des Automobilindustrie Flächen Schnittstelle (VDA-FS), Industrielle Automation -
Rechnergestützter Konstruieren: Format zum Austausch geometrischer informationei
Deutsches Institut für Normung, Germany, 1986
 24. EASTMAN, C., SACKS, R. & LEE, G.
Strategies for Realizing the Benefits of 3D Integrated Modeling of Buildings for the AEC
Industry, in NIST Special Publication 989; *International Symposium on Automation and
Robotics in Construction, 19th (ISARC). Proceedings*. National Institute of Standards and
Technology, Gaithersburg, Maryland. September 23-25, 2002, 9-14 pp, 2002
 25. GREINER, R. & SALZGEBER, G.
A Structured Approach to Design, Analysis and Code Check of Constructional Steelwork:
Engineering Part, Version 1 (Deliverable of the EUREKA 130 CIMsteel Overall Design
and Analysis Working Group)
Technical University of Graz, Austria, 1996
 26. HAMBURG, S.E.
AISC Endorse CIMsteel Standard, pp. 22-28 in *Modern Steel Construction*, Vol. 39, No. 1
American Institute of Steel Construction, Inc., Chicago, USA, January 1999
 27. HAMBURG, S.E. & HOLLAND, M.V.
Leaping ahead with EDI, pp. 42-48 in *Modern Steel Construction*, Vol. 39, No. 2
American Institute of Steel Construction, Inc., Chicago, USA, February 1999

28. IAI, 1997A
Industry Foundation Classes - Release 1.5: IFC End User Guide
International Alliance for Interoperability, Washington, DC, November 1997
29. IAI, 1997B
Industry Foundation Classes - Release 1.5: IFC Specifications Development Guide
International Alliance for Interoperability, Washington, DC, November 1997
30. IAI, 1997C
Industry Foundation Classes - Release 1.5: IFC Object Model Architecture Guide
International Alliance for Interoperability, Washington, DC, November 1997
31. IAI, 1997D
Industry Foundation Classes - Release 1.5: Specifications
Volume 1 AEC/FM Processes supported by IFC
International Alliance for Interoperability, Washington, DC, November 1997
32. IAI, 1997E
Industry Foundation Classes - Release 1.5: Specifications
Volume 2 IFC Object Model Guide
International Alliance for Interoperability, Washington, DC, November 1997
33. IAI, 1997F
Industry Foundation Classes - Release 1.5: Specifications
Volume 3 IFC Object Model Reference
International Alliance for Interoperability, Washington, DC, November 1997
34. IAI, 1997G
Industry Foundation Classes - Release 1.5: Specifications
Volume 4 IFC Software Implementation Certification Guide
International Alliance for Interoperability, Washington, DC, November 1997
35. ILLINGWORTH, V.
Oxford Dictionary of Computing
Oxford University Press, Oxford, England, 1997
36. ISO 8879: 1986
Information processing - Text and office systems - Standard Generalized Markup Language (SGML), ISO, Geneva, Switzerland, 1986
37. ISO/DTR 9007: 1985
Concepts and terminology for the conceptual schema and information base
ISO / TC97, 1985
38. ISO 10303-1: 1994
Industrial automation systems - Product data representation and exchange
Part 1: Overview and Fundamental Principles
ISO/IEC, Geneva, Switzerland, 1994
39. ISO 10303-11: 1994
Industrial automation systems - Product data representation and exchange
Part 11: The EXPRESS Language Reference Manual
ISO/IEC, Geneva, Switzerland, 1994
40. ISO 10303-21: 2002
Industrial automation systems - Product data representation and exchange
Part 21: Clear text encoding of the exchange structure
ISO/IEC, Geneva, Switzerland, 2002
(Incorporates Technical Corrigendum 1 published 1996-08-15)
41. ISO 10303-22: 1998
Industrial automation systems - Product data representation and exchange
Part 22: Standard Data Access Interface

- ISO/IEC, Geneva, Switzerland, 1998
42. ISO/TS 10303-28: 2002(E)
Industrial automation systems - Product data representation and exchange
Part 28: Implementation methods: XML representations of EXPRESS schemas and data
ISO/IEC, Geneva, Switzerland, 2002
 43. ISO DIS 10303-31: 1992
Industrial automation systems - Product data representation and exchange
Part 31: Conformance Testing Methodology & Framework: General Concepts
ISO/IEC, Geneva, Switzerland, 1992
 44. ISO 10303-41: 2000
Industrial automation systems - Product data representation and exchange
Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
 45. ISO 10303-42: 2000
Industrial automation systems - Product data representation and exchange
Part 42: Integrated Generic Resources: Geometric & Topological Representation
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
 46. ISO 10303-43: 2000
Industrial automation systems - Product data representation and exchange
Part 43: Integrated Generic Resources: Representation Structures
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
 47. ISO 10303-44: 2000
Industrial automation systems - Product data representation and exchange
Part 44: Integrated Generic Resources: Product Structure Configuration
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
 48. ISO 10303-45: 1998
Industrial automation systems - Product data representation and exchange
Part 45: Integrated Generic Resources: Materials
ISO/IEC, Geneva, Switzerland, 1998
 49. ISO WD 10303-106: 1997
Industrial automation systems - Product data representation and exchange
Part 106: Integrated Application Resources: Building Construction Core Model
ISO/IEC, Geneva, Switzerland, 1997
 50. ISO 10303-201: 1994
Industrial automation systems - Product data representation and exchange
Part 201: Application protocol: Explicit Draughting
ISO/IEC, Geneva, Switzerland, 1994
 51. ISO 10303-203: 1994
Industrial automation systems - Product data representation and exchange
Part 203: Application protocol: Configuration Controlled Design
ISO/IEC, Geneva, Switzerland, 1994
 52. ISO WD 10303-221 (N194): 1995
Industrial automation systems - Product data representation and exchange
Part 221: Application protocol: Process Plant Functional Data and its Schematic
Representation, ISO/IEC, Geneva, Switzerland, 1995
 53. ISO FDIS 10303-225 (N710): 1997
Industrial automation systems - Product data representation and exchange
Part 225: Application protocol: Building Elements using Explicit Shape Representation
ISO/IEC, Geneva, Switzerland, 1997

54. ISO WD 10303-227 (N367): 1995
Industrial automation systems - Product data representation and exchange
Part 227: Application protocol: Plant Spatial Configuration
ISO/IEC, Geneva, Switzerland, 1995
55. ISO/WD 10303-230 (N551): 1996
Industrial automation systems - Product data representation and exchange
Part 230: Application protocol: Building Structural Frame: Steelwork,
ISO TC184/SC4/WG3 (T12), 1996
56. ISO/IEC 14772-1:1997
Information technology -- Computer graphics and image processing -- The Virtual Reality
Modeling Language (VRML) -- Part 1: Functional specification and UTF-8 encoding
ISO/IEC, Geneva, Switzerland, 1997
(<http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm>)
57. KERR, R.
Knowledge-Based manufacturing management: applications of artificial intelligence to the
effective management of manufacturing companies
Adison-Wesley, Singapore, 1991
58. KHANZODE, A.R. & FISCHER M.A.
*Potential Savings from Standardized Electronic Information Exchange: A Case Study for the
Steel Structure of a Medical Office Building*, Technical Report 121: Stanford University -
Civil & Environmental Engineering, Stanford, CA, July, 2000.
59. LIPMAN, R.
Visualising CIS Data, pg. 19 in *New Steel Construction*, September/October 2001,
BCSA/SCI, UK, 2001
60. LIPMAN, R. & REED, K.A.
Using VRML In Construction Industry Applications, in *Web3D - VRML 2000 Symposium*,
Monterey, CA, February 21-24, 2000
61. LOS, R. & STORER, G.
Taking Control of the Business Process, pp. 57-68 in *Competitive Deployment of Product
Data Technology*, Proc. 3rd European Conf. on Product Data Technology, 15-16th April
1997, Sophia Antipolis, France
QMS, Sandhurst, UK, 1997
62. MCWILLIAMS, R.G.
Project Toyota, pp 45-48 in *Modern Steel Construction*, Vol. 39, No. 8
American Institute of Steel Construction, Inc., Chicago, USA, August 1999
63. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
IDEF0 (ICAM Definition Language 0)
Federal Information Processing Standards Publication 183
Integration Definition for Function Modeling (IDEF0)
FIPS PUB183, NIST, 1993
64. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
IDEF1X (ICAM Definition Language 1 Extended)
Federal Information Processing Standards Publication 184
Integration Definition for Information Modeling (IDEF1X)
FIPS PUB184, NIST, 1993
65. PARTRIDGE, S.
Fabrication Management Software, pp. 55-57 in *Steel Construction Yearbook 2001*,
McMillan-Scott, UK, 2000
66. POST, N.
Steel Sector Plans Paradigm Shift, *Engineering News Record*,
McGraw Hill Construction, April 2003

67. REED, K.A.
The Role of The CIMsteel Integration Standards in Automating the Erection and Surveying of Structural Steelwork, in NIST Special Publication 989; *International Symposium on Automation and Robotics in Construction, 19th (ISARC). Proceedings*. National Institute of Standards and Technology, Gaithersburg, Maryland. September 23-25, 2002, 9-14 pp, 2002
68. REED, K. A., HARROD, D., AND CONROY, W.
The Initial Graphics Exchange Specification, Version 5.0
National Institute of Standards and Technology (NIST)
Interagency report 4412, September 1990
69. REMBOLD, U., NNAJI, B.O. & STORR, A.
Computer Integrated Manufacturing and Engineering
Addison-Wesley, England, 1993
70. RESCHKE, R. & THOMSON, A.
The PIPPIN Data Warehouse, pp. 155-166 in *Competitive Deployment of Product Data Technology*, Proc. 3rd European Conf. on Product Data Technology, 15-16th April 1997, Sophia Antipolis, France
QMS, Sandhurst, UK, 1997
71. ROBINSON, C.
Information Flow with 3D Product Modelling, pp. 45-49 in *Steel Construction Yearbook 2001*, McMillan-Scott, UK, 2000
72. SCHENCK, D. & WILSON, P.
Information Modeling the EXPRESS Way
Oxford University Press, Oxford, 1994
73. STEMMER, M.
Dynamic Exchange, in *Modern Steel Construction*, Vol. ??, No. 1, January 2003
American Institute of Steel Construction, Inc., Chicago, USA, January 2003
74. STROUD R.G.
Electronic Data Interchange Liability Issues, in *Modern Steel Construction*, Vol. ??, No. 3, March 1998, American Institute of Steel Construction, Inc., Chicago, USA, March 2003
75. WATSON, A.S.
CAD Data Exchange in Construction, pp. 955-969 in *Proc. Instn. Civ. Engrs.*, Part 1, 88
Thomas Telford, London, December. 1990
76. WEST, M.
Managing Data Quality, Report No IC92-124
Shell, The Hague, NL, 1993

8.2 Abbreviations

AAM	Application Activity Model
AEC	Architecture, Engineering & Construction
AFNOR	Association Francais de Normalisation; the national standards organization in France.
ANSI	American National Standards Institute
AP	Application Protocol
API	Application Programming Interface
ARM	Application Reference Model
ASCII	American Standard Code for Information Interchange
ASN1	Abstract Syntax Notation 1
ASTM	American Society for Testing and Materials
BSI	British Standards Institution; the national standards organization in the UK
CAD	Computer Aided Draughting
CADCAM	Computer Aided Design - Computer Aided Manufacturing
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CASE	Computer Aided Software/Systems Engineering
CC	Conformance Class
CIC	Computer Integrated Construction
CIM	Computer Integrated Manufacturing
CIMsteel	Computer Integrated Manufacturing for Constructional Steelwork
CIS	CIMsteel Integration Standards
DBMS	Database Management System
DDL	Data Definition Language
DIN	Deutsches Institut fur Normung; the national standards organization in Germany
DML	Data Manipulation Language
DSTV	Deutscher Stahlbau-Verband; the German Steel-construction Association
EDI	Electronic Data Interchange
GUI	Graphical User Interface
ICAM	Integrated Computer Aided Manufacturing
IDEF	ICAM Definition (Language)
IDEF1X	ICAM Definition (Language) One Extended
IEC	International Electrotechnical Committee
IGES	Initial Graphics Exchange Specification
ISO	International Standardization Organization
KBS	Knowledge Based System
LAN	Local Area Network
LPM	Logical Product Model

MIS	Management Information System
NIAM	Nijssen Information Analysis Method
NIST	National Institute of Standards and Technology
OO	Object-oriented
PDES	Product Data Exchange Specification
PDT	Product Data Technology
PM	Product Model
PMR	Product Model Repository
SDAI	Standard Data Access Interface
STEP	Standard for the Exchange of Product model data (ISO 10303)
UoD	Universe of Discourse
UoF	Unit of Functionality
WAN	Wide Area Network.

8.3 Glossary

Application

A group of one or more processes using product data [ISO 10303-1: 1994].

Application Activity Model (AAM)

A model that describes an application in terms of its processes and information flows [ISO 10303-1: 1994].

Application Context

The environment in which the integrated resources are interpreted to support the use of product data in a specific application [ISO 10303-1: 1994].

Application Interpreted Model (AIM)

An information model that uses the integrated resources necessary to satisfy the information requirements and constraints of an application reference model, within an application protocol [ISO 10303-1: 1994].

Application Object

An atomic element of an application reference model that defines a unique application concept and contains attributes specifying the data elements of the object [ISO 10303-1: 1994].

Application Programming Interface (API)

An interface that is defined in terms of a set of functions and procedures, and enables a program to gain access to facilities within an application.^[35]

Application Protocol (AP)

A (STEP) standard that specifies an application interpreted model satisfying the scope and information requirements for a specific application. [ISO 10303-1: 1994].
(Note - This definition differs from the definition used in OSI standards)

Application Reference Model (ARM)

An information model that describes the information requirements and constraints of a specific application context [ISO 10303-1: 1994].

Application resource

An integrated resource whose contents are related to a group of application contexts [ISO 10303-1: 1994].

Conformance testing

The testing of a candidate product for the existence of specific characteristics required by a standard in order to determine the extent to which that product is a conforming implementation. [ISO 10303-31: 1994]

Computer Integrated Manufacturing (CIM)

The application of computer science technology to the enterprise of manufacturing in order to provide the right information to the right place at the right time, which enables the achievement of its product, process, and business goals.^[3]

CIM combines, in one system, the activities of CAD (computer-aided design), CAP (computer-aided planning), CAM (computer-aided manufacturing), CAQ (computer-aided quality control), and PP&C (production, planning and control).^[69]

Concurrent Engineering

One of several names given to the process of parallel design and manufacture, which refers to the philosophy of reducing total manufacturing time by having several processes overlapping as opposed to sequential execution. Also known as simultaneous engineering.^[11]

Database Management System (DBMS)

Software specifically designed to store, manipulate and access information held on a computer. Key features are the use of schemas to define the structure of the data held and the provision of software interfaces to facilitate (and standardize) interaction with the stored information. In a computer database system, all data is held in some physical form known as 'database'. Access to the data is controlled by software known as the DBMS; it is the DBMS that 'knows' where and in what physical format the data is stored. The DBMS itself is accessed - for adding, updating, querying and deleting data - by a database language such as SQL.

Electronic Data Interchange (EDI)

The electronic exchange of structured and normalized data between the computer applications of parties involved in a trade transaction. (It is really aimed at the exchange of documents.) In EDI Transactions, one party requests a service and the other party offers the service and receives payment for the service. (This also means that both parties have a strong relationship supported by a business agreement.) In general, a transaction is not a single message sent from one party to another, but will consist of a series of messages (order, change, order, accept, reject). This means that from start to finish, a transaction can be in different intermediate states. EDI messages are structured according to strictly defined rules such that a receiving application is able to interpret the message and perform the necessary sequel actions, without human interference. The structure is flat, i.e. no segments within segments.

Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)

OSI Application Standard ISO 9735 - to define the interchange of data in a variety of business environments. The EDIFACT syntax was designed to describe the contents of documents. For this reason, the building blocks of EDIFACT are data segments and data elements.

EXPRESS

The data definition language of STEP.

EXPRESS is an object-flavoured information model specification language that was initially developed in order to enable the writing of formal information models describing mechanical products^[72]. EXPRESS was created specifically to define the information that is consumed or produced throughout a product's life cycle.

EXPRESS-G

A formal graphical notation for the display of data specifications defined in the EXPRESS language. The notation only supports a subset of the EXPRESS language.

Graphical user Interface (GUI)

The mechanism by which the user interactively accesses the tools provided by the software. Modern user interfaces use hierarchical menu icons and mouse input to access the software.

Information Model

A formal model of a bounded set of facts, concepts or instructions to meet a specified requirement [ISO 10303-1: 1994].

Integrated Resource

A STEP Part that defines a group of resource constructs used as the basis for product data [ISO 10303-1: 1994].

Information Resource Dictionary (IRD)

A shareable repository for definitions of the information relevant to all or part of an enterprise (data relevant to the organization; computerized and non-computerized processes for handling the data; the physical hardware environment).

Interpretation

The process of adapting a resource construct from the integrated resources to satisfy a requirement of an application protocol. This may involve the addition of restrictions on attributes, the addition of constraints, the addition of relationships among resource constructs and application constructs, or all of the above [ISO 10303-1: 1994].

Language

A set of characters, conventions and rules that are used to convey information. The three aspects of language are pragmatics, semantics, and syntax.

Product Information Model

An information model which provides an abstract description of facts, concepts and instructions about a product [ISO 10303-1: 1994].

Schema

Those definitions which describe the concept of the data and the relationship between the various elements or components of the data.

A collection of items forming part or all of a model [ISO 10303-11: 1994].

A definition of data structure^[21].

Semantics

The meanings of words and sentences in a language, or of constructs in a model.

Structured Query Language (SQL)

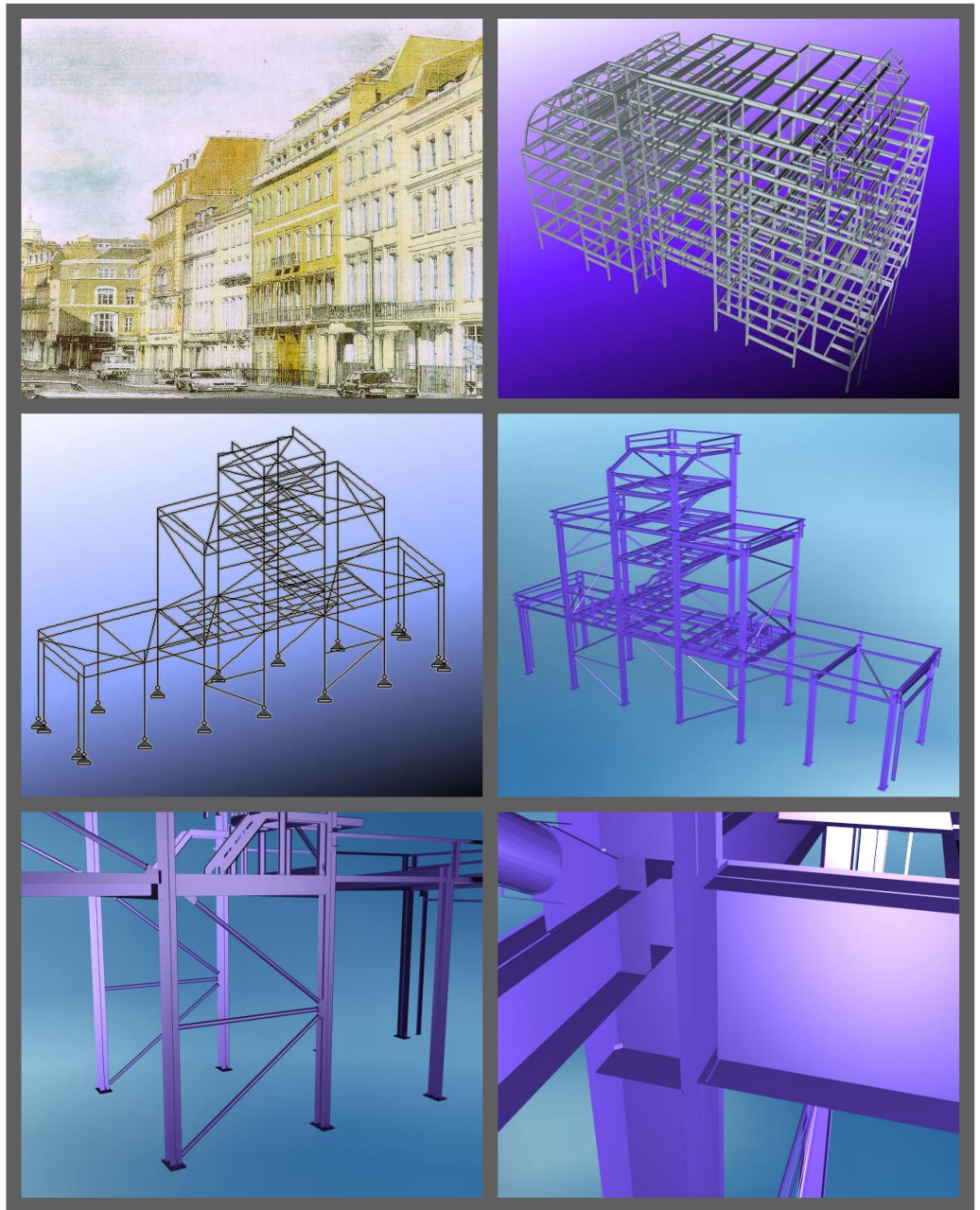
OSI Application Standard ISO 9075 - Standard database access language. It is a powerful data manipulation language (DML), based on relational ideas. It is a powerful way to create portable database systems over hardware and software environments. It is designed to be used interactively, and embedded within procedural languages. It is not a database management system (DBMS), a procedural programming language, nor a fourth generation language (4GL). (The use of the word "Standard" is a misnomer, as there are many different implementations of SQL, none of which is identical to the other, and most of the current SQL implementations are not fully relational.)

Syntax

Grammar, a set of rules for forming meaningful phrases and sentences from words in a vocabulary.

Unit of Functionality (UoF)

A collection of application objects and their relationships that define one or more concepts within the application context such that removal of any component would render the concepts incomplete or ambiguous [ISO 10303-1: 1994].



The Steel Construction Institute

CIMsteel Integration Standards Release 2: Second Edition

Volume 2 Implementation Guide

CIS/2.1



The Steel Construction Institute

The Steel Construction Institute is an independent, technical, member-based organization, dedicated to the development and promotion of the effective use of steel in construction. Founded in 1986, the SCI now has over 600 corporate members in 37 countries. The SCI's research and development activities cover many aspects of steel construction including multi-storey construction, industrial buildings, light gauge steel framing systems, stainless steel, fire engineering, bridge and civil engineering, offshore and hazard engineering, structural analysis systems, environmental engineering and information technology. The results of these projects are fed back to SCI members via a comprehensive package of benefits. Membership is open to all organizations and individuals that are concerned with the use of steel in construction. Members include designers, contractors, suppliers, fabricators, academics and government departments in the United Kingdom, elsewhere in Europe and in countries around the world. The SCI is financed by subscriptions from its members, revenue from research contracts and consultancy services, publication sales and course fees.

A Membership Information Pack is available free on request from:

The Membership and Development Manager, The Steel Construction Institute, Silwood Park, Ascot, Berkshire, SL5 7QN, UK.

Telephone: +44 (0)1344 623345, Fax: +44 (0)1344 622944.



School of Civil Engineering

The School of Civil Engineering in the University of Leeds is the largest in the United Kingdom, and has excellent contacts with the construction industry and professional institutions. Its staff members have a wide range of experience in the practice of engineering, planning, architecture, design, construction and management. The School is committed to Quality Assurance of its teaching. Contacts with industry include an Advisory Committee of senior engineers and architects in practice, an industrial tutor scheme for undergraduates involvement of practising specialists in student's projects, and lectures by eminent engineers, architects and landscape architects. Industry also participates by giving sponsorships, prizes, providing vacation work and welcoming its graduates on completion of their degree. The emphasis on multi-discipline work, design projects, communication skills and teamwork seems to make Leeds graduates particularly useful.

The Computer Aided Engineering (CAE) group is a multi-disciplinary research group concerned with the application of Information Technologies to the overall engineering process. The group gained an international reputation through its innovative work applying the concepts of product modelling to achieve a more efficient information flow in the construction sector.

For details of current research or a prospectus, please contact:

The School Secretary, School of Civil Engineering, University of Leeds, Leeds, LS2 9JT, UK. Telephone: +44 (0)113 343 2248, Fax: +44 (0)113 343 2265.

CIMsteel Integration Standards

Release 2: Second Edition

Volume 2 – Implementation Guide

A J Crowley BEng, PhD

A S Watson BTech, PhD, CEng, MICE

Published by:

The Steel Construction Institute
Silwood Park
Ascot
Berkshire
SL5 7QN

Tel: 01344 623345
Fax: 01344 622944
URL: <http://www.steel-sci.org/>

In association with:

Computer Aided Engineering Group
School of Civil Engineering
The University of Leeds
Leeds
LS2 9JT

Tel: 0113 343 2282
Fax: 0113 343 2265
URL: <http://www.leeds.ac.uk/civil/>

This publication is derived from the deliverables of the CIMsteel Project

© 2000, 2003 The University of Leeds

© 2000, 2003 The Steel Construction Institute

The University of Leeds and the Steel Construction Institute wish to acknowledge the valuable contribution from other CIMsteel Collaborators, and from other parties, in the generation of this material.

Apart from any fair dealing for the purposes of research or private study or criticism or review, as permitted under the Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the UK Copyright Licensing Agency, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organisation outside the UK.

Enquiries concerning reproduction outside the terms stated here should be sent to the publishers, The Steel Construction Institute, at the address given on the title page.

Although care has been taken to ensure, to the best of our knowledge, that all data and information contained herein are accurate to the extent that they relate to either matters of fact or accepted practice or matters of opinion at the time of publication, The Steel Construction Institute, The University of Leeds, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data and/or information or any loss or damage arising from or related to their use.

Publications supplied to the Members of the Institute at a discount are not for resale by them.

Publication Number: SCI-P-266

ISBN 1 85942 100 8 (1st Edition)

British Library Cataloguing-in-Publication Data. (1st Edition)

A catalogue record for this book is available from the British Library. (1st Edition)

Front cover images have been reproduced by kind permission of Whitby Bird & Partners, Rowen Structures Ltd, Taywood Engineering Ltd, and Philip Quantrill (Structural Engineers) Ltd.

This publication is supported by a companion web site at <http://www.cis2.org/>

 [®] is a Registered Trademark

FOREWORD

2nd Edition

This publication is one of a series of SCI publications that document the Second Edition of the Second release of the CIMsteel Integration Standards (CIS/2.1). This publication provides a guide for those wishing to implement the CIS/2, and covers some of the technology used in the development and implementation of the CIS, including the IDEF0, IDEF1X, EXPRESS, and EXPRESS-G languages, STEP data exchange files and SDAI. It also provides some example translator programs.

This Second Edition should be seen as a ‘point release’, rather than a whole ‘new’ release of the CIMsteel Integration Standards. Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

The development of the second edition was coordinated by the CIS/2 International Technical Committee (ITC); the body responsible for improving and maintaining CIS/2 as an open standard. Chaired by Chuck Eastman, the ITC comprises The Steel Construction Institute and Leeds University (the joint custodians of CIS/2), together with The American Institute of Steel Construction (AISC), Georgia Institute of Technology and the National Institute of Standards and Technology (NIST). The successful deployment of CIS/2 technology over the past years has been due to the efforts of software vendors who have developed commercially viable CIS/2 translators. Consequently, several of these companies are represented on the ITC, including: Bentley, CSC (UK) Ltd, CSI, Design Data, Fabtrol, Intergraph, RAM International, and Tekla. The development of the second edition was sponsored by the American Institute of Steel Construction (AISC).

Previous Editions

The first release of the CIMsteel Integration Standards (CIS/1) and the draft release of CIS/2 were deliverables of the Pan-European Eureka EU130 CIMsteel Project. Over a ten-year period, this extensive research and development project involved seventy organizations in eight countries; representing a wide cross section of the constructional steelwork industry, including designers, fabricators, software vendors, universities, research and trade organizations. In addition to the contributions of the individual collaborators, the CIMsteel project also received financial support from:

- the Forschungsförderungsfonds für die gewerbliche Wirtschaft (FFF) in Austria
- the Erhvervsfremme Styrelsen, Industrie-ministeriet in Denmark
- the Ministère de l'Industrie, Département Communication et Commerce Extérieur in France
- the Istituto Mobiliare Italiano in Italy
- the Senter Den Haag in The Netherlands
- the Department of Trade and Industry in the United Kingdom

The principal authors of this publication are Andrew Crowley of the SCI and Alastair Watson of the University of Leeds. The development of this publication was funded by the SCI, the University of Leeds and Corus Group plc (formerly British Steel plc). The authors would like to thank their (former) colleagues in the Computer Aided Engineering (CAE) Group within the School of Civil Engineering at the University of Leeds for their work on the CIMsteel Project; in particular Steven Bennett, Dimitris Christodoulakis, Paul Hirst, Nashett El-Kaddah, George Kamparis, Gareth Knowles, Peter Riley, Alan Smith, and Mike Ward. The authors would also like to acknowledge the valuable outputs of the CIMsteel 'Overall Design and Analysis Working Group'. This group was led by Richard Greiner of the Technical University of Graz, Austria, and included representatives from Leeds and Nottingham Universities, QSE, CSC, SCI, Ove Arup, Taywood from the UK, Ramboll (Denmark), TDV (Austria), CTICM (France), Sidercad (Italy) and FCSA (Finland).

The CIS are the result of considerable efforts by many persons representing CIMsteel collaborators, associate collaborators and other interested parties. The principal developers of the CIS were Leeds University & SCI (UK), CTICM & LGCH (France), TNO (Netherlands), Ramboll (Denmark), Italsiel & Sidercad (Italy). Inputs from beyond Europe, particularly from the USA and Japan are acknowledged. The issues raised by the international review team of AP230 have been invaluable in the development of CIS/2.

AISC endorsement of CIS/2

CIS/2 has been endorsed by the American Institute of Steel Construction (AISC) as the standard for the electronic exchange of structural steel project information for the North American structural steel design and construction industry.^[13] During 1998, the Electronic Data Interchange (EDI) Review Team of the AISC evaluated data transfer standards with a view to adopting one. On December 7th 1998, their recommendation of CIS/2 was approved by the AISC Board of Directors as part of the AISC Business Plan for Standardizing the Electronic Exchange of Structural Steel Project Information. Phase I of the AISC Business Plan (now completed) included the public endorsement of CIS/2 as the standard for the electronic exchange of structural steel project information for the entire U.S. structural steel design and construction industry, as well as the recommendation that it be adopted as an international standard. Phase II of the AISC Business Plan includes several activities that will promote and support CIS/2 and its implementation.

AISC, headquartered in Chicago, is a not-for-profit organization established in 1921 to serve the structural steel industry in the United States. The organization's mission is to promote the use of structural steel through research activities, market development, education, codes and specifications, technical assistance, quality certification and standardization. AISC maintains the specification for the design of structural steel framing in the U.S. It has a long tradition of more than 75 years of providing assurance and service to the steel construction industry by providing reliable information.

For further details, please contact:

Director of Information Technology,
American Institute of Steel Construction,
One East Wacker Drive, Suite 3100, Chicago, IL 60601-2001, USA.
Telephone: +1 312 670 5413, Fax: +1 312 670 5403

CONTENTS

	Page No.
FOREWORD	III
2 nd Edition	iii
Previous Editions	iii
AISC endorsement of CIS/2	iv
CONTENTS	V
List of Figures	viii
List of Tables	ix
SUMMARY	X
1 INTRODUCTION	1
1.1 Types of CIS implementation	1
1.2 Levels of STEP implementation	2
2 STEP IMPLEMENTATION LEVEL 1	5
2.1 The role of CIS translators	5
2.2 Components required for file exchange	6
3 STEP IMPLEMENTATION LEVEL 2	8
4 STEP IMPLEMENTATION LEVEL 3	9
5 BASIC CIS TRANSLATORS	11
5.1 Developing a Basic CIS Translator	11
5.2 The procedure for translator development	13
6 DMC TRANSLATORS	18
6.1 ‘DMC Translators’ and ‘DMC Applications’	18
6.2 The purpose of data management	18
6.3 Data Management and CIS/2	18
6.4 The meta-data in LPM/6	19
6.5 Data management in practice	24
6.6 Developing a DMC Translator	26
7 IDI TRANSLATORS	28
7.1 What is ‘Incremental Data Import’?	28
7.2 Different types of ‘Incremental Data Import’	28
7.3 Developing IDI Translators	30
7.4 The procedure for Incremental Data Import	31
8 PMR-ENABLED TRANSLATORS	33

8.1	Developing PMR-Enabled Translators	33
9	PRODUCT MODEL REPOSITORIES	35
9.1	Developing a PMR	35
9.2	Types of PMR User	35
9.3	The components of the PMR	36
9.4	Using the PMR	42
9.5	Using PMRs together	44
10	THE 'COMPARATOR'	45
10.1	The tasks of the comparator	45
10.2	The 'Comparator Logic'	47
10.3	Comparing attribute values	52
11	REFERENCES	67
APPENDIX A	IMPLEMENTATION TOOLS	71
A.1	Modelling tools	71
A.2	Implementation tools	71
A.3	STEP file parsers	72
APPENDIX B	IDEFO	73
B.1	The IDEFO notation	73
APPENDIX C	IDEF1X	76
C.1	Entities	76
C.2	Attributes	76
C.3	Relationships	77
C.4	Cardinality	77
C.5	Categorisation Relationships	78
APPENDIX D	EXPRESS	79
D.1	Summary of the EXPRESS Language	79
D.2	Defining entities in EXPRESS	80
D.3	EXPRESS data types	81
D.4	Domain rules and where clauses	83
D.5	Relationships and cardinality in EXPRESS	83
APPENDIX E	EXPRESS-G	91
E.1	EXPRESS-G Notation	91
E.2	Examples of EXPRESS-G as used in LPM/6	93
APPENDIX F	STEP FILE STRUCTURE	97
F.1	The exchange structure	97
F.2	Mapping of EXPRESS data types to the exchange structure	99

F.3	The Scope structure	110
F.4	Use of STEP Part 21 for the CIS	111
APPENDIX G	STANDARD DATA ACCESS INTERFACE (SDAI)	113
G.1	Introduction	113
G.2	SDAI Overview	113
G.3	Early & Late Bindings	116
G.4	Example SDAI C programs	117

List of Figures

Figure 2.1 Components of a Typical 'CIS/2-conformant system'	1
Figure 2.2 Concurrent use of 3 forms of implementation	3
Figure 3.1 Data exchange via (Level 1) CIS Translators	5
Figure 3.2 Conceptual view of a CIS Translator for file exchange	6
Figure 4.1 Conceptual view of data exchange via Level 2 CIS Translators	8
Figure 5.1 Conceptual view of data exchange via Level 3 CIS Translators	9
Figure 6.1 An example of an export mapping table	14
Figure 6.2 Example of an import mapping table	14
Figure 6.3 Conceptual view of CIS translator development	17
Figure 7.1 Assigning Meta-data to data in LPM/6	19
Figure 7.2 Assigning the 'who' and the 'when' of the Meta-data	22
Figure 7.3 Advanced data management constructs	23
Figure 7.4 Conceptual view of a DMC Translator	26
Figure 8.1 Data exchange via IDI Translators ('master-slave' scenario)	29
Figure 8.2 Data exchange via IDI Translators ('feedback' scenario)	29
Figure 8.3 Conceptual view of an IDI Translator	31
Figure 9.1 Example interface for a PMR-Enabled Translator	33
Figure 9.2 Conceptual view of PMR-Enabled Translators	34
Figure 10.1 Conceptual view of a PMR	35
Figure 10.10 PMR in 'integrated' mode	43
Figure 10.2 Example PMR logon dialogue box	37
Figure 10.3 Example PMR Main dialogue box	37
Figure 10.4 Example PMR Project Management dialogue box	38
Figure 10.5 Example PMR User Management dialogue box	39
Figure 10.6 Example PMR Data Management dialogue box	40
Figure 10.7 Example dialogue box for updating models	41
Figure 10.8 Example dialogue box for locking and unlocking models	41
Figure 10.9 PMR in 'stand-alone' mode	42
Figure 11.1 Flowchart for the logic of comparator operations (without history)	49
Figure 11.2 Flowchart for the logic of comparator operations (with history)	50
Figure B.1 The Basics of IDEFO	73
Figure B.2 Decomposing a high level activity	74
Figure B.3 The top level diagram for the CIS/2 Activity Model	74
Figure B.4 The first decomposition of the CIS/2 Activity Model	75
Figure C.1 Independent and Dependent Entities in IDEF1X	76

Figure C.2 Identifying and Non-Identifying Relationships in IDEF1X	77
Figure C.3 Incomplete & Complete Categorization Relationships in IDEF1X	78
Figure D.1 Representation of a ‘one-to-many’ relationship	83
Figure D.2 Representation of a “many-to-many” relationship	84
Figure E.1 Representing Simple and defined data types in EXPRESS-G	93
Figure E.2 Representing Entity data types in EXPRESS-G	93
Figure E.3 Representing SUPERTYPEs and SUBTYPEs in EXPRESS-G	94
Figure E.4 Representing ENUMERATION data types in EXPRESS-G	94
Figure E.5 Representing SELECT data types in EXPRESS-G (1)	95
Figure E.6 Representing SELECT data types in EXPRESS-G (2)	95
Figure E.7 Representing aggregation data types in EXPRESS-G	96
Figure E.8 Representing INVERSE clauses in EXPRESS-G	96
Figure F.1 Example entity definition represented EXPRESS-G	100
Figure F.2 Example entity definition represented EXPRESS-G	100
Figure F.3 Example entity definition represented EXPRESS-G	101
Figure F.4 Example entity definition represented EXPRESS-G	102
Figure F.5 Example entity definition represented EXPRESS-G	103
Figure F.6 Example of internal mapping represented in EXPRESS-G	107
Figure F.7 Example of external mapping represented in EXPRESS-G	108
Figure G.1 Repositories, schema instances, and SDAI-models	114
Figure G.2 Dictionary and session data	115

List of Tables

Table 2.1 Types of CIS Translator and implementation level	3
Table 11.1 Summary of the ‘Comparator’ rules	51
Table D.1 Derived dependency matrix	86
Table D.2 Understanding the ‘Nesting’ of EXPRESS Data types	88
Table E.1 Representing definitions in EXPRESS-G	91
Table E.2 References in EXPRESS-G	92
Table E.3 Representing Relationships in EXPRESS-G	92
Table G.1 Early & Late Bindings for various Programming Languages	116
Table G.2 Early Bindings vs Late Bindings	116
Table G.3: SDAI operations required for CIS/2 translators	117

SUMMARY

This publication is one of a series of SCI publications that documents the Second Edition of the Second release of the CIMsteel Integration Standards (CIS/2.1). Other documents in this series include:

- *Volume 1 - Overview*^[6]
- *Volume 3 - The Information Requirements*^[7]
- *Volume 4 - The Logical Product Model (LPM/6)*^[8]
- *Volume 5 - Conformance Requirements*^[9]
- *Volume 6 - Worked Examples*^[10]

CIS/2 is a set of formal computing specifications that allow software vendors to make their engineering applications mutually compatible. Taken as a whole, the CIS/2 documentation specifies what information may be transferred between software applications, and how that information must be structured in a repository or data exchange file.

This publication guides software vendors through the process of developing a CIS/2-conformant system. It also covers some of the technology used in the development and implementation of the CIS, including the IDEF0, IDEF1X, EXPRESS, and EXPRESS-G languages, the STEP physical files format and SDAI. It also provides some example translator programs.

CIS/2 has been developed to facilitate a more integrated method of working through the sharing and management of information within and between companies involved in the planning, design, analysis and construction of steel framed buildings and similar structures. As this new method of working is likely to evolve gradually, the standards may be implemented in various degrees of complexity, from basic file exchange through to advanced implementation in a Database Management System (DBMS).

CIS/2 substantially extends the engineering scope of CIS/1, and introduces advanced data management capabilities to enable data sharing. At its simplest, the CIS provides specifications and guidelines for the development and implementation of translators that enable the users of engineering software to export data from one application into another. However, CIS/2 also allows software vendors to support concurrent engineering via more direct mechanisms for information sharing and management. Thus, CIS/2 also provides specifications and guidelines for the development and implementation of DBMSs built around the CIS and its related technology. Such a DBMS is known as a Product Model Repository (PMR).

This Second Edition should be seen as a ‘point release’, rather than a whole ‘new’ release of the CIMsteel Integration Standards. Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

1 INTRODUCTION

This publication guides software vendors through the process of developing a CIS/2-conformant system. Typically, software vendors will be implementing CIS/2 to write export or import translators for an existing piece of application software that was specifically written for the specialist structural engineering market. Ultimately, CIS/2-conformant systems will include purpose-built database management systems written around the CIS/2 specifications such as a Product Model Repository (PMR).

As illustrated in Figure 1.1, a ‘CIS/2-conformant system’ may include the combination of a piece of specialist application software together with its ‘front end interface’, its ‘back-end operating system’ and any ‘internal’ databases.

Although this publication is primarily concerned with creation of the translator program itself, software vendors must ensure that the whole system satisfies the demands of the *Conformance Requirements*^[9], as it is the whole system that will be the subject of any formal Conformance Testing

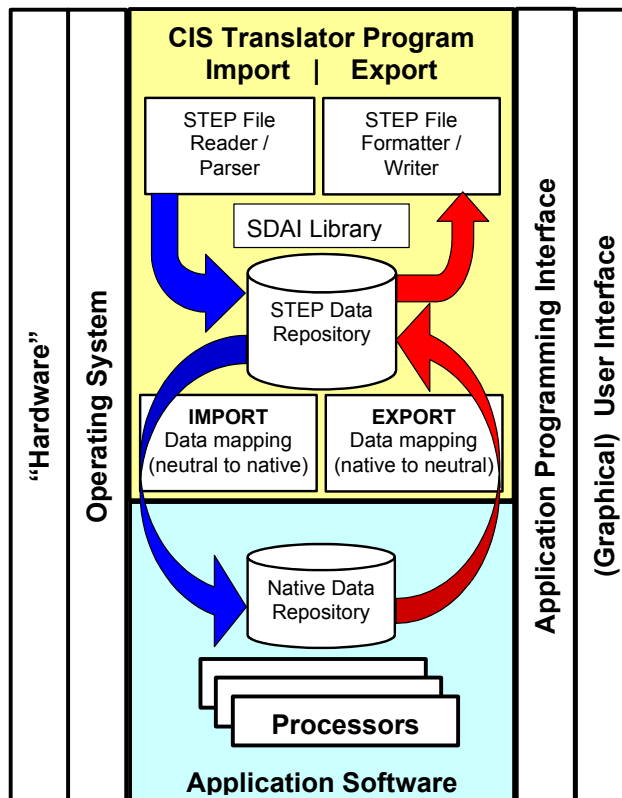


Figure 1.1 Components of a Typical ‘CIS/2-conformant system’

1.1 Types of CIS implementation

CIS/2 may be implemented in various degrees of complexity, from basic file exchange through to advanced implementation in a Database Management System (DBMS). This publication explains how software vendors may implement CIS/2 at each of five degrees of complexity. Thus, it addresses, in order of increasing complexity:

1. Implementation of a CIS translator that supports the exchange of engineering data via physical files (known as a ‘Basic CIS Translator’).
2. Implementation of a CIS translator that supports data sharing and management that is Data Management Conformant (known as a ‘DMC-Translator’).
3. Implementation of a CIS translator that also provides for incremental data import and incremental updating (known as an ‘IDI Translator’).
4. Implementation of a CIS translator that supports data sharing and management through a CIS Product Model Repository (known as a ‘PMR-enabled Translator’).
5. Implementation of a CIS Product Model Repository (PMR).

It should be noted that these five degrees of complexity allow vendors to extend the functionality of their CIS implementations incrementally. Separate Sections of this publication describe the *additional* requirements that each type of implementation brings. It should also be noted that these five degrees of complexity run *parallel* to the levels of implementation referred to by STEP. That is, a DMC-translator may use physical data exchange files as its medium of communication (see Table 1.1). Similarly, a PMR is one type of database implementation of the CIS, but not the only one.

Before developing any CIS Translator (or a PMR), software vendors are advised to read the requirements set out in Volume 5 of the CIS/2 documentation^[9].

1.2 Levels of STEP implementation

Implementation of the CIS is based upon the standards and technology emerging from STEP - the international STandard for the Exchange of Product model data^[15]. STEP aims to provide a basis for communicating product information at all stages in the product life cycle. The fundamental components of STEP are information models and implementation methods for sharing information corresponding to such models.

Initially, STEP focused primarily on the use of standardized exchange files for communicating information about a particular product. However, there has always been a perceived need to define a mechanism for sharing such information more dynamically and at a finer level of granularity using database management systems (DBMSs). Within STEP, the different types of data sharing are referred to as levels of implementation.

The four levels are:

- Level 1 data sharing by means of exchange files.
- Level 2 data sharing using a standard in-memory data format.
- Level 3 data sharing using a database management system as the means of data storage and access.
- Level 4 data storage and access via a knowledge-base system.

As it is likely to be many years before CIS/2 is implemented in an advanced knowledge-base system, only levels 1, 2 and 3 are of sufficient interest to the CIS to be discussed in this publication. When implementing either a ‘Basic’, ‘DMC’ or ‘IDI’ translator, software vendors need to satisfy the requirements of a Level 1 STEP implementation. When implementing a ‘Product Model Repository’ or a ‘PMR-enabled Translator’,

software vendors must also satisfy the requirements of a Level 3 STEP implementation. These requirements are shown in Table 1.1.

Table 1.1 *Types of CIS Translator and implementation level*

Implementation Level	Type of CIS Translator				
	<i>Basic</i>	<i>DMC</i>	<i>IDI</i>	<i>PMR-enabled</i>	<i>PMR</i>
1	●	●	●	●	●
2		○	○	○	○
3		○	○	●	●
4					○

Key:

- mandatory
- optional

It should be noted that the ‘implementation levels’ discussed above are the forms that have been identified within the STEP community and are not the only possible forms of implementation. For example, researchers are currently investigating the possibility of combining the technology of STEP – which is primarily concerned with the *representation* of data – with the technology of the World Wide Web – which is primarily concerned with the *presentation* of data. It is likely that XML^[3] (eXtensible Markup Language) will become an accepted alternative to STEP Part 21^[17] as an implementation form of an EXPRESS schema^[23].

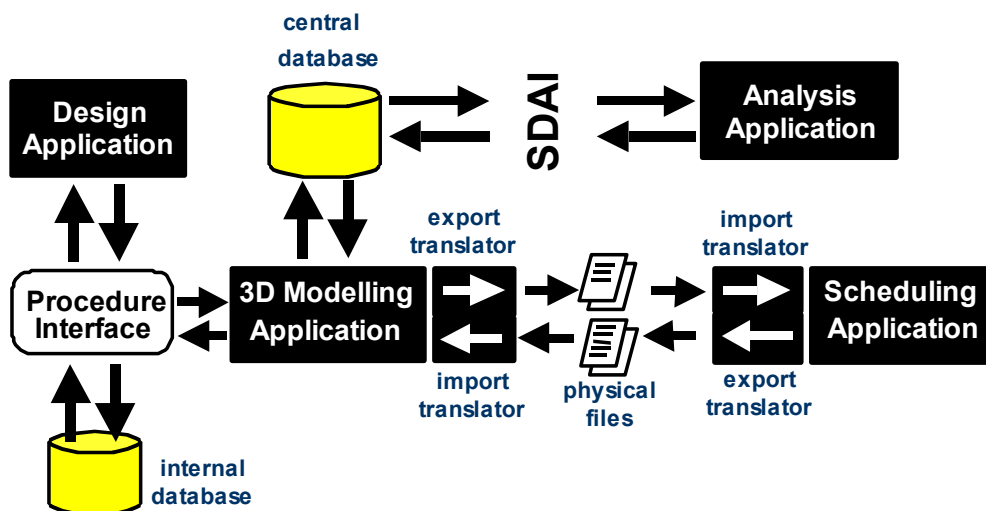


Figure 1.2 *Concurrent use of 3 forms of implementation*

It should also be noted that the forms of implementation are not mutually exclusive. Indeed, it is envisaged that CIS/2 will be implemented in different forms to suit the different needs of the applications and their users. A possible scenario is shown in Figure 1.2. Here, three forms of implementation are used concurrently, based around a 3D CAD / modelling system. The database of the modelling system is freely accessible,

and thus acts as a central database. The system exports and imports physical files, and calls procedures. Upon request from the user, the procedural interface calls procedures that activate applications to manipulate information that has been captured by other CIS-compatible applications.

The underlying assumption when sharing data using STEP is that the data in question corresponds to a standardized and integrated schema (e.g. the Logical Product Model). For the exchange of data using a neutral file format, this correspondence to a standardized schema represents most of the technical complexity. However, sharing data directly through a database management system requires more complex capabilities, and the following considerations arise:

- i. How to access the data
- ii. How to control access to the data
- iii. How to locate data in a shared and distributed system
- iv. How to limit access to data in such a way that a remote system would be able to access only that data to which has been given rights and not other information in the system
- v. How to integrate the STEP data models into an enterprise's data systems
- vi. How to integrate legacy (i.e. non-STEP) applications and systems with those that are based on STEP

Consideration of the above issues had a significant impact on the underlying schema of CIS/2 (i.e. on the Logical Product Model). Because LPM/5 was developed for implementation in a DBMS, it (and LPM/6) is very different from LPM4CIS (the basis of CIS/1), which was developed primarily for physical file exchange.

2 STEP IMPLEMENTATION LEVEL 1

2.1 The role of CIS translators

Aside from the more detailed CIS Conformance Requirements, such as those relating to log-files, error messages etc., the role of a CIS Translator at this level of implementation is simply to enable data exchange via a file whose format is specified by STEP Part 21^[17] and whose content is defined by the *neutral* data structures of the Logical Product Model (LPM)^[8].

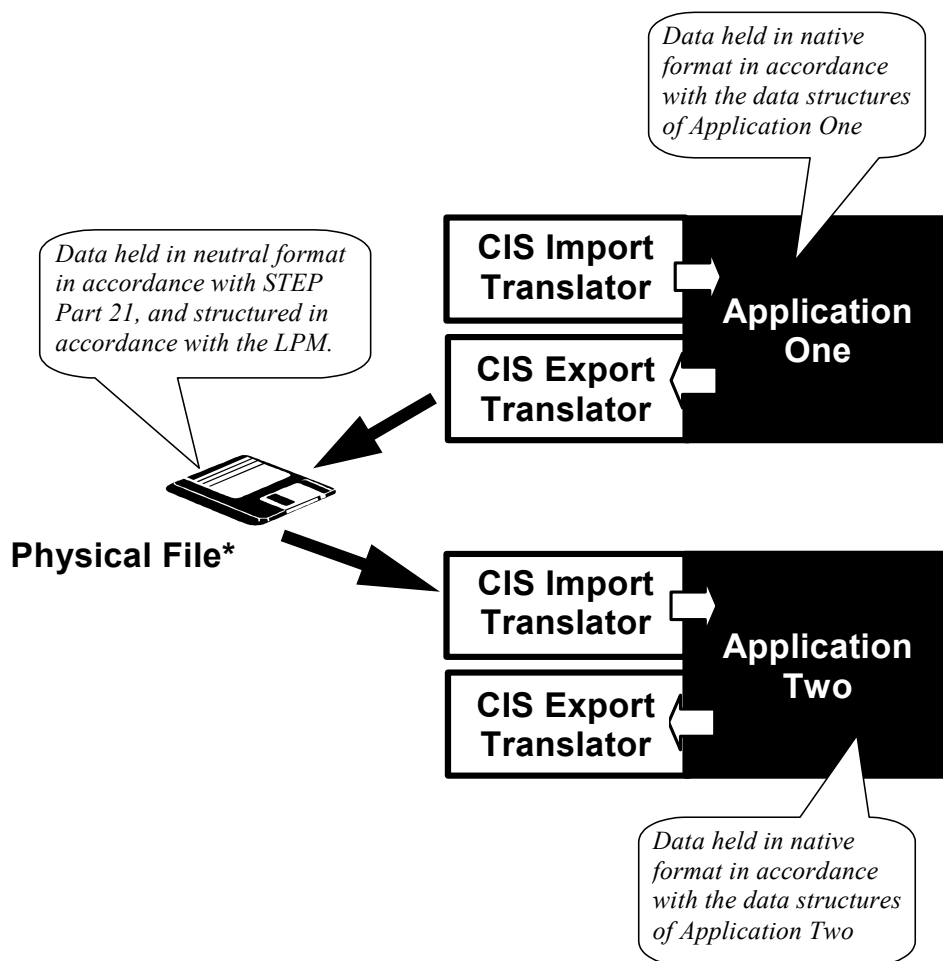


Figure 2.1 Data exchange via (Level 1) CIS Translators

*The physical file is shown in Figure 2.1 and Figure 2.2 as a floppy disk for illustration purposes only; the file may reside physically in any media form including magnetic tape and CDROM. Furthermore, the file may be shared via a networked server, an ftp site or an email attachment.

The role of the export translator developed for Application One (shown in Figure 2.1), is to write out a selection of the information held in the application's own *native* data structures as *neutral* data. That neutral data is contained within a physical file structured in accordance with the CIS specifications and formatted in accordance with STEP Part 21. Similarly, the role of the import translator developed for Application Two is to read

the neutral data and (for information relevant to the application) create instances of corresponding data in the native data structures of Application Two.

2.2 Components required for file exchange

For neutral file exchange, the components of the existing applications are supplemented by translators; one for import and one for export. As shown in Figure 2.2, import and export are (potentially at least) two halves of the same system. The import side contains a file reader and file parser, while export contains a file formatter and file writer. Each works around a basic translator program that puts data into (and takes data out of) a neutral repository. For file exchange, this repository is transient, as it is only required during the translation process. To the end user, the translation process is invisible. The translators are written as ‘add-ons’ to the application. Access to the translator is via the graphical user interface of the application.

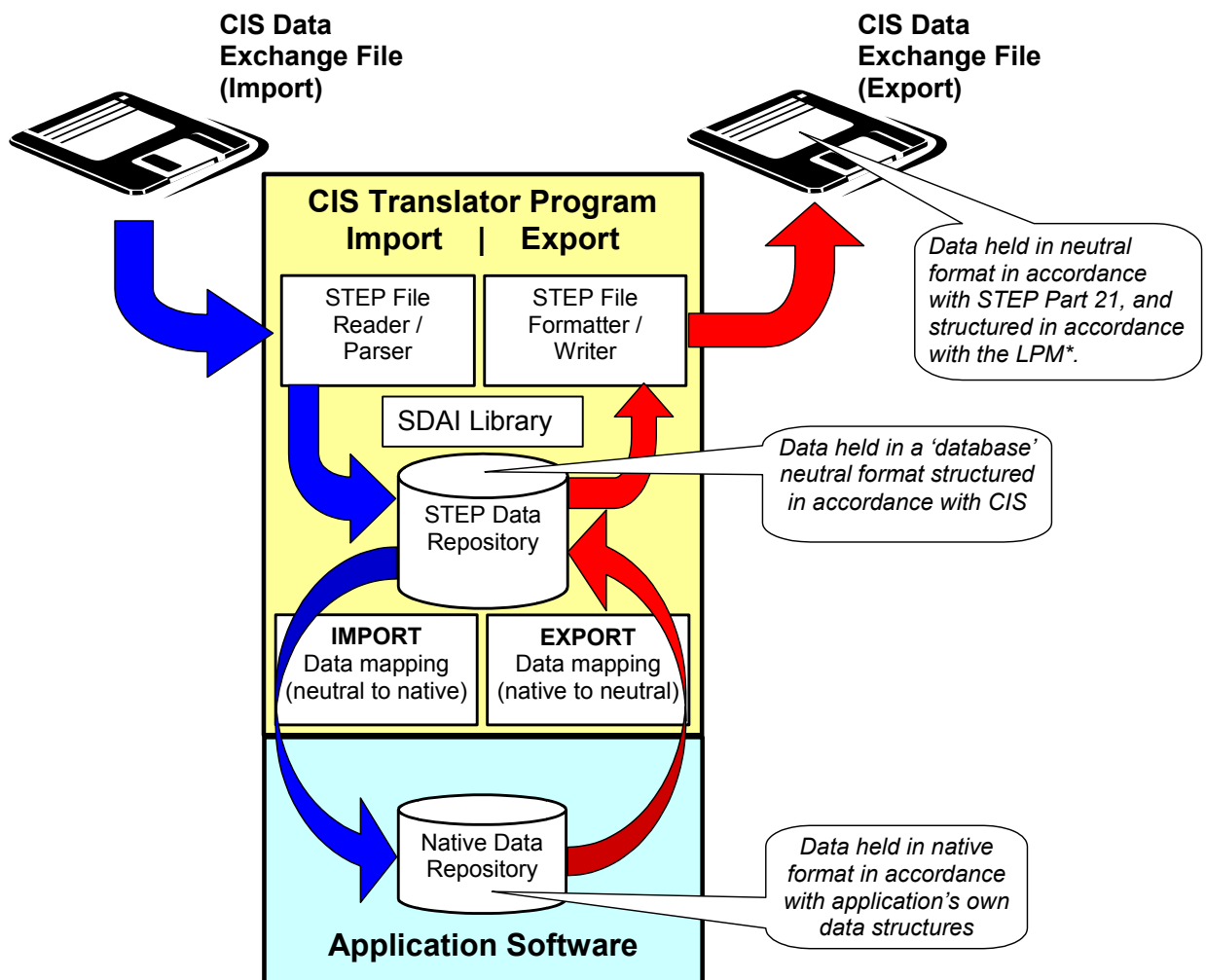


Figure 2.2 Conceptual view of a CIS Translator for file exchange

Figure 2.2 shows the primary components of a CIS file import/export translator. As can be seen, this translator has been implemented using a STEP Toolkit that supports SDAI (see APPENDIX G). A number of third party software vendors currently supply STEP toolkits that can be used for Level 1 implementations. Suitable STEP toolkits will automatically generate several of the primary components of Basic CIS Translator.

These toolkits can take an EXPRESS schema and parse them into a ‘working form’ (e.g. SQL, C++) which can be used to build an empty repository (or temporary ‘database’) that fully conforms to LPM/6. The same toolkits also provide the necessary functions for accessing those repositories (e.g. within a C++ Class library), and for reading and writing data exchange files in accordance with STEP Part 21^[17].

Thus, in common with other types of STEP translators, the primary components of a CIS file import/export translator can be generated automatically by using a suitable STEP toolkit. However, ‘hand’ coding is required to build the mapping logic that is specific to an application. The primary components are used to map all relevant information between the native data structures of the application and the neutral CIS data structures (on export), and visa versa (on import). The complexity of the logic of these procedures is overcome by the fact that the CIS instances are ‘buffered’ in the repository, and may be created or read in any order.

3 STEP IMPLEMENTATION LEVEL 2

In spite of the benefits of neutral file exchange, what interests CIS end-users the most is the prospect of neutral data sharing. This can be achieved when the translator of one application has direct access to the data in the STEP repository of another application. This avoids the need to create physical files, which results in a significant improvement in performance¹. Figure 3.1 shows how Applications One and Two are able to exchange data directly via their translators. The prerequisite for this is that the translators have to be compatible; i.e. they must be able to access and understand each other's repository. In practice this means that the translators have to be written using the same toolkit, since the format of the data held in the repository is dependent upon the toolkit that created the translator.

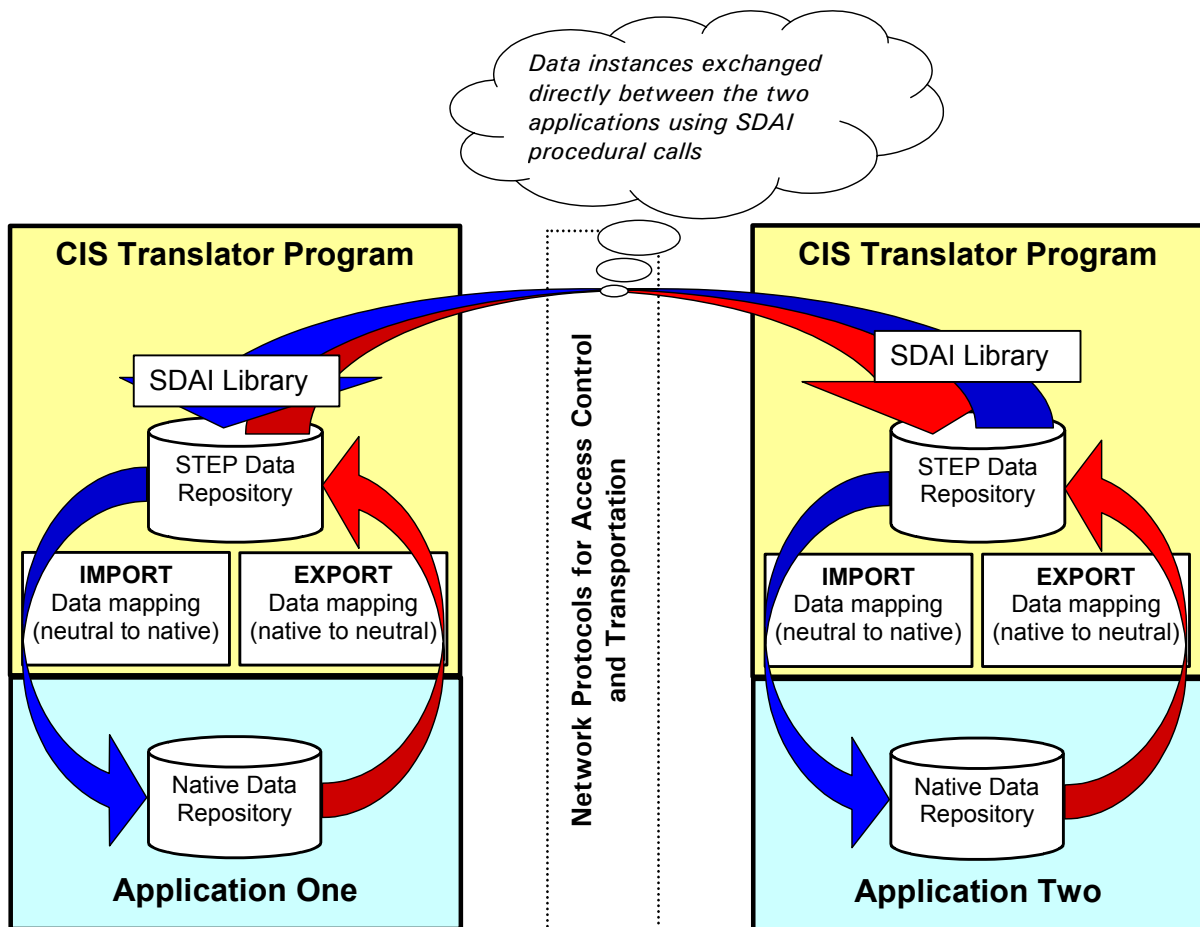


Figure 3.1 Conceptual view of data exchange via Level 2 CIS Translators

¹ Experience within the CIMsteel Project showed that over 90% of data processing time is taken up by reading or writing the physical file.

4 STEP IMPLEMENTATION LEVEL 3

A minimal Level 3 implementation would involve a single shared DBMS. The typical application in this context is one analogous to that performed by a Level 1 data exchange. In a Level 1 use, product data is generated as a STEP data exchange file and passed as a file to another application for processing. In a Level 3 data sharing use, product data generated by one application is first placed in a DBMS used as a STEP data store. Another application accesses this data from the same DBMS. No exchange file transfer is needed, since in the simplest case, only one DBMS is involved, either application could access the DBMS using the database's Data Modelling Language (DML).

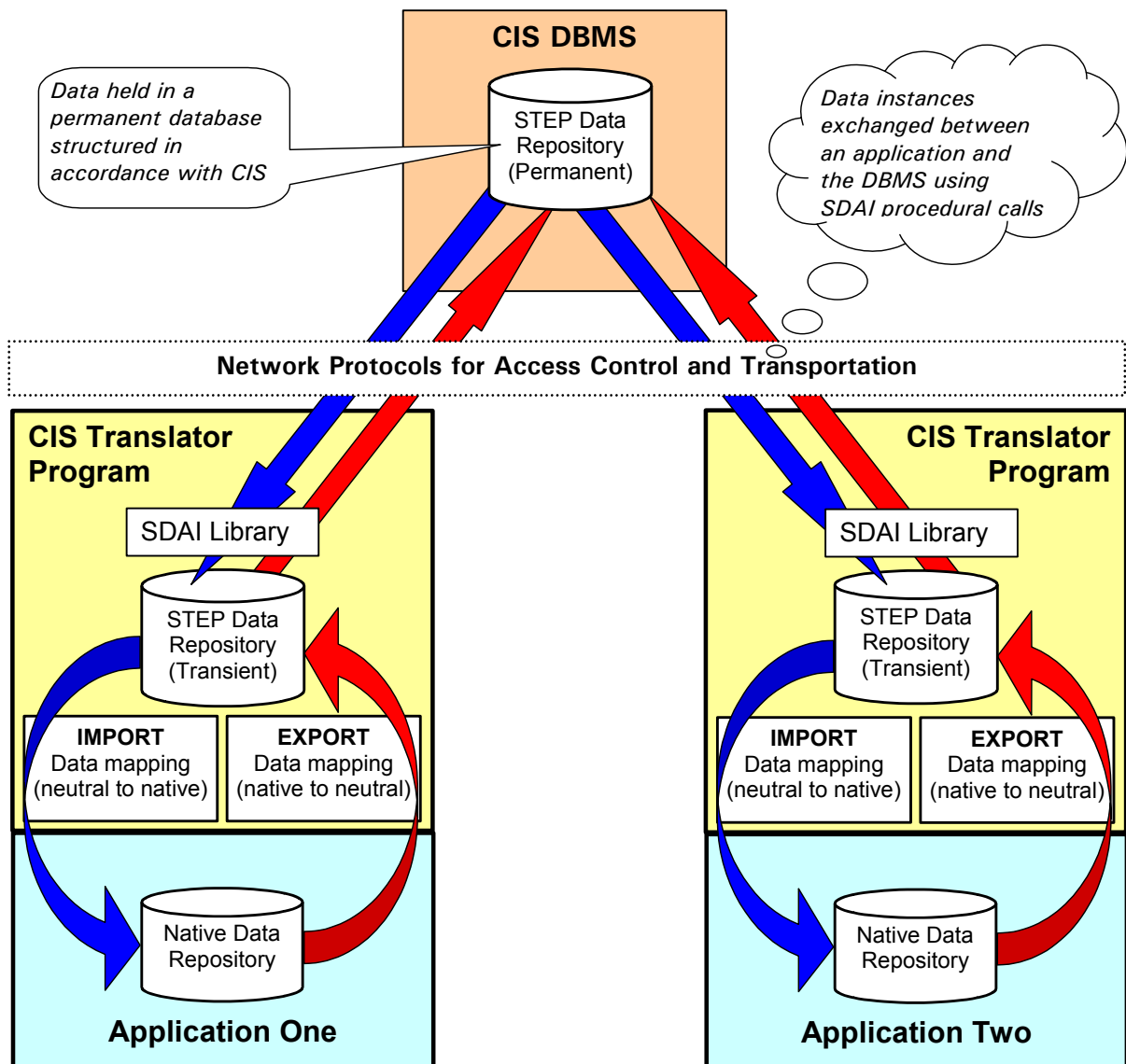


Figure 4.1 *Conceptual view of data exchange via Level 3 CIS Translators*

In a more elaborate environment, one should consider groups of design engineers working simultaneously on various aspects of a design (i.e. concurrent engineering). DBMS concurrence control assures data coherence in an environment where different

users can access the same data ‘simultaneously’. One of the features needed to support concurrent engineering is versioning, i.e. the ability to support evolving versions of the same STEP physical models. Other features required for project data management include:

- controls over data access based on authorisation or need, and
- control over data qualification and release.

A precedent for data sharing via a DBMS is the mechanism for data sharing via STEP physical files, as discussed earlier. This is the medium through which Level 1 data exchange takes place. A minimum file-based data sharing is outwardly similar to a Level 1 data exchange. However, more is required for Level 3 data sharing, including a new type of interface. The new interface would permit a STEP-compatible application to obtain the collection of entities that would have constituted the physical file. This data will be kept within the database in the form of ‘STEP physical models’ in SDAI repositories. The STEP interface seen by the application would then present back to the application the equivalent of what in a Level 1 implementation is called a ‘working form’. The application can examine the accessed entity collection in any order available through entity-to-entity links.

This distinguishes it from a Level 1 access in that no translation of the exchange file into a working form is necessary (which is required in Level 1). This working form is used for navigation purposes in a fashion similar to incremental data access of a database. Again, the difference is that the exchange file must be transferred and processed to permit this data access to emulate a Level 3 interface. Level 3 data exchange - to be most general - must utilize a mechanism that will permit STEP data to be accessed in terms of the data semantics of STEP and not in terms of the data semantics of a particular DBMS implementation. That is, to keep the ‘conversation’ on the level of the STEP conceptual data schema.

The greatest advantage of this file-based data store technique is that one does not have to consider limiting the database access to applications of the same technology. By attaching a tool to the DBMS for simple transformations of data stored in STEP physical files to data recognisable by the internal DML and vice versa, and by separating the internal data representation tool from the SDAI environment, the database can be accessible by users of different applications as long as they are compatible with the LPM. Thus, by using a file-based data store architecture for a DBMS, one should achieve a better and clearer integration of applications.

5 BASIC CIS TRANSLATORS

This Section describes how ‘Basic CIS Translators’ for CIS-compatible file exchange are implemented. The procedures for writing the Basic CIS Translators are effectively the same as those for writing ‘STEP-compatible’ translators for physical file exchange.

5.1 Developing a Basic CIS Translator

The development of a Basic CIS Translator for neutral file exchange consists of two stages: engineering specification and software implementation.

The primary objective of the first stage is to map the native data structures of the application to the neutral data structures of LPM/6 (and visa versa). The outcome of this will be mapping tables, one for export and one for import. Unless the application’s data structures were based on the LPM, one-to-one mappings between the two will be unusual. For an import translator, a field in the native data structures may need to be populated by several attributes of several entities in the LPM, and a function will be required to relate them to one another. In other words, the value for a field in the native structure will be derived from some function of the LPM entity-attribute construct. The mapping table should define the algorithm that satisfies this function.

The aim of the mapping process is to provide a complete specification for the translator that is to be implemented. For an export translator, this means specifying the data to be mapped to the LPM and any algorithms needed to derive data that are not explicitly available from the application. For an import translator, this means specifying the data to be mapped to the application and any algorithms needed to derive data not explicitly available from the LPM.

The second stage is to write and test the translator code, guided by specifications in the mapping tables and taking advantage of functions provided by the STEP Toolkit. Before embarking on this task it is advisable to be clear how the various additional Conformance Requirements, such as error messages, will be addressed.

5.1.1 Critical factors for success

The prerequisites for writing a successful Basic CIS Translator for an existing application include:

1. A detailed understanding of the application’s own internal data structures at both a micro and a macro scale; knowing what each line of the data structures really means when processed together, rather than simply basing a translator on what may have been documented.
2. An understanding of LPM/6 and the relevant Sections of the CIS/2 documentation set.
3. An appreciation of the *Conformance Requirements*^[9] and in particular, the procedures for Conformance Testing.
4. An understanding of the relevant STEP technology, including:
 - the EXPRESS data definition language^[16]

- the physical file format^[17]
- the Standard Data Access Interface SDAI^[18] and either
- C++ Language Binding to SDAI^[19] or
- C Language Binding to SDAI^[20].

5. Significant C and/or C++ programming experience.

6. An understanding of the particular STEP Toolkit.

Selecting the right people for the task is also critical to the success of the implementation, having a very significant impact on the overall time to completion. Although the most successful translators to-date have been written by senior software engineers with experience in both structural engineering and programming, the two stages of engineering specification and software implementation are likely to involve different specialists. This will demand a high level of co-ordination within the development team. It should also be recognized that the time consumed by the development and implementation of a vendor's first CIS translator will be much longer than that required for subsequent translators, which benefit from the development team's increasing familiarity with the new technology. It is, therefore, advisable to tackle a Basic CIS Translator with a small engineering scope before moving onto more advanced implementations.

5.1.2 Two-stage translators

Although not the preferred solution, CIS translators may be developed as two-stage translators. Software vendors who have already written one-to-one native links between their application and the applications of other vendors may see this as seen a more efficient way of developing CIS-compatible translators. In this situation, vendors may chose to convert the ASCII neutral STEP data exchange file into an ASCII native file, rather than importing data directly from the STEP data exchange file into the application. This separates the file read/interpret from the import. This also allows some applications to be 'CIS-compatible' with no additional effort, by going through another application. In effect, two applications that are not CIS-compatible can exchange data via two other applications that are CIS-compatible.

If this technique is employed, it must be invisible to the end user; i.e. the translator should operate as if it were importing or exporting the STEP data exchange file. Furthermore, it is a Conformance Requirement of the CIS that translators be activated by a single menu command or icon.

5.1.3 Translator programming

Before writing the translator program it is necessary to identify how the data will be passed between the translator and the application. This may be achieved by using the application's native data files or an external database. It may also be achieved by embedding the translator within the application and writing the STEP file directly.

If the application's native data files are used it is necessary to identify the physical format of the data files and their logical structure. For example, the data may be arranged in tables, 'C' structures, sequentially, or comma delimited. ASCII files may be easier to deal with than binary files since they are essentially independent of language, compiler, and platform. However, some applications may use binary files for improved speed of reading and writing. In this case the translator programmer may need access to the import/export source code of the application in order to access those files.

When using the application's native data files the following sequence occurs:

1. The export translator program opens the data file.
2. The export translator program reads the data.
3. The export translator converts the data to the neutral (STEP) format.
4. The export translator places the data in the neutral (STEP) repository.
5. The export translator takes the data from the repository and writes the data to a STEP physical file.

The import translator program undertakes the reverse procedure (reading rather than writing).

If the programmer is able to embed the translator within the application, the task is easier. The functions that currently write the application's data files are modified to write data directly to the STEP repository using the mapping algorithms, and export a STEP exchange file.

5.2 The procedure for translator development

The following procedure should be followed when developing a Basic CIS Translator for importing or exporting data exchange files:

1. Analyse and understand the application's own internal data structures. (If possible, create an EXPRESS model representing the application's data structures).
2. To define the engineering scope of the translator, identify which Units of Functionality (UoF) the application could support by examining the CIS Application Activity Model (AAM). (Both the AAM and the UoF are specified in Volume 3 of the CIS/2 documentation^[7])
3. Consider which units and unit systems the translator will support. The mechanisms for unit systems are described elsewhere^[8, 9] and may be defined for SI units or US Imperial units. Alternatively, vendors may implement their own unit system, so long as it accords with the data constructs of LPM/6.
4. Consider which 'item references' the translator will support. Item references are documented elsewhere^[8, 9] and may be defined for generic products such as hot rolled section profiles (which come under the class of 'standard items'), or cold formed purlins (which come under the class of 'proprietary items or library items').
5. From the available Conformance Classes (CCs) specified in Volume 5 of the CIS/2 documentation^[9], select the most appropriate combination of CCs for the application to support. The combinations of the CCs that contain the data constructs for the Units of Functionality identified in step 2 will be the most appropriate.
6. From the chosen CCs, analyse the list of included entities (specified in Volume 5 of the CIS/2 documentation^[9]). Refer to the definitions, explanations, and diagrammatic representations of those entities provided in Volume 4 of the CIS/2 documentation^[8].

7. Identify and document those entities, attributes, and relationships in the application's data structures that have a direct mapping to entities, attributes and relationships in the list of included entities provided for the CCs. Check whether the information content of those entities can be fully replicated within the application's own native data structure.
8. Where appropriate, write the export mapping table.

Neutral	Function	Native
entity_a		
.attribute_1	<<	table_x.field_3
.attribute_2	x1000	table_z.field_9
entity_b		
.attribute_1	<<	table_y.field_1
.attribute_2	<<	table_p.field_9
.attribute_3	<<	table_z.field_2
.attribute_4	<<	table_z.field_3
CIMsteel		

Figure 5.1 *An example of an export mapping table*

Native	Algorithm	Neutral
table_x		entity_a.attribute_1
.field_1		entity_a.attribute_2
.field_2	/1000	entity_c.attribute_5
table_y		entity_b.attribute_1
.field_1		entity_b.attribute_2
.field_2		entity_c.attribute_2
.field_3		entity_f.attribute_2
		CIMsteel

Figure 5.2 *Example of an import mapping table*

This is simply a structured tabular presentation that has:

- All the relevant CIS entities and attributes listed in the left column.
- All the corresponding native data entities in the right column.
- A definition of how each item of information is mapped from the native data structures to populate the CIS entities and attributes described in a central column, together with the appropriate algorithm.

This export mapping should also include the mapping of units from native to neutral systems, and the algorithms required to convert the numerical values of physical quantities from one system into another. An example of an export mapping table is shown in Figure 5.1.

Alternatively, write the import mapping table. (An example is shown in Figure 5.2.) This is simply a structured tabular presentation that has:

- All the relevant native data entities listed in the left column.
- All the corresponding CIS entities and attributes in the right column.
- A definition of how each item of information is mapped from the CIS structures to populate the native structures described in a central column, together with the appropriate algorithm.

In some cases, there will be a direct mapping of an entity, an attribute, or a relationship. In other cases, it will be necessary to define adequate indirect mappings. The required conversion algorithms should be specified in the mapping table. If a mapping for some CIS attributes cannot be identified this should be recorded in the mapping table and the intended scope of the translator reviewed.

This import mapping should also include the mapping of units from neutral to native systems, and the algorithms required to convert the numerical values of physical quantities from one system into another.

If translators need information that is not available from the application's native data structures to populate the neutral data exchange file completely, this should also be recorded in the mapping table. The mapping table should make clear whether the value is derived from:

- the operating system supporting the application,
 - from user input during the installation of the translator,
 - from user input during the operation of the translator, or
 - from default values hard coded into the translator.
9. If appropriate, define and document the mappings for the 'item references' identified above. These should map between each neutral item reference and its equivalent in the application's native database. The application should thereby have access to the data that is associated with each neutral item reference.
 10. Purchase a STEP toolkit that is suitable for commercial translator development. Preferably the toolkit should be one that supports the same operating system, programming language, and compiler as the application for which the translator is being written. The toolkit may offer a choice of early² and/or late³ SDAI bindings.
 11. Using the STEP toolkit, compile the LPM/6 schema provided with CIS/2. The form of the output will depend upon the STEP technology purchased, but for the purpose of this guide, C++ classes are assumed.

² Schema specific bindings to the repository (typically specific C++ classes).

³ Schema independent bindings to the repository (typically generic C functions).

12. Using the Application Programming Interface (API) provided with the STEP toolkit (e.g. SDAI), write the import and export translator code. This code will map data between the application's data structures and, in this case, the C++ class definitions of the entities in the list of included entities. The code will be based on the documents produced as a result of steps 1, 2, 7, 8, and 9.
13. Using the toolkit's API, the translator code from step 9, and the C++ classes from step 8, create a STEP exchange file that contains data mapped from the application. Using the reverse procedure read the data from the STEP exchange file back into the application.
14. Progressively implement, test and debug the logic of data mappings of the translator and check the adequacy of the required monitoring and error messages generated during import or export.
15. Complete the formal documentation and conformance statement.
16. Submit the application and the Basic CIS Translators to a recognized third party testing authority for formal Conformance Testing.

This is a simplified view of the complex tasks of analysis, design, coding, testing, and debugging Basic CIS Translators for commercial software applications. Some translator developers will find translator development relatively trivial. Others will encounter some difficulties. Much will depend on the application involved, the STEP technology purchased, and the CCs supported. Consideration should also be given to structuring the Basic CIS Translator to allow future upgrading to more advanced implementations. Software vendors must also bear in mind the more detailed CIS Conformance Requirements described elsewhere^[9].

Figure 5.3 illustrates the distinction between those aspects of the CIS Translator that will only be seen as part of the development process, and those aspects that will be part of the CIS Translator during its operation.

5.2.1 Why use a STEP Toolkit?

Software vendors are at liberty to develop CIS/2 translators without a STEP Toolkit. For simple applications, this may be a practical proposition. However, vendors should note that:

- A STEP Toolkit creates an object-oriented database based on the given EXPRESS schema(s), and provides a standard (SDAI) interface to such a database
- A STEP Toolkit enables direct populating of such database by scanning a STEP file and vice-versa (dumping of database in the form of STEP file)
- A STEP Toolkit also enables manipulation of an EXPRESS schema that will prepare the environment for development of the translators. The simple once-only operation is as follows:
 - Create the EXPRESS definition library
 - Parse and link the LPM EXPRESS schema supplied.

- Create the Master Working Form Library using the settings in the toolkit's configuration file

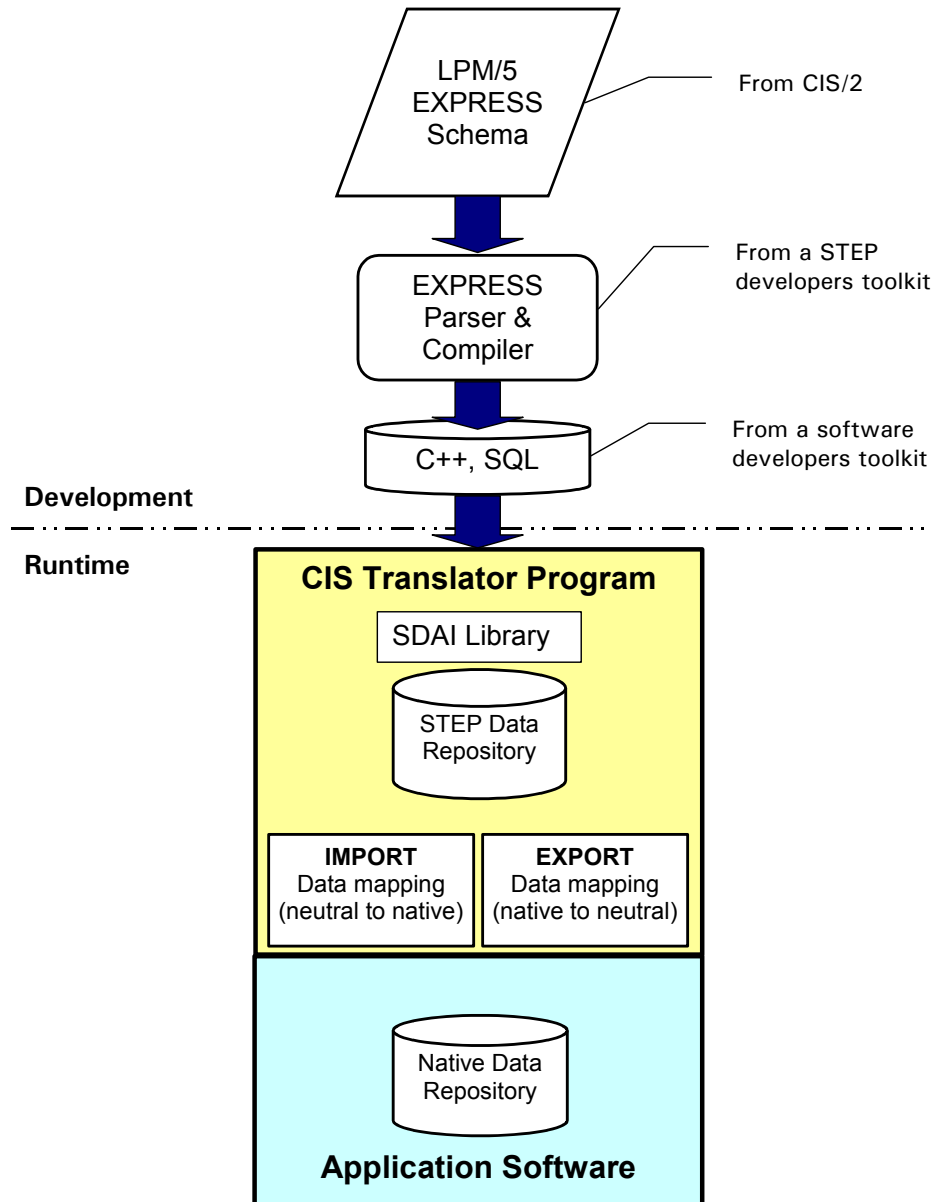


Figure 5.3 *Conceptual view of CIS translator development*

6 DMC TRANSLATORS

This Section describes how Data Management Conformant (DMC) CIS Translators are implemented. ‘DMC Translators’ are expected to support data sharing and management through physical file exchange or through direct interfaces. The procedures for writing DMC Translators discussed here are *additional* to the procedures for writing Basic CIS Translators for file exchange.

6.1 ‘DMC Translators’ and ‘DMC Applications’

A translator written for a CIS/2-conformant system that has data management capabilities is known as a DMC Translator. The extent of the data management functionality will depend upon the capabilities of the underlying application for which the translator is written, i.e. whether the native application can manage data. It will also depend upon whether the system has both an import translator and an export translator. Where a native application can manage data and is provided with both an import translator and an export translator it is referred to as a DMC application.

The number of data entity constructs (used to represent engineering information) that a DMC translator supports will be dependent upon the scope of the application for which the DMC translator is written. The creation and processing of these data entity constructs is handled in the same way as those for a Basic CIS Translator.

6.2 The purpose of data management

The main purpose of data management is to allow engineering data created by one software application to be updated and passed onto another application or fed back to the original application. This information flow and feedback must be made possible regardless of the application’s scope, functionality, or position in the production process. To manage the data, an application must be able to ‘recognize’ incoming data instances as modified versions of existing data instances.

6.3 Data Management and CIS/2

Providing an adequate and efficient mechanism for data management within the LPM was seen as the key to the long-term success of the CIS. CIS/2 provides data management functionality in the LPM by separating the data from the ‘meta-data’, and then allowing those applications that can associate the ‘meta-data’ with the data (i.e. DMC applications) to do so. A further constraint was that DMC and non-DMC applications have to be able to share data with each other. This is because different software applications will develop at different rates; some will include data management capabilities before others, but users of DMC applications will still wish to share data with users of non-DMC applications.

The ‘meta-data’ includes some or all of the following information:

1. The unique identifier for the item.
2. The person who created the data.
3. The date when it was created.

4. Whether the data was new or old data that had been modified.
5. The date when it had been modified.
6. Whether the data has been approved or not.

For example, a DMC-application would be used to assign a unique identifier to an instance of ‘assembly’ and place it in a set that states who created the instance, when it was created, and whether it had been modified. This is discussed in more detail elsewhere^[8].

In simple terms, DMC applications produce ‘managed data’ such that each item of data is given a unique identification and placed in a data set, while non-DMC applications produce ‘unmanaged data’. Thus, assigning meta-data to data requires associations to be made between the engineering data and (for example) the data about the person who created that data. Furthermore, it is a Conformance Requirement of CIS/2 that DMC applications are able to map appropriate meta-data onto ‘unmanaged data’ to produce ‘managed data’.

It should be noted that the engineering data can exist without the meta-data, and is not dependent upon it. On the other hand, the meta-data is dependent upon the data for which it has been created.

6.4 The meta-data in LPM/6

A simplified view of the basics of data management constructs in LPM/6 is shown in Figure 6.1, and Figure 6.2, while the more advanced facilities are illustrated in Figure 6.3.

By definition, a DMC translator must be able to support all of the data management constructs specified in Section 19 of *The Logical Product Model*^[8] and illustrated in EXPRESS-G in diagram 1 of Appendix B of the same document.

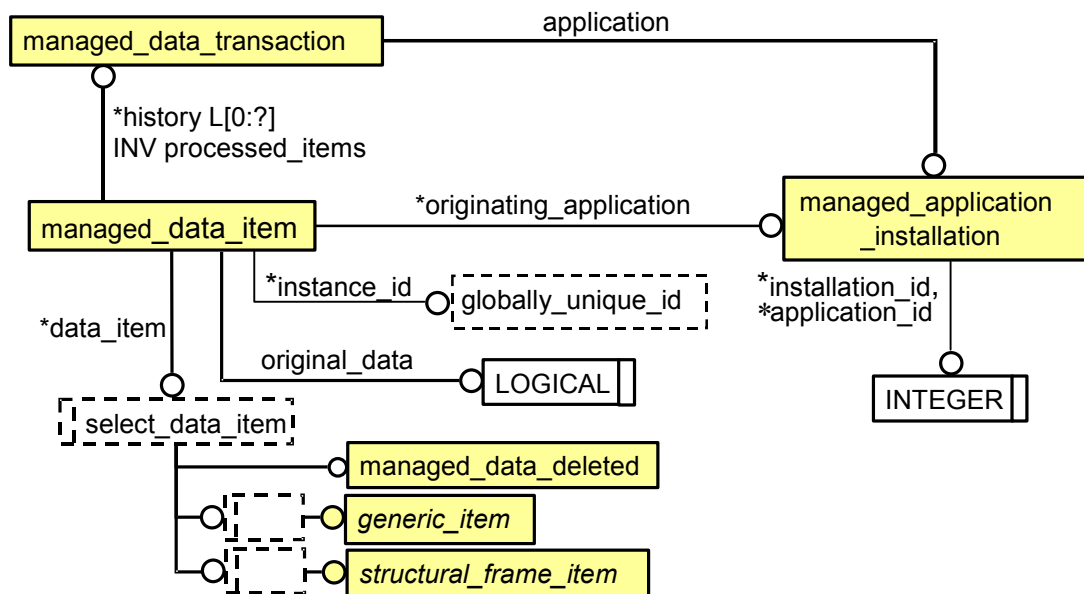


Figure 6.1 Assigning Meta-data to data in LPM/6 (Revised for 2nd Edition)

Of most concern to the writers of DMC-Translators are the two entities ‘managed_data_item’ and ‘managed_data_transaction’, which form the essential EXPRESS constructs in the LPM for data management by capturing the meta-data. A ‘managed_data_item’ represents an individual item of managed data and allows that data item to be assigned a unique identifier, and a history. A ‘managed_data_transaction’ is simply a record of the use of a DMC application when performing some external data transaction (e.g. an export or import process).

6.4.1 Referencing the data

LPM/6 uses a cascading series of SELECT constructs to create a reference to a single instance of an entity representing engineering information. Each of the elements in the SELECT list is itself a SELECT TYPE. Each one ultimately references a single data entity.

Because a SUBTYPE is a type of its SUPERTYPE, the SELECT construct extends to the entire depth of the SUBTYPE-SUPERTYPE ‘tree’. In this way, a ‘managed_data_item’ may reference any data entity through its ‘data_item’ attribute. Developers of DMC Translators should be aware of the complexities that are associated with these SELECT constructs. In particular, they should study clause 11.1.8 of the ‘Technical Corrigendum to STEP Part 21^[17]’, which explains how SELECT constructs are mapped onto the exchange structure.

6.4.2 Identifying the data

When a DMC translator imports data into its application, it must be able to recognize the different types of data. It does this by the unique identifier associated with the data instance. Thus, at its simplest level, data management is the process of associating engineering data with a mechanism for identification.

The unique rules

The entity ‘managed_data_item’ contains two UNIQUE rules, which ensure that the identifier (as captured by the attribute ‘instance_id’) is unique to one particular instance of ‘managed_data_item’, and that the ‘managed_data_item’ is unique to one particular instance of the underlying data entity. The second rule creates the required one-to-one relationship between the ‘managed_data_item’ and the data entity. Put together, these rules create the unique identification for the data entity instance. The ‘uniqueness’ created by these rules is only effective within one ‘exchange structure’; e.g. a physical file or data repository.

Thus, so-called ‘unique identifiers’ are forced to be unique within one physical file, but unless further constrained, could actually appear in two or more physical files. The following Sections explain how the identifiers are constrained further by CIS/2.

How long does an identifier have to last?

Users of DMC applications require unique identifiers to be consistently and permanently unique ‘throughout the universe’ (or at least, throughout the world of steel related software applications) for all time (or at least, for the ‘lifetime’ of the data, which could be over 50 years). Once created, a unique identifier for a ‘managed_data_item’ (and thus for an instance of an engineering data entity) must last forever. That does not mean that the unique identifier must appear in every occurrence of data exchange or every physical file. It does mean that whenever a particular instance of data is exchanged, it is exchanged with its unique identifier. Furthermore, that identifier must be maintained

when the data instance is modified. And, given any number of physical files, a DMC translator must be able to recognize a particular item of managed data. More importantly, a DMC translator must be able to recognize that two instances of ‘managed_data_item’ bearing the same unique identifier represent the same engineering information and that the associated data entities may represent two different versions of that information.

The form of the unique identifier

As discussed above, it is expected that a DMC translator would be able to recognise any piece of information created by any other DMC translator because of the unique identification system that is provided by CIS/2. This identification system ensures that once created, a unique id remains valid throughout the life of the data. Or put another way, once created, a particular unique id must never be used again – it may only be used to identify one particular data entity instance.

In the Second Edition, the underlying data type of the attribute `instance_id` has been changed from an INTEGER to a STRING. For upward compatibility, an `instance_id` written in accordance with the first edition of CIS/2 needs to be converted from an INTEGER to a STRING.

An `instance_id` is defined as an alphanumeric identifier that is unique throughout the software world. This identifier is a string encoding of a globally unique unsigned 128bit integer and is also known as a Globally Unique Identifier (GUID) or a Universal Unique Identifier (UUID). Algorithms for generating the 128bit integer and for encoding this integer in a string representation are defined in documentation published by ISO^[36] and by the Open Group^[42]. The string representation consists of a sequence of hexadecimal fields separated by single dashes totalling 36 characters. An example of the `globally_unique_id` is the string “1A81FD96-D7A8-47d3-8DF7-BEEA6FF45184”.

The Microsoft Foundation Class (MFC) function “CoCreateGuid”, which is an implementation of the referenced algorithms, may be used by CIS/2 implementers on Windows platforms to create this identifier (as a GUID). (See the Microsoft Online Documentation^[40].) Other implementations are available as UUID generators written in Java and other languages for use on MacOS and the various Unix platforms.

Because the domain of the data is so large, it has to be placed within a specified context. The context for CIS data is the world of structural steel software applications. This allows us to place our data within the context of the software application. If the software application is given a unique identification, then the combination of the two identifiers provides our unique identification system. LPM/6 splits the software identification into two parts; an ‘application id’ and an ‘installation id’.

Thus, as illustrated in Figure 6.1, each instance of ‘managed_data_item’ will be assigned a three part unique identifier derived from the attributes ‘application_id’, ‘installation_id’, and ‘instance_id’. Once created, the unique identifier must be maintained by all DMC Translators. Thus, even though several applications will contribute to and modify data instances, the unique identifier will be assigned by the application that originally created that managed data.

For uniqueness, the requirements for each of the three parts of the identifier are as follows:

- The **‘application id’** should be unique to an application and its vendor as it is intended to identify the software vendor, the software application and the version of the application. To ensure its uniqueness within the domain of CIS/2, the ‘application id’ should be registered with the **CIS International Technical Committee** before it is used. Before developing a DMC translator, a software vendor needs to apply to the **CIS International Technical Committee** to obtain their registered application_id.
- The **‘installation id’** should be unique to one installation of a particular application as it is intended to identify the ‘registered keeper’ of the software. To ensure its uniqueness within the domain of the particular software application, the software vendor is required to assign and maintain the installation id for each installation of each version of each DMC-application.
- The **‘instance id’** should be unique to an instance created by a particular implementation of an application, as it is intended to identify the instance of data. To ensure its uniqueness within the domain of the particular CIS implementation, the DMC Translator is required to assign and maintain these instance ids.

To represent an installation of a DMC application on a particular computer system LPM/6 uses the entity ‘managed_application_installation’. This allows every instance of ‘managed_data_item’ to have an instance of ‘managed_application_installation’ associated with it (through the attribute ‘originating_application’) representing the DMC application that originally created that managed data.

Where all that is required is the passing of a GUID, then the application id and the installation id will not be populated. However, all three components of the CIS identifier will be populated when applications are required to capture and maintain the history of a data collection.

6.4.3 The ‘who’ and the ‘when’

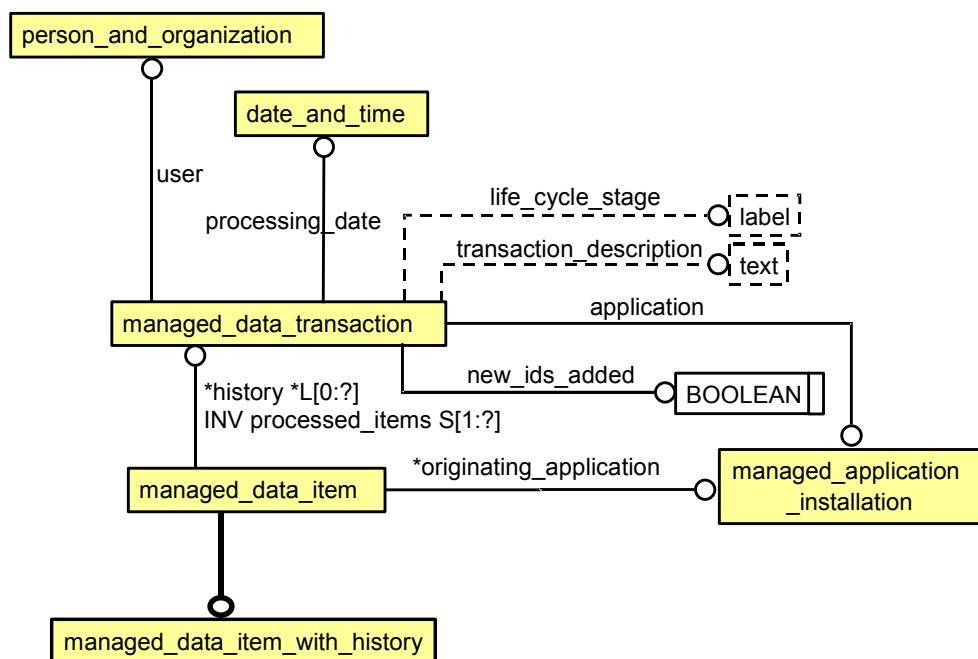


Figure 6.2 Assigning the ‘who’ and the ‘when’ of the Meta-data
(Revised for 2nd Edition)

To assign ‘*the who*’ and ‘*the when*’ information to the data, which will be repeated for several thousand data instances, LPM/6 uses the entity ‘managed_data_transaction’ to represent a set of managed data. As shown in Figure 6.2, this allows collections of managed data that have been processed by a particular person at a particular time to be placed in that set.

A ‘managed_data_transaction’ collects together managed data, and assigns the additional ‘meta-data’ that may be associated with each and every item of data in the collection. An instance of the entity ‘managed_data_transaction’ represents the use of a DMC application by a particular person at a particular time. For example, a DMC application may be used to import or export managed data. It may also be used to create or modify managed data. Thus, as illustrated in Figure 6.3, the entity ‘managed_data_transaction’ is a SUPERTYPE of the entities ‘managed_data_import’ and ‘managed_data_export’. It also has the SUBTYPE entities (not shown in Figure 6.3) ‘managed_data_creation’ and ‘managed_data_modification’. (In this context deletion is considered to be a type of modification.)

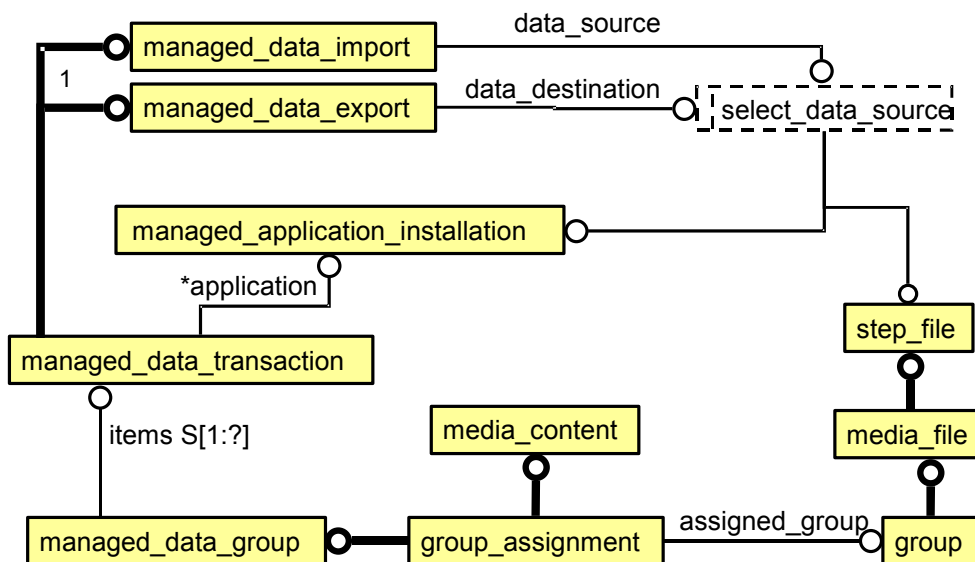


Figure 6.3 *Advanced data management constructs*

These SUPERTYPE-SUBTYPE constructs provide more explicit definitions of the role of the person and date associated with the data. LPM/6 also contains rules that force an instance of ‘managed_data_item’ to be associated with only one instance of a ‘managed_data_creation’.

The SUBTYPES also add a ‘data_source’ attribute for import processes and a ‘data_destination’ for export processes, to indicate where the data has come from or is going to. Both of these attributes have the data type ‘select_data_source’ which selects one from a choice of either a ‘managed_application_installation’ or a ‘step_file’. The ‘step_file’ is simply a type of group.

6.5 Data management in practice

6.5.1 Capturing data history

DMC applications are not required to maintain values for every version of every attribute of every data entity that the DMC application encounters. To do so, would involve the accumulation of a vast quantity of data, most of which would be irrelevant. However, engineering end users of DMC applications will expect to be able to capture the most recent engineering data, and its development process. Thus, the meta-data that is associated with the various sets of managed data (via the entity ‘managed_data_transaction’) indirectly provides the history of the engineering data. Although each stage of the development of the engineering data is not captured explicitly, LPM/6 does allow a date and time as well as a person and organization to be associated with each instance of ‘managed_data_transaction’.

Every time a DMC translator exports data from its application, it must create at least one new instance of ‘managed_data_transaction’ (or one of its SUBTYPES), giving the details of the use of that application. The set of data associated with the instance(s) of ‘managed_data_transaction’ will include every single instance of data that the application has processed. This set will include:

- the new data instances that the application has created
- the data instances that have been modified by the application
- the data instances that have been deleted by the application, and
- the data instances that have passed through unchanged.

The original data instances that have been ‘passed through’: i.e. imported and then exported unchanged, will be exported by the DMC application if these lie within the scope of the export translator. Entities lying beyond the scope of the export translator will not be exported. DMC applications are only required to maintain the data that lies within the scope of their translator, and record the history of the data that it knows about. The scope of the data in an export file is dependent upon the scope of export translator, rather than the scope of every application that has processed the data. Thus, the scope of the export file could (potentially) be smaller than that of the imported file. It should be remembered, however, that any transaction (e.g. via a physical file) represents a ‘snapshot’ of the data rather than the totality of the data.

6.5.2 More advanced data management

As a construction project progresses, the number of instances of ‘managed_data_transaction’ that are created by a DMC application could increase as the scope of the data included in an export file grows to include (potentially) the scope of every application that has operated on the data. However, only a PMR (discussed in Section 9) is required to capture the totality of the data because it is the only type of DMC application that could. Because the scope of a PMR is effectively the scope of the LPM, full data history capture is possible. The scope of the data held within a PMR will grow with every transaction and will encompass the scope of every application that has operated on the data. In this case, there will be as many instances of ‘managed_data_transaction’ as there have been transactions.

When connected directly to a PMR, an advanced DMC application may also provide an option to ‘export the unchanged original data’ or ‘export only the changes’. If they are exported, the export file will contain one instance of ‘managed_data_transaction’ for each

usage of a PMR enabled application. These more advanced applications could also offer the user additional data management capabilities such as the opportunity to accept or reject changes, or replace one version of the data with another by its date or life cycle stage.

6.5.3 Inter-working between DMC and Non-DMC applications

If ‘managed data’ is modified by a non-DMC application, then the meta-data will be lost. The underlying data will, however, be useful to the non-DMC application. If ‘unmanaged data’ is received by a DMC application, its translator is required to be able to convert the data into ‘managed data’ by assigning meta-data as if the DMC application had created the data itself. To do this, DMC applications will create an instance of ‘managed_data_item’ with a unique identifier for every instance of every data entity. DMC applications will also create an instance of ‘managed_data_transaction’ for this set of data. The unique identifier will be created by (and depend on) the importing DMC application. That is, the ‘application_id’ and the ‘installation_id’ will be that of the DMC application that imported (and first managed) the data rather than the (non-DMC) application that originally created the data.

The data must be converted into managed data *before* being processed and exported as managed data. Thus, the ‘processing_date’ (the attribute of the entity ‘managed_data_transaction’) will refer to the moment when the unmanaged data was imported and converted into managed data. In such a case, there will be (at least) two instances of ‘managed_data_transaction’; one for the *import and convert to managed data*, and one for the operations performed on the data by the application. Although the application referred to will be the same instance of ‘managed_application_installation’ for both instances of ‘managed_data_transaction’, there will be two distinct instances of ‘date_and_time’ referenced by the attribute ‘processing_date’. The ‘person_and_organization’ referred to may, or may not, be the same for both.

When a non-DMC application imports a file containing managed data, it will simply ignore the meta-data by not importing the instances of the data management entities of LPM/6 (e.g. ‘managed_data_item’, ‘managed_application_installation’ and ‘managed_data_transaction’).

6.5.4 Deleted data

There is no real need to maintain the details of deleted data – merely the unique identifier of the data and the fact that it has been deleted. This information should be passed on to any application that may have received the original data. LPM/6 uses an entity that allows the ‘managed_data_item’ instance to be given a ‘deleted data flag’. A non-DMC application simply ignores the instances of ‘managed_data_item’, and can safely assume that all the data it receives is current and useful to that application.

The maintenance of instances of ‘managed_data_item’ will mean that whenever a data instance is deleted by an application, the associated instance of ‘managed_data_item’ will be exported with a reference to an instance of the entity ‘managed_data_deleted’. The data instance itself is not exported. Furthermore, when that deleted ‘managed_data_item’ instance is imported into another DMC application, it will be ‘passed through’ to any export file created from the imported data.

6.5.5 The size of the data set

Following the rules described above, the size of a physical file could be kept to a minimum, since the scope of the data in the file is dependent on the scope of the application exporting the file. The only ‘overhead’ will be the instances of managed_data_item marked as deleted, which is not seen as a major problem.

6.6 Developing a DMC Translator

The first stage in the development of a DMC Translator is the same as that for a Basic CIS Translator. The logic of the native-to-neutral and neutral-to-native data mappings remains valid. Thus, the greater part of the translator coding will have already been written for the Basic CIS Translator.

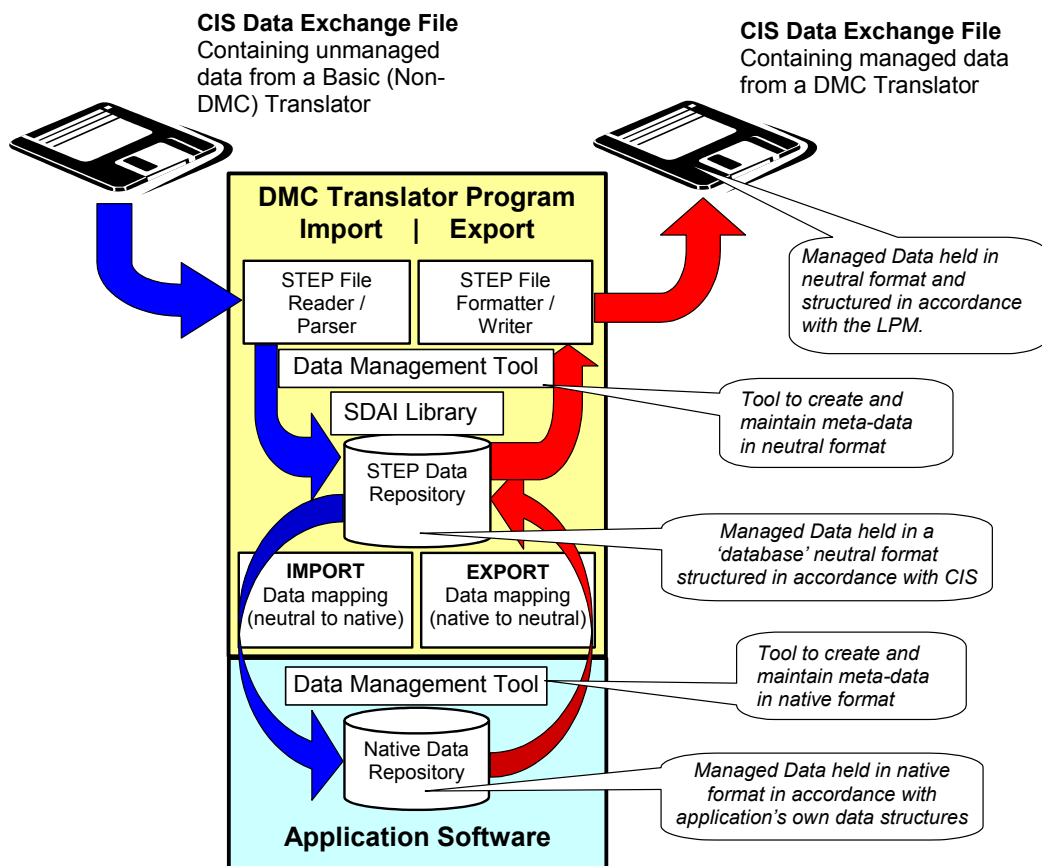


Figure 6.4 Conceptual view of a DMC Translator

As shown in Figure 6.4, a DMC Translator has an additional component known as a ‘Data Management Tool’. This is a tool for assigning the meta-data (including the unique identifiers) to the data. Both the STEP data repository and the native data repository will contain ‘managed data’. For an application that already has data management capabilities, the meta-data will be stored in the native data repository in the format and structure of the application. This native meta-data needs to be mapped to and from the neutral meta-data structured in accordance with CIS/2, and held in the STEP data repository. Applications that do not already have data management capabilities may be enhanced by having the data managed within the STEP data repository. In this case,

the STEP data repository must be permanent, rather than transient, and the Data Management Tool will only exist within the DMC Translator.

Figure 6.4 shows a DMC-translator sharing data via CIS data exchange files. DMC-translators may be implemented at any level of implementation form. It is likely that a DMC-translator will be incorporated into some form of database, in which case, data sharing may, but need not, involve direct procedure calls from the application to this database. A physical file that has been created by a DMC translator is effectively a 'partial dump' of the underlying STEP database.

6.6.1 Completeness of data management

It is a conformance requirement of CIS/2 that DMC translators are able to manage (and assign meta-data to) all of the engineering data that the export or import translator supports. In other words, partial data management is not acceptable.

7 IDI TRANSLATORS

This Section describes how Data Management Conformant CIS Translators are implemented such that they support Incremental Data Import and incremental updating. ‘IDI-Translators’ are expected to support data sharing and management through CIS-compatible file exchange or through direct interfaces. The procedures for writing IDI Translators discussed here are *additional* to the procedures for writing DMC Translators and Basic CIS Translators.

Section 10 provides the detailed rules for ‘incremental data import’ as defined for CIS/2. These rules, which are based on the EXPRESS language, are fundamental to Product Model-based information sharing and form the basis of concurrent engineering via CIS/2-compatible computer systems.

7.1 What is ‘Incremental Data Import’?

‘Incremental Data Import’ is the ability of a software application to add new information to an existing information set within the application while maintaining the integrity of the information set. This was identified as an essential requirement if software applications were to be fully integrated with each other, allowing information to flow to and from each application. This information flow and feedback must be made possible regardless of the application’s scope, functionality, or position in the production process. In order for heterogeneous systems to support incremental data import they have to be able to create and maintain the meta-data described in Section 6.

For effective incremental data import, an application must be able to ‘recognize’ incoming data instances as modified versions of existing data instances. This is made possible by the fact that LPM/6 provides a unique identifier for every instance of every data entity. It is important that an IDI Translator is able to maintain these unique identifiers for each and every entity instance, since these identifiers are the basis of the CIMsteel approach to incremental data import. In fact, incremental data import would be impractical (if not impossible) without these unique identifiers.

7.2 Different types of ‘Incremental Data Import’

When applications wish to share information, the type of incremental data import required will depend upon the scope of the applications concerned. Where the scopes are radically different, with no overlap or connectivity between the information sets manipulated by each application, then a simple addition will suffice. However, the information created, processed and shared between construction professionals is very complex. The life cycle of a construction project involves many activities performed by many different actors. These activities are not stand-alone processes, but intimately linked to each other. Consequently, the information set produced by an application serving one activity overlaps with the information set produced by another application serving a separate activity. These information sets may even be mutually dependent.

There are two scenarios for ‘Incremental Data Import’: the ‘master-slave’ scenario (illustrated in Figure 7.1), and the ‘feedback’ scenario (illustrated in Figure 7.2).

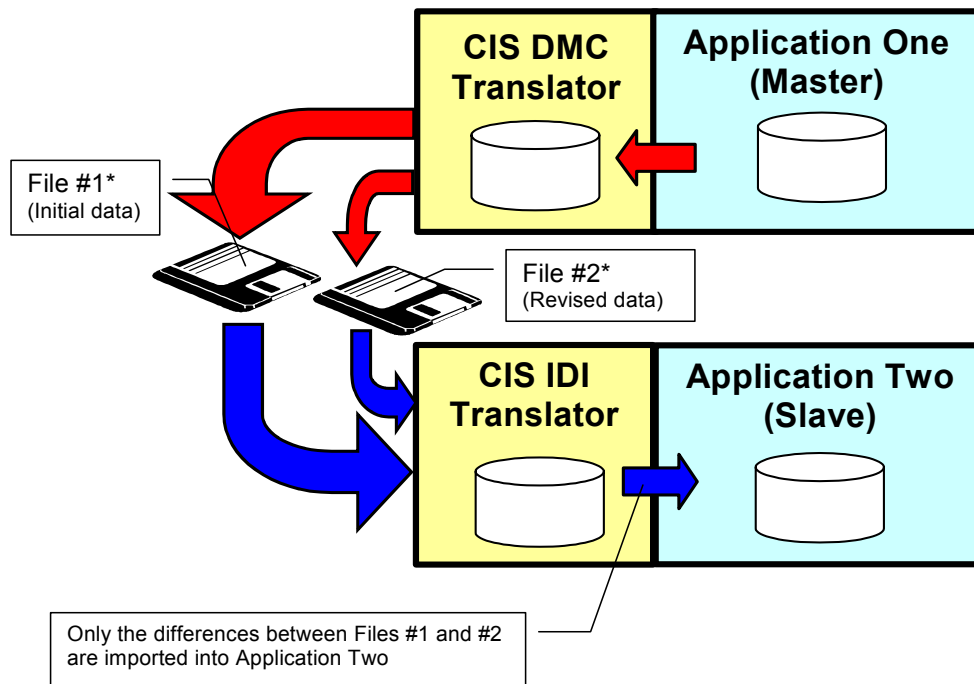


Figure 7.1 Data exchange via IDI Translators ('master-slave' scenario)

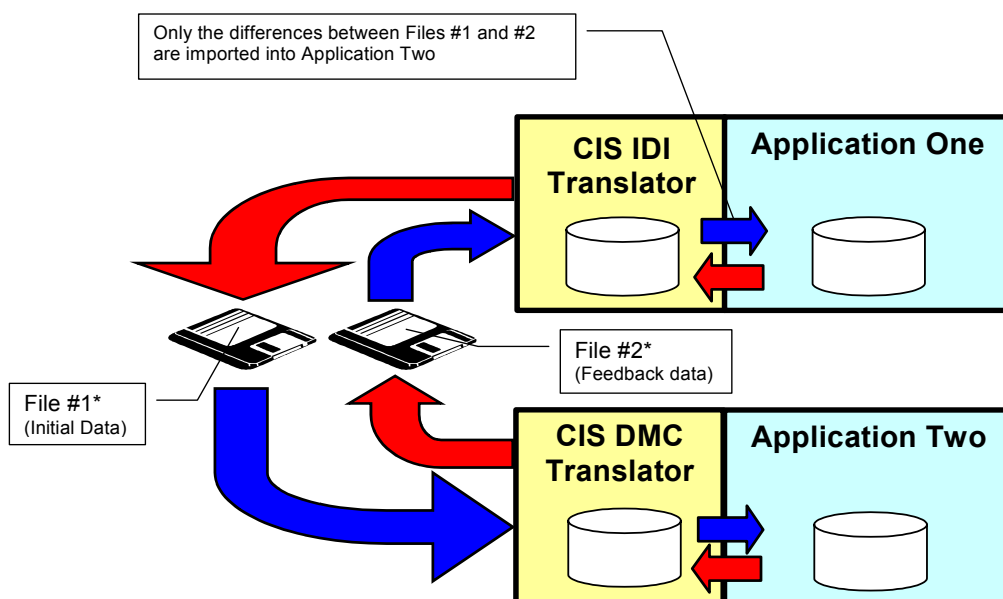


Figure 7.2 Data exchange via IDI Translators ('feedback' scenario)

**Although Figure 7.1 and Figure 7.2 show the incremental import of data from CIS data exchange files, other forms (or levels) of implementation are also likely.*

In the 'master-slave' scenario shown in Figure 7.1, Application One is acting as the 'master' in the sense that it is the originator of the project data. Application Two is the 'slave' application, which depends on Application One for the larger part of its source data. The slave application may, but need not, have an export translator. Similarly, the master may, but need not, have an import translator. The only proviso for this scenario

is that Application One has a DMC export translator, and that Application Two has an ‘Incremental Data Import’ Translator. The purpose of the IDI translator is to import **only** the differences between the revised data and the initial data. It does this by comparing the data instances contained in the original file it imported (File #1 in Figure 7.1) with those in the new file presented to it (File #2 in Figure 7.1).

In the ‘feedback’ scenario, Application One supplies data to Application Two, so that Application Two can add to, or modify, that data and feed it back to Application One. Here, Application One has an IDI Translator, while Application Two has a DMC Translator. The purpose of the IDI translator is to import only the differences between the new data and the old data. It does this by comparing the data instances contained in the original file it exported (File #1 in Figure 7.2) with those in the new file presented to it (File #2 in Figure 7.2).

7.3 Developing IDI Translators

The first stage in the development of an IDI Translator is the same as that for a Basic CIS Translator. The logic of the native-to-neutral and neutral-to-native data mappings remains valid. The second stage is to add the Data Management Tool, as described in Section 6.6. Thus, the greater part of the translator coding will have already been written for the DMC Translator.

An IDI Translator is a DMC Translator with an additional component known as a ‘Comparator’, as shown in Figure 7.3. This component is a tool for comparing instances of managed data based on its the meta-data. As with the DMC Translator, both the STEP data repository and the native data repository will contain ‘managed data’. A prerequisite for the comparator operations is that it has access to the previous version of the data, either within its own internal data structures, or as a separate file. In the simplest case (level 1 implementation), the original CIS data exchange file will be compared with the incoming file.

7.3.1 The role of the EXPRESS language

To understand the process of incremental data import and the logic of the ‘comparator’ operation, one must first understand the basics of the EXPRESS Data Definition Language, which is discussed in APPENDIX D. This is because the ‘comparator logic’, and the rules for incremental data import developed by the CIMsteel Project, are a direct result of the EXPRESS Language and its use in the LPM.

7.3.2 ‘STEP models’

In ‘Basic CIS Translators’, the primary mechanism for data exchange is provided by ‘CIS data exchange files’. As mentioned earlier, this is equivalent to a level 1 STEP implementation form. Within the STEP data repository of any CIS Translator, data is held (at least temporarily) in a ‘working form’ that is dependent upon the STEP developer’s toolkit employed to write the translator. A collection of data in this ‘working form’ is known as a ‘**STEP model**’. When writing an IDI Translator, it is important that the software developer knows about these ‘STEP models’ and how to manipulate the data within them using the procedure calls defined in SDAI^[18].

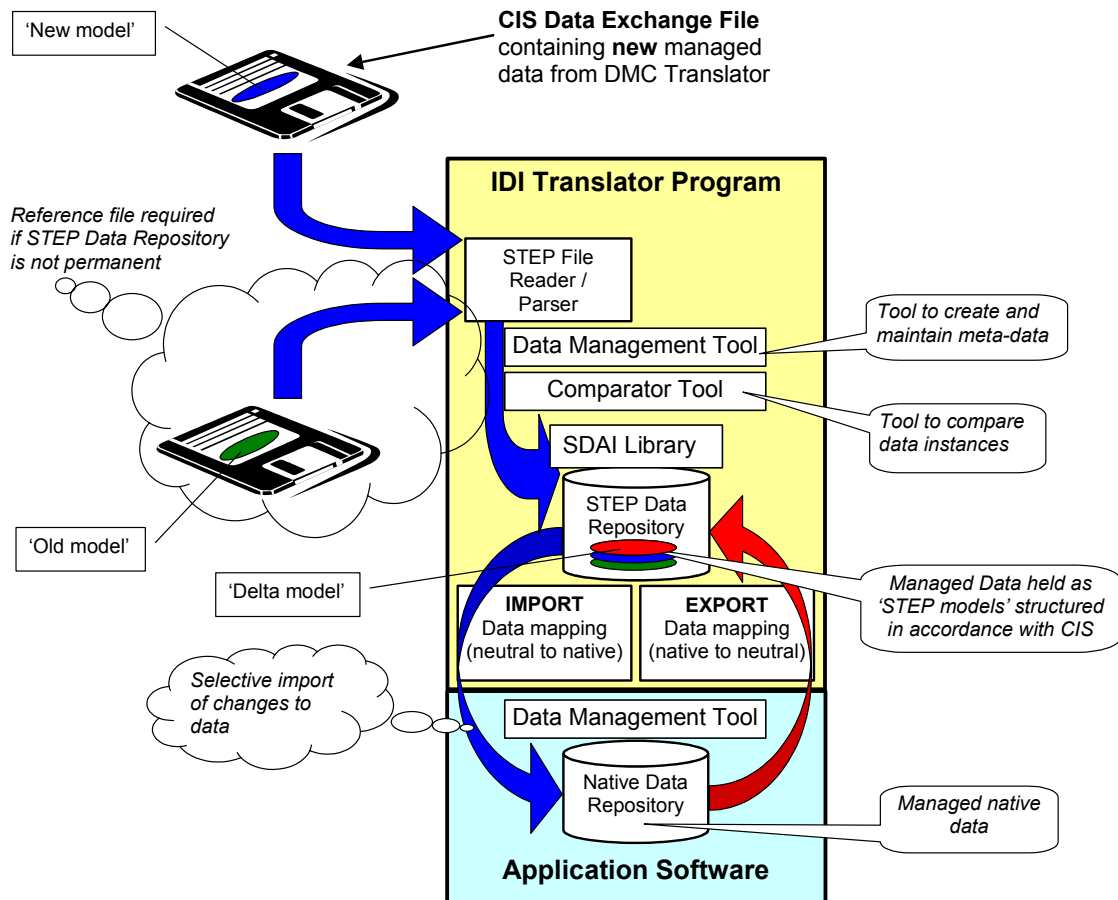


Figure 7.3 Conceptual view of an IDI Translator

7.4 The procedure for Incremental Data Import

For the purposes of this publication, the existing 'STEP model' held in the STEP repository of the IDI Translator program is referred to as the '**alpha model**' (α). Where the repository is not permanent, the alpha_model may be re-created from the original CIS data exchange file. A new 'STEP model' is created based on the data within the imported CIS data exchange file. This model is referred to as the '**beta model**' (β). The beta_model is considered to be a new version of **alpha_model**, containing several modified instances of entities that were in alpha_model. The two models (alpha_model and beta_model) are compared using the rules discussed in Section 10. A third STEP model is created by the IDI Translator program using the 'comparator tool'. This model contains only the differences between beta_model and alpha_model, and is referred to as the '**delta model**' (δ).

The purpose of the IDI Translator is import only the differences between alpha_model and beta_model and the changes that have occurred since alpha_model was last imported (in the 'master-slave' scenario) or exported (in the 'feedback' scenario). Therefore, the IDI Translator program has to establish what has changed between the two models by a detailed comparison. Where the unique identifiers in beta_model are not matched with those in alpha_model, the data items associated with them are simply added to delta_model, while those in alpha_model are ignored. Where unique identifiers are matched, the comparator has to establish whether the data items associated with them in beta_model are different from those in alpha_model or merely an exact copy. If they are

an exact copy, then the data items are ignored and not included in the delta_model. If any aspect of the data item has changed, then the whole construct is added to the delta_model.

The procedure for Incremental Data Import is based on the 'Comparator Logic', which operates on the two STEP models as discussed in Section 10.

8 PMR-ENABLED TRANSLATORS

This Section describes how Data Management Conformant CIS Translators are implemented such that they support data sharing and management through a CIS Product Model Repository (PMR) using direct interfaces. The procedures for writing ‘PMR-enabled Translators’ discussed here are *additional* to the procedures for writing DMC Translators and Basic CIS Translators. PMR-enabled import translators are also expected to fulfil the requirements of IDI translators.

When implementing a PMR or a ‘PMR-enabled Translator’, vendors need to satisfy the requirements of a Level 3 STEP implementation.

8.1 Developing PMR-Enabled Translators

In simple terms, a PMR-Enabled Translator is one that has either DMC export capabilities, or IDI import capabilities, or both. Thus, the first stage in the development of a PMR-Enabled Translator is the same as that for a Basic CIS Translator. The logic of the native-to-neutral and neutral-to-native data mappings remains valid. The second stage is to add the Data Management Tool, as described in Section 6.6. The third stage is to add the ‘Comparator’ as described in Section 7.3. Thus, by the end of the third stage the greater part of the translator coding will have already been written for an IDI Translator.

The PMR-Enabled Translator has the additional capability of being able to log on directly to a suitable PMR. (An example of an interface for a PMR-Enabled Translator is shown in Figure 8.1.)

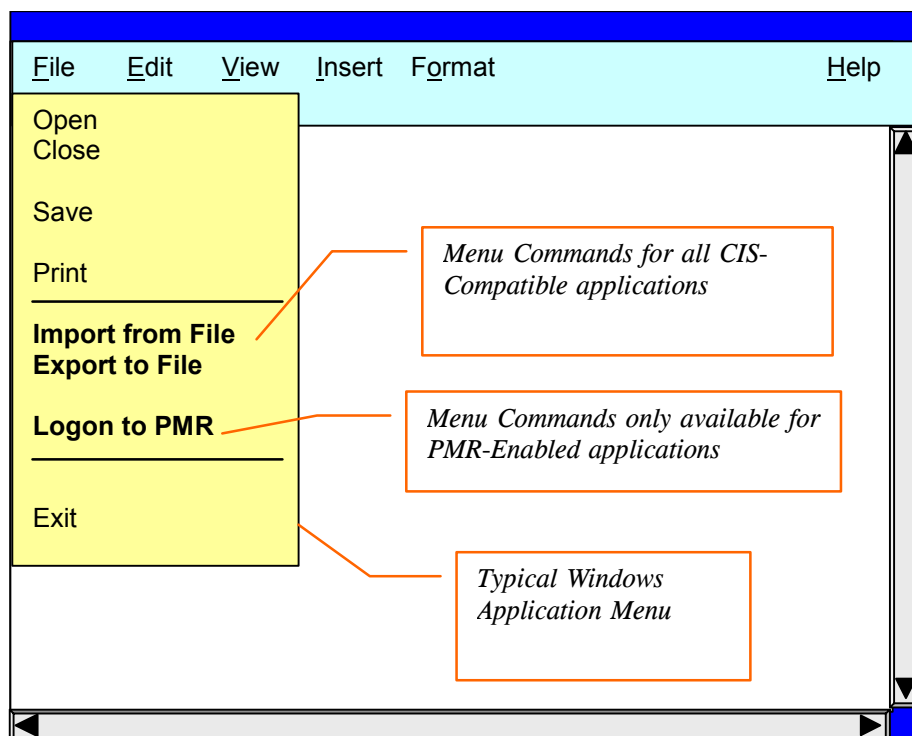


Figure 8.1 Example interface for a PMR-Enabled Translator

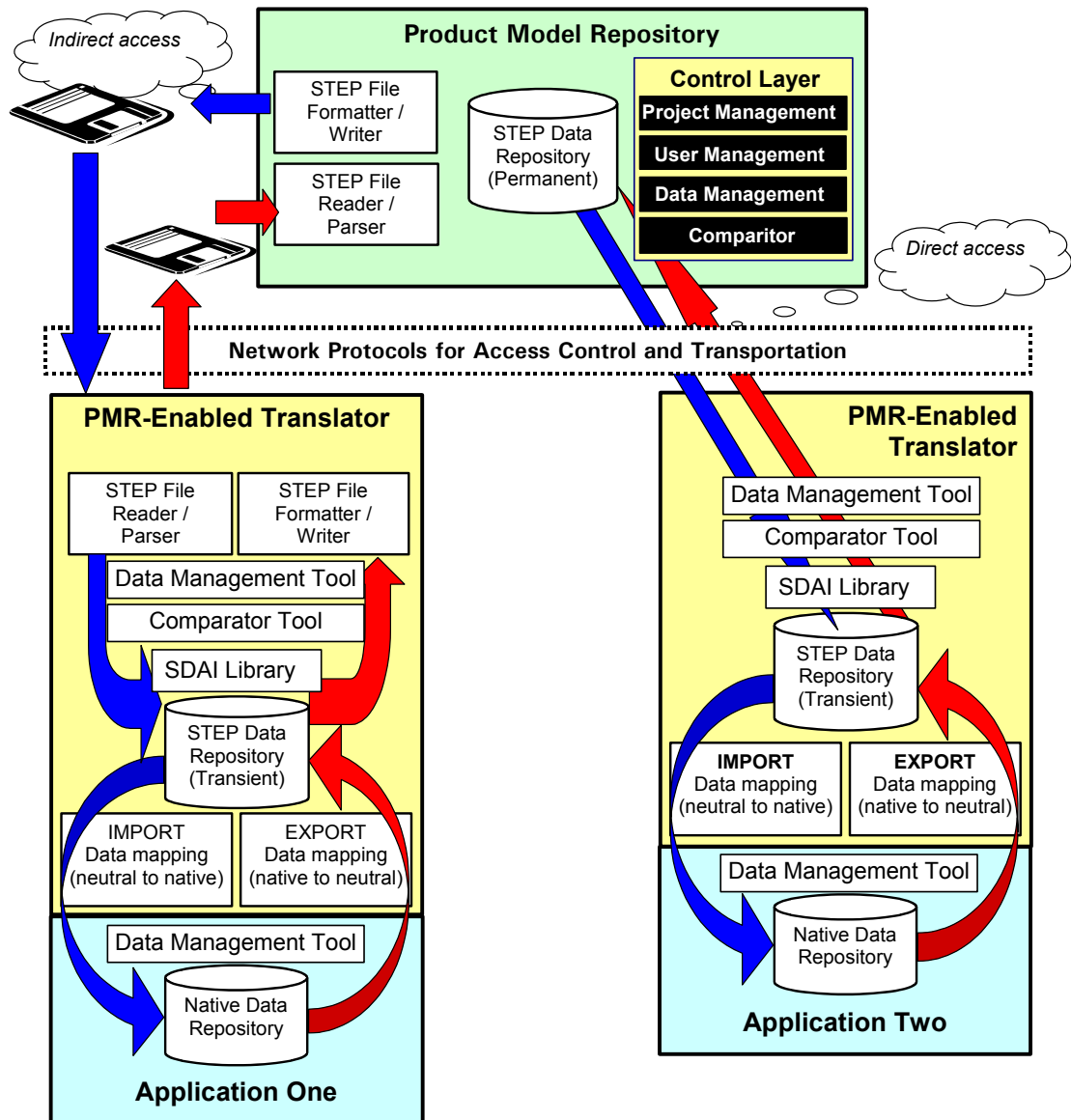


Figure 8.2 Conceptual view of PMR-Enabled Translators showing 'Indirect' access (One) and 'Direct' access (Two).

The data stored in a PMR may be accessed either *directly* or *indirectly*, as shown in Figure 8.2. Direct access requires the use of SDAI calls, while indirect access makes use of physical files. It is a prerequisite for direct access that both the PMR-Enabled Translator and the PMR use the same implementation of SDAI. In practice, this means that PMR-Enabled Translator has to be written using the same STEP developer's toolkit as was used to write the PMR. (This problem will be resolved when a 'standard' method of implementing SDAI has been agreed.)

For optimum performance, PMR-Enabled Translators (at least those seeking direct access operations) need to be developed in conjunction with a PMR. Even if the PMR is written by a third party software developer, developers of PMR-Enabled Translators should liaise closely with these third parties.

PMRs are discussed in more detail in the next Section.

9 PRODUCT MODEL REPOSITORIES

This Section describes how CIS/2 is implemented to produce a Product Model Repository (PMR). A PMR is expected to support data sharing and management using direct interfaces. A PMR should also support CIS data exchange file operations. The procedures for writing a 'PMR' discussed here build upon those for writing other types of CIS Translators.

When developing a 'Product Model Repository' or a 'PMR-enabled Translator', vendors need to satisfy the requirements of a Level 3 STEP implementation.

9.1 Developing a PMR

As shown in Figure 9.1, a PMR contains several tools. The Data Management Tool was described in Section 6.6, while the 'Comparator' Tool was described in Section 7.3. These components are discussed in more detail in the following Sections.

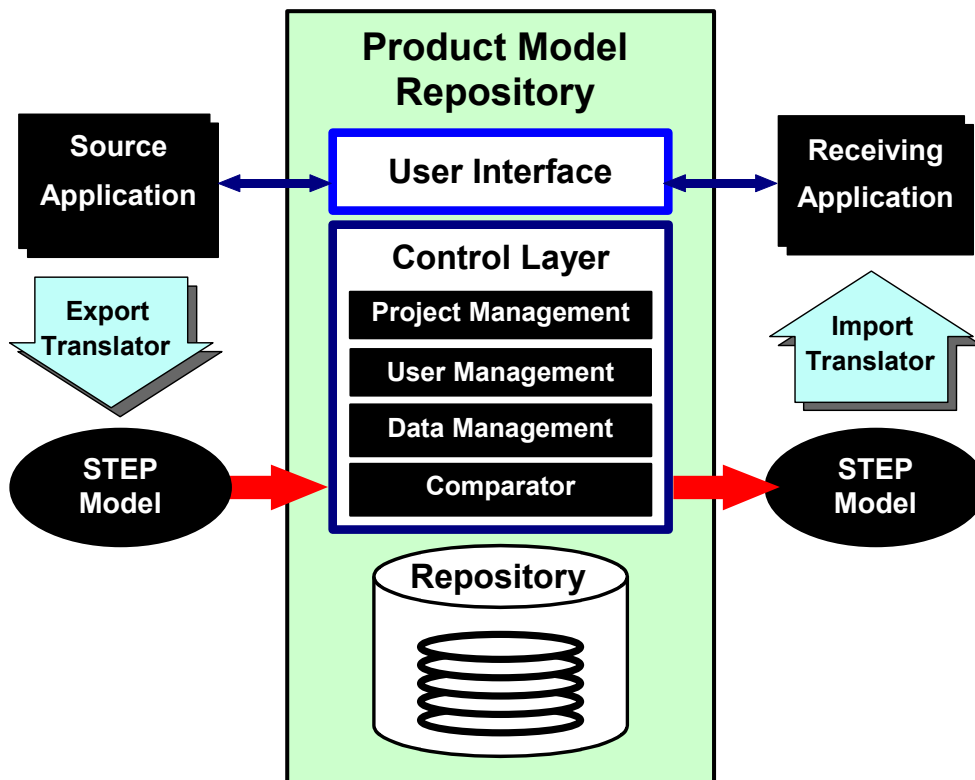


Figure 9.1 Conceptual view of a PMR

9.2 Types of PMR User

Before discussing *what* the PMR consists of and *how* it is intended to be used, this Section will explain *who* the PMR is intended to be used by.

For a CIS/2-based PMR, four types of user are defined, as follows:

- PMR Administrator

- Project Manager
- Project User
- Project Visitor

Any person can hold the status of Project Manager, Project User or Project Visitor on any project if they had been given that authority by the PMR Administrator. The different roles of each type of user are explained in the following subsections.

9.2.1 The PMR Administrator

At the highest level (i.e. with the greatest authority) is the PMR Administrator. The Administrator has read and write access to all information within the PMR. The Administrator is the only person with the authority to create or delete a “Project”. The project is the key to all data access for all other users, and is the means of classifying data within the PMR. The PMR Administrator can also create and delete data concerning any project within the PMR, but it is expected that actual data input/output will be performed at the “lower” levels. The Administrator is responsible for managing the PMR, and is therefore responsible for managing the projects above the project manager level.

9.2.2 The Project Manager

The Project Manager has read and write access to a particular project. The Manager cannot create or delete a project, but can create and delete data concerning that particular project within the PMR. Only the administrator has the authority to create the first Project Manager on a project, but once created, a Manager can create Project Users and Project Visitors for that project. There can be only one Manager for a project at any one time, but a Project Manager can appoint his successor. The Project Manager is responsible for managing a particular project, and is therefore responsible for the data entering and leaving the PMR. The Project Manager is also responsible for managing the Project Users and Project Visitors of that particular project.

9.2.3 The Project User

The Project User has read and write access to data on a particular project. The Project User cannot delete a Project, but can create and delete data concerning the particular project within the PMR, to which he is authorized.

9.2.4 The Project Visitor

The Project Visitor has *read only* access to data on a particular project. A Visitor cannot delete data within the PMR, nor can he write data to the PMR.

9.3 The components of the PMR

A PMR comprises three basic components: an **interface**, a **control layer**, and a **repository**. The functionality and use of each of these components are explained in the following subsections.

9.3.1 The PMR Interface

The PMR Interface is the only aspect of the PMR that the users will normally see, as the rest will be hidden. PMR-compatible applications should provide a common menu

command that will give access to a standard CIS user interface. It is likely that the PMR will be accessed from the user interface of the application. Where on a level 1 implementation a user clicked to launch a CIS translator (to import or export data from or to a file), the user on a level 3 implementation clicks to connect (and logon) to the PMR. An example of an interface for a CIS-compatible application is shown in Figure 8.1. This should launch the PMR interface in his workstation; i.e. the user becomes a 'client' to the PMR 'server'. Thus, the data exchange process (via the PMR) becomes closer to the data creation and manipulation process performed within the application.

Before being given access to models in the repository, the user must first logon to the PMR. A "Welcome to the PMR" dialogue box will appear (similar to that shown in Figure 9.2) prompting the user to input a PMR username and password.

Figure 9.2 Example PMR logon dialogue box

Once a user has been verified as a valid user of the PMR, the PMR main dialogue box will appear (similar to that shown in Figure 9.3). This will offer the user a number of options to choose from, depending on the type of user (i.e. PMR Administrator, Project Manager, Project User or Project Visitor). Thus, a PMR Administrator will be allowed to create and delete projects. A Project Manager will be allowed to access the User Management Tool of the PMR. Project Users and Project Visitors will only be allowed access to the Data Management Tool of the PMR (the rest will be unavailable and appear 'greyed out').

Figure 9.3 Example PMR Main dialogue box

9.3.2 The Repository

The **Repository** is where the models will actually be held by the PMR. The PMR is built on top of an existing proprietary ‘STEP developer’s toolkit’ which provides the basic SDAI calls required to exchange the STEP physical models. The toolkit is based around its own model repository - which becomes the core component of the PMR. The repository can contain a number of models. The Repository is divided into separate Project areas by the PMR Administrator. Each model belongs to only one Project.

9.3.3 The Control Layer

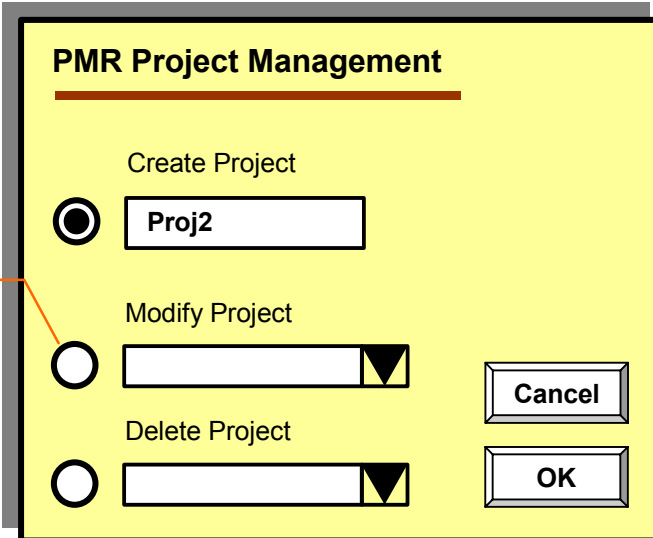
The **Control Layer** lies immediately behind the user interface and controls user access to the models and access rights to the various projects held within the PMR. It contains three main components: a *project management tool*, a *user management tool*, and a *data management tool*.

9.3.4 The Project Management Tool

The data management tool incorporates a **project management tool**. This is operated by the PMR Administrator to create, modify or delete a project. Only the Administrator has authority to use the Project Management Tool. (An example interface for the Project Management Tool is shown in Figure 9.4.)

The Project Management Tool functions

1. Create a new Project.
2. Delete an existing Project.
3. Modify information associated with an exiting Project (e.g. project description).



The dialog box is titled "PMR Project Management" and has a yellow background. It contains three radio buttons for "Create Project", "Modify Project", and "Delete Project". The "Create Project" radio button is selected, and its corresponding text box contains "Proj2". The "Modify Project" and "Delete Project" radio buttons are unselected, and their corresponding text boxes are empty. To the right of the "Modify Project" and "Delete Project" text boxes are "Cancel" and "OK" buttons respectively. A callout box with an orange border and an arrow pointing to the radio buttons contains the text "Alternatives - depending on the radio button selected".

Figure 9.4 Example PMR Project Management dialogue box

9.3.5 The User Management Tool

The User Management Tool is operated by the PMR Administrator to give people rights as either project managers, users or visitors. A Project manager may also use the User Management Tool to give people rights as either users or visitors or to appoint his successor as project manager. User rights may be added, modified or removed from

people via the User Management Tool. An example interface for the User Management Tool is shown in Figure 9.5.

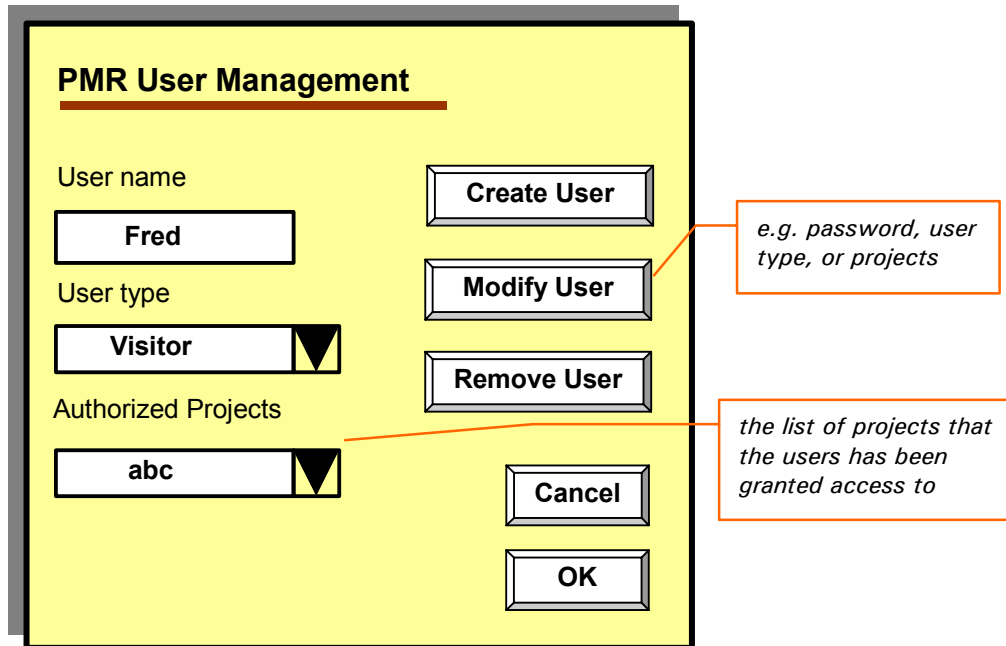


Figure 9.5 Example PMR User Management dialogue box

The User Management Tool functions

1. **Add** new Manager / User / Visitor
2. **Modify** rights of a Manager / User / Visitor
3. **Remove** a Manager / User / Visitor

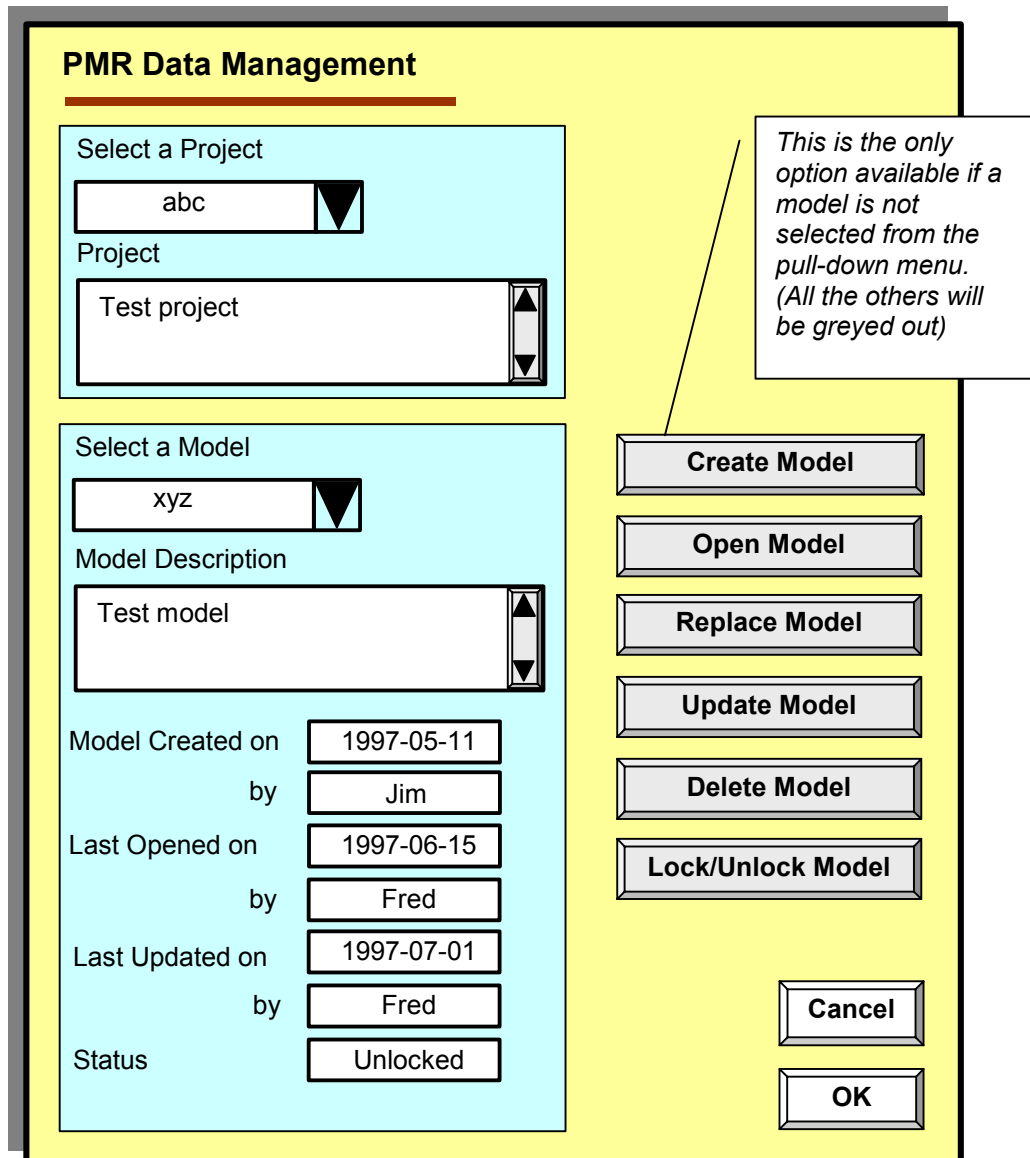
9.3.6 The Data Management Tool

The Data Management Tool is normally operated by Project Users to export engineering data from an application and place the data in the repository of the PMR. Project Visitors may only operate the PMR's Data Management Tool to import engineering data from the PMR's repository into their application. An example interface for the Data Management Tool is shown in Figure 9.6.

The Data Management Tool functions

Any of the following four functions of the Data Management Tool may be performed either **directly** or **indirectly**. Only applications that have translators compatible with the PMR (i.e. PMR-enabled translators) can operate **directly** by passing STEP physical models between application and the PMR. Applications that do not have translators compatible with the PMR must operate **indirectly** by passing data via STEP physical files. This is illustrated in Figure 8.2.

1. **Create Model** - this exports engineering data from an application and places it in a new STEP physical model held in the repository. The new STEP physical model is placed within the repository under the Project defined earlier.
2. **Open Model** - this imports engineering data (in the form of an existing STEP physical model) from the repository into the application.



PMR Data Management

Select a Project

abc

Project

Test project

Select a Model

xyz

Model Description

Test model

Model Created on 1997-05-11

by Jim

Last Opened on 1997-06-15

by Fred

Last Updated on 1997-07-01

by Fred

Status Unlocked

Create Model

Open Model

Replace Model

Update Model

Delete Model

Lock/Unlock Model

Cancel

OK

This is the only option available if a model is not selected from the pull-down menu. (All the others will be greyed out)

Figure 9.6 Example PMR Data Management dialogue box

3. **Replace Model** - this exports engineering data from an application and places it in an existing STEP physical model held in the repository. Effectively, all the data in the STEP physical model is overwritten by the new data after a backup copy of the old model has been made.
4. **Update Model** - this exports engineering data from an application and places it in an existing STEP physical model held in the repository. There are two ways to update a model: *'Update & Maintain Data'*, and *'Update & Replace Data'*. The latter is intended to eliminate redundant data from the model. Before a new model is created, a backup of the old model is created. In both cases, the incoming model is compared with the existing model held in the repository. This updating process involves several levels of comparison. The rules and procedures for *'Incremental Updating'* are highly complex and are explained in Section 10 of this publication. An example "Update" dialogue box is shown in Figure 9.7.

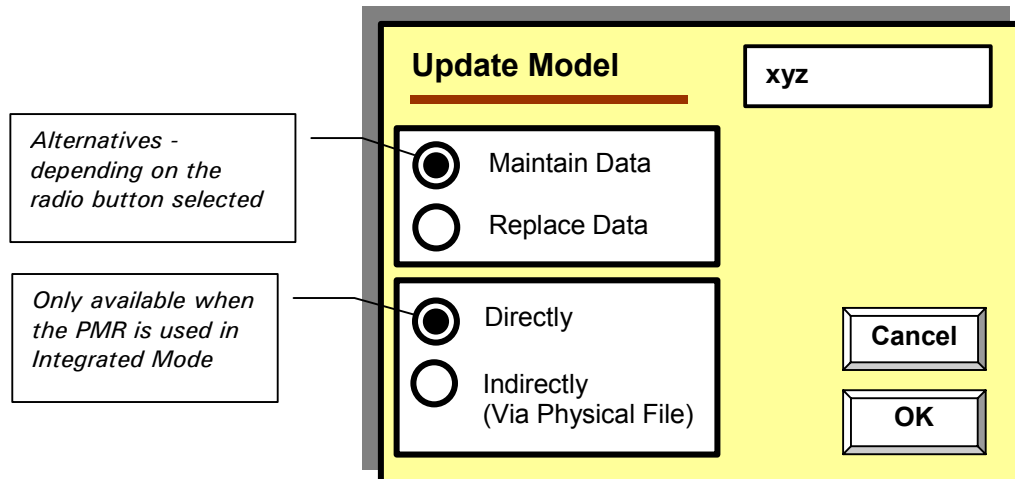


Figure 9.7 Example dialogue box for updating models

It should be noted that the updating operations will only be available for models containing managed data (since the comparator tool can only work with managed data). In common with DMC Translators, a PMR should be able to import unmanaged data and add the required meta-data to produce (and store) managed data.

In addition to the four Data Management Tool functions discussed above, applications with PMR-enabled translators operating directly will also have the following functions

5. **Delete Model** – This removes the selected STEP physical model from the repository. All data contained in the model will be lost.
6. **Lock / Unlock Model** - This controls access to models held in the PMR in either of two ways (as shown in the example dialogue box in Figure 9.8):
 - **Lock Read and Write Access** - this gives a Project Manager the ability to prevent anyone else reading from, or writing to, the model.
 - **Lock Write Access** - this gives a Project Manager the ability to prevent anyone else writing to a model.

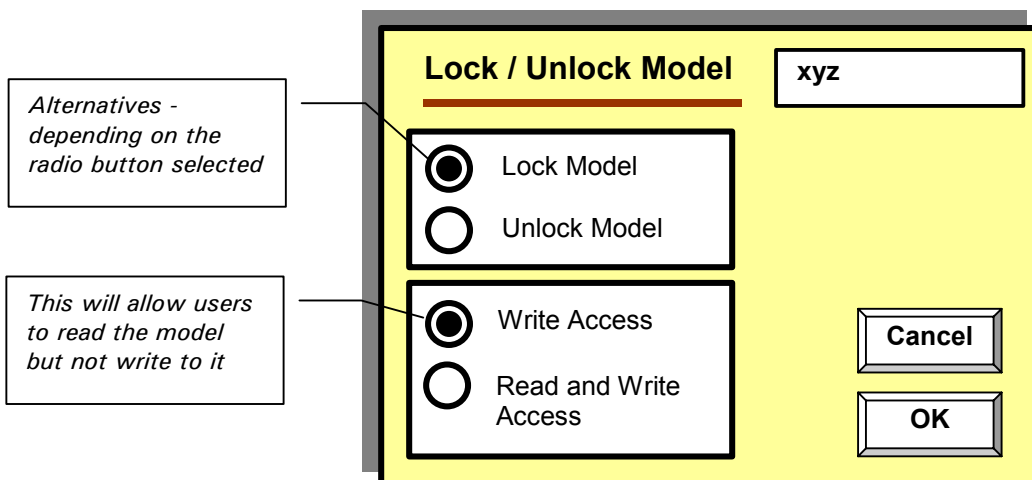


Figure 9.8 Example dialogue box for locking and unlocking models

9.4 Using the PMR

There are two modes of operating the PMR: ‘stand-alone’ and ‘integrated’.

9.4.1 ‘Stand-alone’ mode

In the stand-alone mode, an administrator can create or delete projects, or add and remove users. This is illustrated in Figure 9.9. Effectively, the management of the users and the management of existing data within the PMR would be done in stand-alone mode. Although the data management facilities are limited, an additional use of the ‘stand-alone’ mode would be for the import and export of STEP physical files, facilitating *indirect* data exchange (see later). However, real data can only be placed *directly* into the PMR from an application when the PMR is operating in an integrated mode.

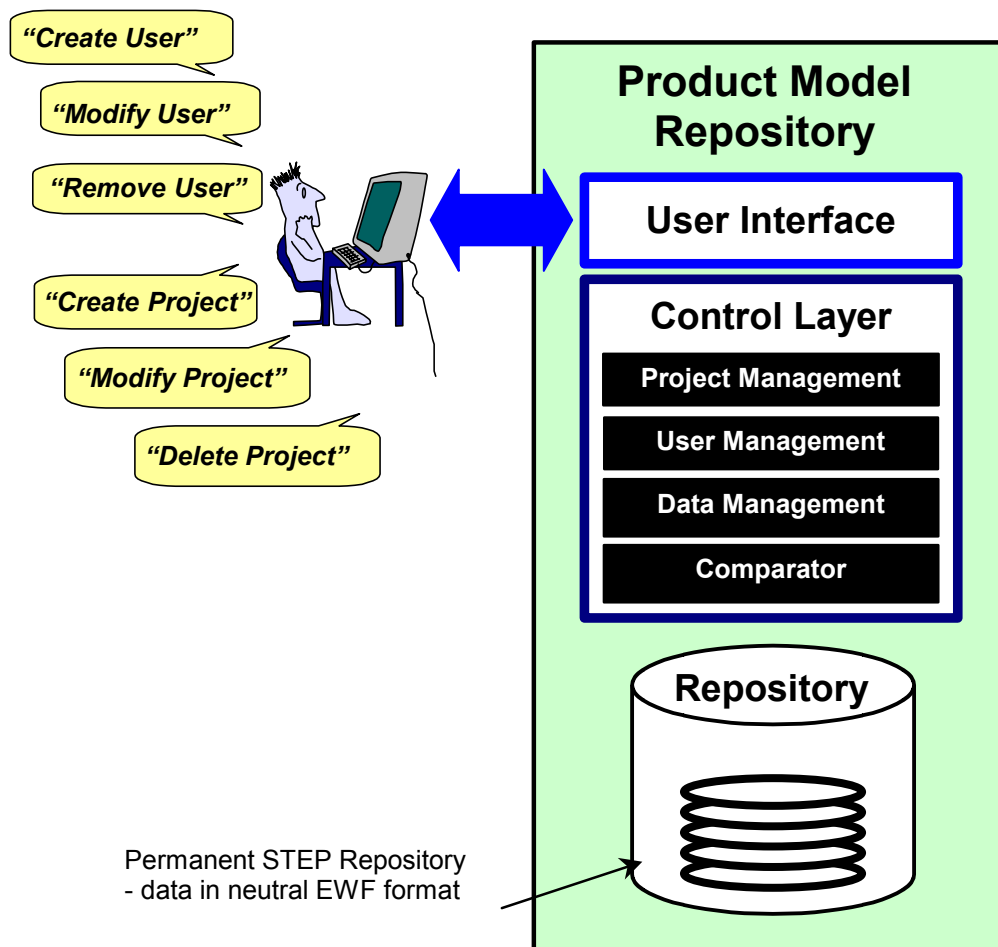


Figure 9.9 PMR in ‘stand-alone’ mode

9.4.2 ‘Integrated’ mode

When the PMR is used in ‘integrated mode’ (illustrated in Figure 9.10), the interface of the PMR will be launched from the interface of the application. Access to and from the **repository** will be controlled by the **control layer**. The control layer establishes what type of user the person is, and what access rights that user has to the various projects held within the PMR.

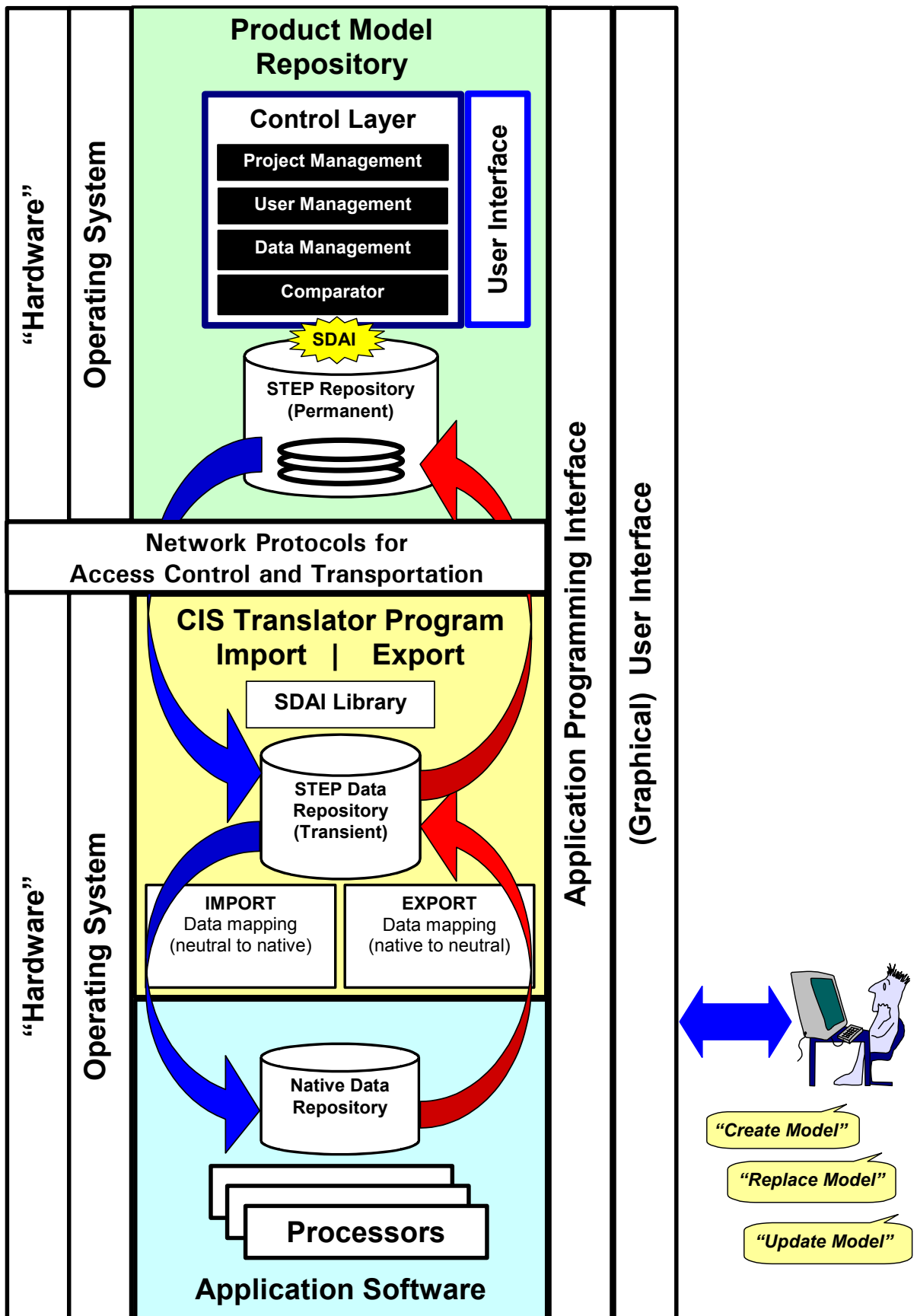


Figure 9.10 PMR in 'integrated' mode

Because the control layer effectively sits between the PMR interface and the application's own Graphical User Interface (GUI), it remains invisible to the user, and to the user of an application, the PMR is accessed through the GUI of the application (see Figure 8.1). The control layer is used to establish what type of user the person is and what access rights that user has to the various projects held within the PMR. The control layer uses SDAI calls to manipulate STEP physical models between the application and the PMR. The STEP physical models are held within the repository in the proprietary form of the toolkit, which is based upon EXPRESS working form (EWF).

All PMR-Enabled Translators and the PMR utilize SDAI to create and manipulate STEP physical models. Each application has a translator that is called upon by the PMR's control layer to create the STEP physical model. Once the STEP physical model has been checked as suitable for import into the repository, the control layer then pulls the model into the repository. Import of STEP physical models from the PMR into the receiving application is the reverse of the export.

9.4.3 Direct and Indirect Opening of Models

The PMR allows two ways of accessing data; i.e. *directly* and *indirectly*. (This is illustrated in Figure 8.2.) Thus, once a user has logged onto the PMR and has been verified, when the user selects a model held within the repository he is given the option to open the model *directly* or *indirectly*. Opening the model indirectly forces the PMR to create a STEP physical file. This can then be passed onto other users (perhaps in other companies) via a network or on disk.

If a user **creates** a model **indirectly**, this launches the file manager dialogue box of the PMR, asks the user to select a file, and forces the PMR to import the selected physical file into the repository. The physical file is effectively converted into a STEP physical model.

Similarly, if a user **opens** a model **indirectly**, this launches the file manager dialogue box of the PMR, and forces the PMR to export the selected model from the repository to a physical file.

Finally, when a user **updates** a model **indirectly**, this launches the file manager dialogue box of the PMR, asks the user to select a file, and forces the PMR to import data from the selected physical file into the selected model in the repository.

9.5 Using PMRs together

It does not necessarily follow that there has to be just one PMR. In fact, it is more likely that several PMRs will be in use at the same time. Each PMR should be able to interact with another in the same way that an application can interact with a single PMR. Where there are several PMRs, each is treated as an application (because that is what it is). Data sharing could take place between PMRs either directly or indirectly, using procedure calls, or physical files.

10 THE ‘COMPARATOR’

This Section describes the rules that are used with the ‘Comparator Tool’ required for IDI Translators, PMR-Enabled Translators, and PMRs to perform certain operations. Depending on the situation, there are effectively three sets of rules:

1. Rules for Incremental Data Import (for IDI Translators)
2. Rules for Update & Maintain operations (for PMRs)
3. Rules for Update & Replace operations (for PMRs)

It should be noted that the comparator tool can only work with managed data. If the comparator encounters unmanaged data it should return an appropriate error.

This Section also requires an in-depth understanding of the EXPRESS language (as described in APPENDIX D) and the physical file format (as described in APPENDIX F).

10.1 The tasks of the comparator

The principal task of the comparator is to compare two sets of data, each of which is contained within a separate STEP model. The comparator must be able to ‘recognize’ that one of the models contains new versions of data contained in the other model. It is able to do this by using the unique identifiers provided for every instance of every ‘data entity’ contained in the ‘meta-data entities’ in LPM/6. Once the comparator has compared the two models, it then has to create a new model. The data contained in the new model will depend on the type of operation that has been requested.

10.1.1 The STEP models

For the purposes of this publication, the existing STEP model held in the STEP repository of the IDI Translator program or PMR is referred to as the ‘alpha_model’ (α). (Where the repository of an IDI Translator is not permanent, the alpha_model may be re-created from the original CIS data exchange file.) On import, a new ‘STEP model’ is created based on the data within the imported CIS data exchange file (for level 1 implementations). This model is referred to as the ‘beta_model’ (β). The beta_model is considered to be a new version of the alpha_model, containing several modified instances of entities that were in the alpha_model. The two models (alpha_model and beta_model) are compared using the rules discussed below.

The IDI Translator program uses the ‘comparator tool’ to create a new STEP model, which contains only the differences between the beta_model and the alpha_model, and is referred to as the ‘delta_model’ (δ).

A PMR uses the ‘comparator tool’ to create a new STEP model, which contains an updated version of the alpha_model modified to incorporate the changes contained in the beta_model. Depending on the type of update requested, the model stored in the repository is referred to as either the epsilon_model (ϵ), for update & maintain operations, or the zeta_model (ζ) for update & replace operations.

10.1.2 Incremental Data Import

‘Incremental Data Import’ is the ability of a software application to add new information to an existing information set within the application (via the repository of the IDI Translator) while maintaining the integrity of the information set.

The purpose of the IDI Translator is to import *only* the differences between α and β , which represent the changes that have occurred since α was last imported (in the ‘master-slave’ scenario) or exported (in the ‘feedback’ scenario). Therefore, the IDI Translator uses the comparator to establish what has changed between the two models by a detailed comparison. Where the unique identifiers in β are not matched with those in α , the data items associated with them are simply added to δ . The data items associated with the unmatched identifiers in α are ignored. Where unique identifiers are matched, the comparator has to establish whether the data items associated with them in β are different from those in α or merely an exact copy. If they are exact copies, the data items are ignored and not included in δ . If any aspect of the data item has changed, then the whole construct is added to δ . If a data item references another data item that has changed, then the whole construct is added to δ .

The new model (δ) contains the changes to the data that have occurred between α and β , and is used to selectively import data into the application. By default, Basic CIS Translators operate in a ‘displace’ mode - such that any existing data within the application will be deleted at the start of the import process. In contrast, IDI Translators operate in an ‘additive’ mode - such that some of the existing data within the application is preserved during the import process, while other data is replaced or updated. The purpose of the IDI Translator is to import only the new data and to ignore the unchanged data.

At the most basic level of IDI operations, the comparator will simply replace all data having matching identifiers. More advanced implementations may give users the option to accept or reject the changes that the comparator finds in the data.

10.1.3 Update & Maintain operations

‘Update & Maintain’ operations add new information to an existing information set within the repository of a PMR, while maintaining the integrity of the information set. This operation allows applications of differing but overlapping scope to share the common information (via a PMR) without destroying the information that lies beyond the scope of one or other of the applications. The scope of the information set is determined by the Conformance Classes supported by the application and populated (or instanced) in α or β . Where the scope of β is smaller than that of α , ‘Update & Replace’ operations will not be available.

The purpose of ‘Update & Maintain’ operations is to allow applications to share partial information.

10.1.4 Update & Replace operations

‘Update & Replace’ operations add new information to an existing information set within the repository of a PMR, while maintaining the integrity of the information set. This operation allows applications whose scopes match exactly to share information (via a PMR) while eliminating redundant information. So that ‘Update & Replace’ operations do not destroy useful information, the scope of the incoming information set (contained in β) must be greater than, or equal to, the scope of the information set (contained in α)

held in the repository. The scope of the information set is determined by the Conformance Classes supported by the application and populated (or instanced) in α or β . The Log Files generated by the PMR and the PMR-enabled translators will assist in making this check.

The purpose of ‘Update & Replace’ operations is to eliminate redundant information from data sets that have been updated from partial data.

10.1.5 Establishing the ‘changes’

Before ‘new’ information is added to a model, the IDI Translator program or the PMR should check that it really is new. To do this, the unique identifiers from the two models are compared. If the same unique identifier appears in both alpha and beta models then the data associated with the `managed_data_item` instance in beta is not new but a modified version of the equivalent data in alpha.

10.2 The ‘Comparator Logic’

The ‘Comparator Logic’ operates on the two STEP models (α and β) as follows:

1. Compile a list of instances of ‘`managed_data_item`’ for both `alpha_model` and `beta_model`. (If there are no instances of `managed_data_item` in both models, then the data is unmanaged and the comparator should stop and display an appropriate error message.)
2. Compile a list of ‘unique identifiers’ for both models. (If any ‘unique identifiers’ are repeated within an individual model, the identifiers are not unique and the comparator should stop and display an appropriate error message.)
3. For each value of each unique identifier in `beta_model`, attempt to find an exact match in `alpha_model`. If there are no matches, it is assumed that all the data in the `beta_model` represents entirely new information. In this case, there is nothing to compare, and the following rules apply:
 - δ the `beta_model` becomes the `delta_model` (for the IDI Translator).
 - ϵ the unique identifiers (and the accompanying data) in the `beta_model` are simply added to the data in the `alpha_model` to create the `epsilon_model` (for the PMR),
 - ζ the unique identifiers (and the accompanying data) in the `beta_model` are simply added to the data in the `alpha_model` to create the `zeta_model` (for the PMR).
4. For each value of each unique identifier in `alpha_model`, attempt to find an exact match in `beta_model`. If a unique identifier exists in `alpha_model` but not in `beta_model` the following rules apply:
 - δ the unique identifiers (and the accompanying data) in the `alpha_model` are ignored (for the IDI Translator).
 - ϵ the unique identifiers (and the accompanying data) in the `alpha_model` are maintained in the `epsilon_model` (for the PMR),

- ζ the unique identifiers (and the accompanying data) in the alpha_model are effectively deleted and do not appear in the zeta_model (for the PMR).
5. Where values of unique identifiers are matched, it is assumed that the associated data items are intended to represent the same ‘real world thing’. The data contained in beta_model may or may not be a modified version of the data contained in alpha_model. The comparator tool then compares the instance of the entity that is associated with the unique identifiers.
 6. Using the attribute ‘data_item’, find the instance of the entity for which this instance of managed_data_item provides a unique identifier in both alpha_model and beta_model.
 7. Check whether both data_items are referring to the same entity type. (If they do not, there is an error and the comparator should stop and display an appropriate error message. The exception to this is where the entity type results in a complex entity instance. For example, the entity type in beta_model could be a SUBTYPE of the entity type in alpha_model; here the new model contained a more specialized version of the information in alpha_model – this is discussed later. *Remember – an instance of a SUBTYPE is an instance of the SUPERTYPE.*)
 8. If both data_items are referring to the same entity type, the value of each attribute of the instance of the entity in the beta_model is compared with its equivalent value in the alpha_model.
 9. Once all the attributes of the entities with matching identifiers have been compared and their values updated, the new model is complete.

In the Second Edition of CIS/2, instances of managed_data_item may be created without corresponding instances of managed_date_transaction. This allows vendors of CIS/2 translators to add minimal data management facilities to their applications with minimal effort by creating and maintaining a unique identifier for each data instance based on the machine-generated GUID. Therefore, two flowcharts for the comparator operations are present here; Figure 10.1 should be used where data history is NOT being captured, while Figure 10.2 should be used where data history is being captured.

Table 10.1 gives a summary of the rules that should be used when comparing attribute values. These comparator rules are then specified in detail in Section 10.3. Table 10.1 and the rules for comparator operations apply whether data history is being captured or not.

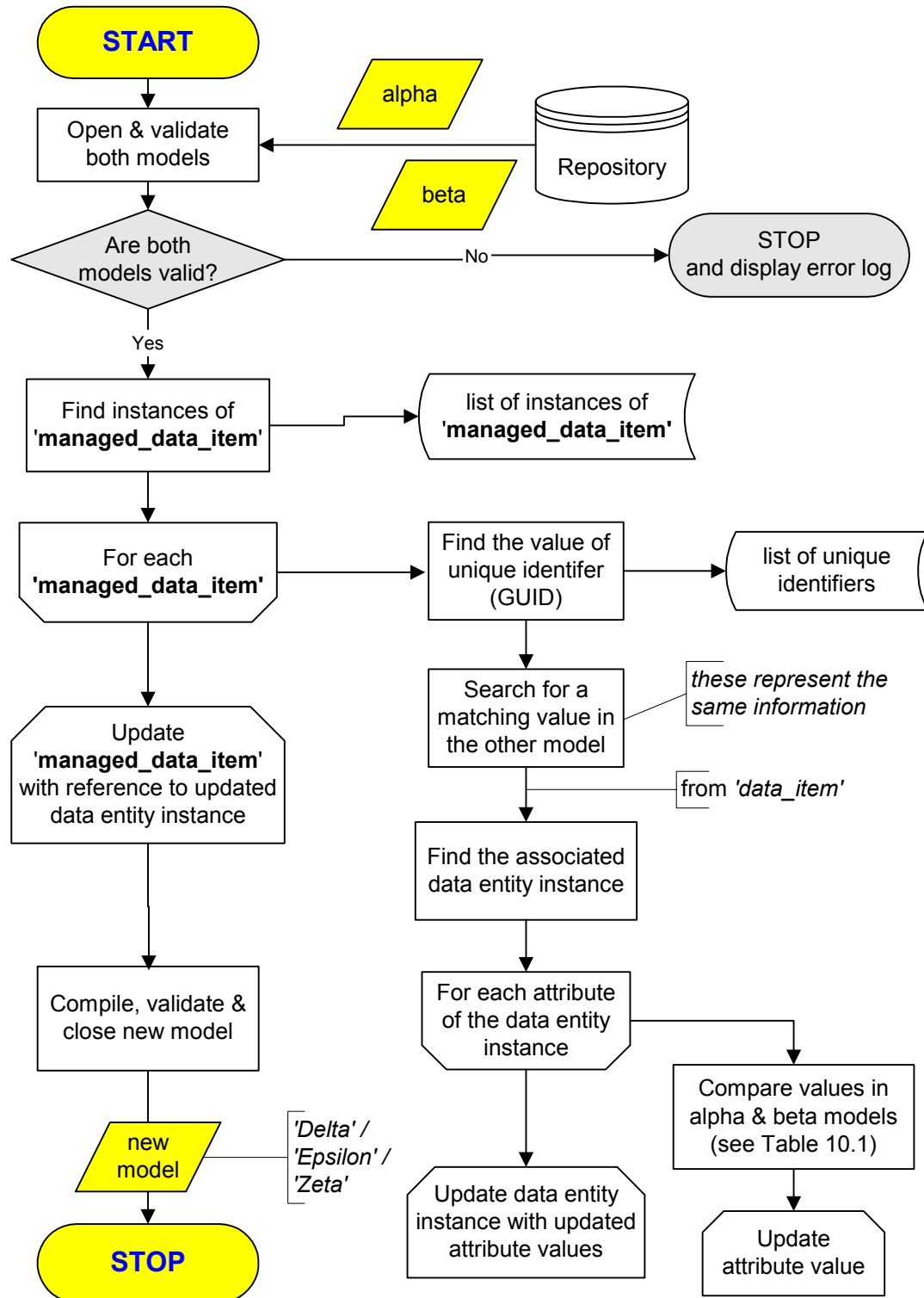


Figure 10.1 Flowchart for the logic of comparator operations (without history)
New for 2nd Edition

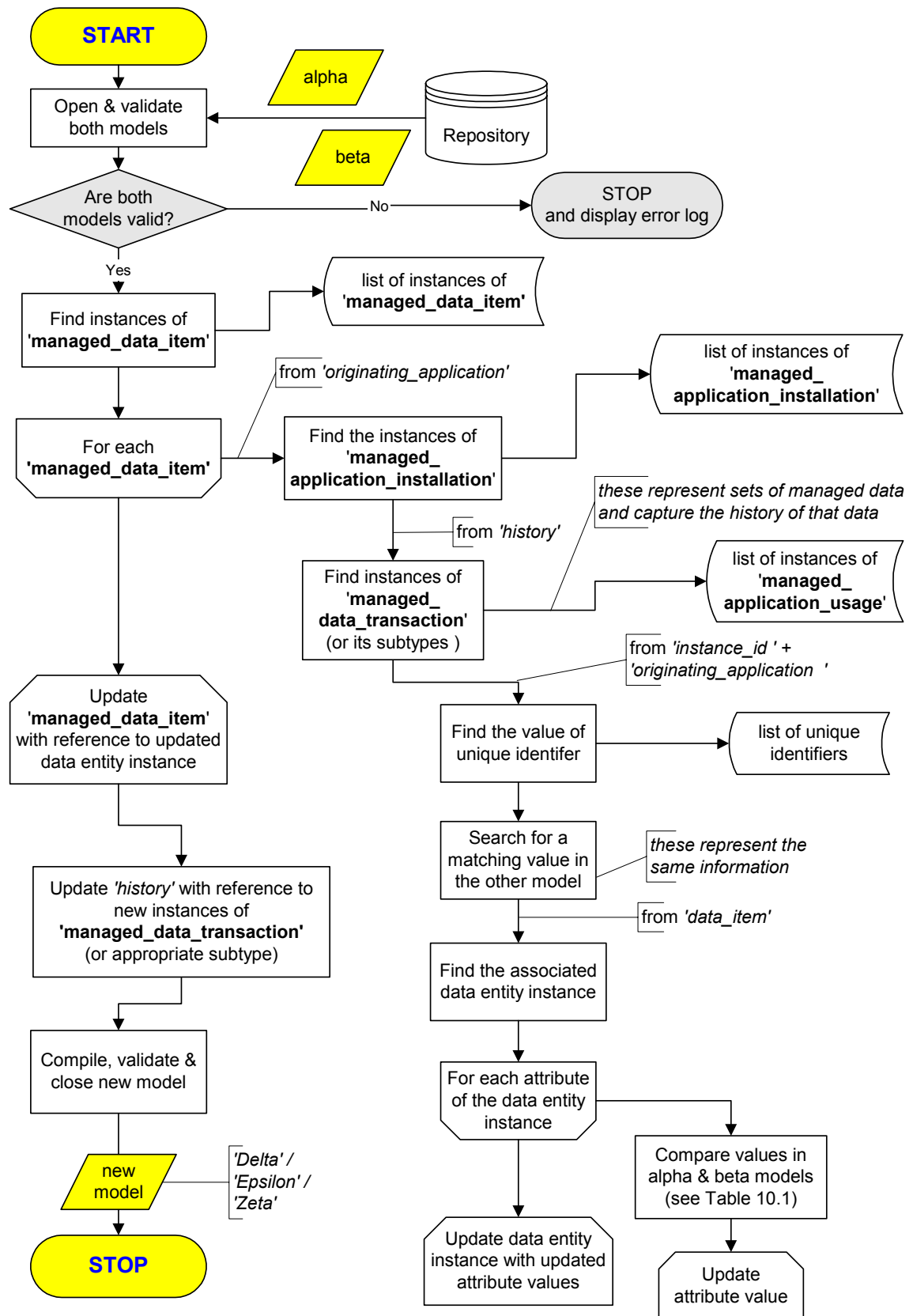


Figure 10.2 Flowchart for the logic of comparator operations (with history)

Table 10.1 *Summary of the ‘Comparator’ rules*

Comparing	Alpha_model	Beta_model	Delta_model	Epsilon_model	Zeta_model
1) Identifier	Present	Not present	Ignore β	Maintain α	Delete α
	Not present	Present	Add Identifier and associated data from β		
	Present	Present	Compare Attribute Values of data instance (2)		
2) Attribute Value	NULL (\$)	NULL (\$)	Maintain NULL (\$) value		
	Present	NULL (\$)	NULL (\$)	Maintain α value	NULL (\$)
	NULL (\$)	Present	Add new value from β		
	Present	Present	Depends on base_type (3)		
3) Base_type					
Simple_type			Replace old value in α with new value from β		
Entity_type			Replace entity reference in α with that in β		
Enumeration_type			Replace ENUMERATION item in α with that in β		
Select_type			Depends on base_type (3)		
Aggregation_type			Depends on aggregation_type (4)		
4) Aggregation Type			<i>(In all cases below, also follow rules for base_type (3))</i>		
ARRAY [m:n] OF [UNIQUE] base_type			Replace all elements of α ARRAY with those in β by index		
ARRAY [m:n] OF OPTIONAL [UNIQUE] base_type	NULL (\$)	NULL (\$)	NULL (\$)	NULL (\$)	NULL (\$)
	Present	NULL (\$)	NULL (\$)	Maintain α value	NULL (\$)
	NULL (\$)	Present	Add new value for element from β by index [Checking UNIQUENESS of element in ARRAY]		
	Present	Present	Replace element of α ARRAY with equivalent element in β ARRAY by index [Checking UNIQUENESS of element in ARRAY]		
LIST [m:n] OF base_type			Replace all elements of α LIST with those in β maintaining the positioning sequence of β		
LIST [m:?] OF [UNIQUE] base_type			Add all [unique] elements of β LIST to those in α	Add all [unique] elements of β LIST to those in α	Replace all elements of α LIST with those in β
BAG [m:n] OF base_type			Replace all elements of α BAG with those in β		
BAG [m:?] OF base_type			Add all elements of β BAG to those in α .	Add all elements of β BAG to those in α .	Replace all elements of α BAG with those in β .
SET [m:n] OF base_type			Replace all elements of α SET with those in β		
SET [m:?] OF base_type			Add all unique elements of β SET to those in α	Add all unique elements of β SET to those in α	Replace all unique elements of α SET with those in β

10.3 Comparing attribute values

When the Comparator encounters matches between the values of unique identifiers, it is assumed that the associated data items are intended to represent the same ‘real world thing’. The logic of the comparator tool used to compare the attribute values of the two instances of the entity associated with the unique identifiers is discussed in the following Section and summarized in Table 10.1. What happens depends on:

1. Whether the attribute is OPTIONAL
2. The data type associated with the attribute, and
3. The type of operation the comparator has been asked to perform.

The rules discussed in the following subsections apply to an instance of a data entity in the *beta_model* that has an equivalent instance in the *alpha_model*. This equivalence is determined by the matching unique identifiers associated with each instance of managed_data_item. The managed_data_item references an instance of a data entity through the *data_item* attribute.

To illustrate the rules of the comparator logic, the following Sections contain sample extracts from STEP Part 21 files. Both the alpha and beta models contain instances of the data management constructs from LPM/6.

The data instances in the *alpha model* (α) are preceded by the following ‘meta-data’ instances. The instances of ‘managed_data_item’ refer back to these instances.

α #5002 = MANAGED_APPLICATION_INSTALLATION (101, 'GT Strudal', 'version 3a',
 'Analysis and design for steel structures', #2101, 11, 'LU installation', #2130);
 #2101 = ORGANIZATION ('101', 'Georgia Institute of Technology', 'Software Developers');
 #2130 = ORGANIZATION ('130', 'University of Leeds', 'Public Educational Institute');
 #2030 = PERSON ('30', 'Watson', 'Alastair', ('S'), ('Dr'), ('BTech', 'PhD', 'CEng', 'MICE',
 'Senior Lecturer'));
 #2230 = PERSON_AND_ORGANIZATION (#2030, #2130);

 #5003 = MANAGED_DATA_CREATION (#5002, #2230, #3015, .T., 'preliminary', \$);
 #5004 = MANAGED_DATA_EXPORT (#5002, #2230, #3017, .F., 'preliminary', \$, #5010);
 #5010 = STEP_FILE ('dmc test', 'test of data management', 'C:\my
 documents\testcase2.stp', 'STP', #3015, 'STEP Part 21 file', (#2230), (#2230));

 #3006 = COORDINATED_UNIVERSAL_TIME_OFFSET (1, \$, .AHEAD.);
 #3011 = CALENDAR_DATE (1999, 30, 5);
 #3014 = LOCAL_TIME (10, 30, 5.90, #3006);
 #3015 = DATE_AND_TIME (#3011, #3014);
 #3016 = LOCAL_TIME (10, 35, 10.0, #3006);
 #3017 = DATE_AND_TIME (#3011, #3016);

The above ‘meta-data’ states that at half past ten on the 30th May 1999, Alastair Watson of the University of Leeds, created a set of managed data using the Leeds University copy of GT-Strudal version 3a (supplied by “Georgia Institute of Technology”. Five minutes later, he exported the data as a STEP file named “testcase2.stp”. All the instances of ‘managed_data_item’ have been placed in the two sets of managed data represented by the instances #5003 and #5004. The unique identifier of each instance in

these sets includes the ‘installation_id’ (= 101) and the ‘application_id’ (= 11) derived from the associated ‘managed_application_installation’ (instance #5002).

The data instances in the **beta model (β)** are preceded by the following ‘meta-data’ instances. The first section is carried forward from the alpha model. The instances of ‘managed_data_item’ refer back to both old and new instances of the meta-data.

β

```
#5002 = MANAGED_APPLICATION_INSTALLATION (101, 'GT Strudal', 'version 3a',
      'Analysis and design for steel structures', #2101, 11, 'LU installation', #2130);
#2101 = ORGANIZATION ('101', 'Georgia Institute of Technology', 'Software Developers');
#2130 = ORGANIZATION ('130', 'University of Leeds', 'Public Educational Institute');
#2030 = PERSON ('30', 'Watson', 'Alastair', ('S'), ('Dr'), ('BTech', 'PhD', 'CEng', 'MICE',
      'Senior Lecturer'));
#2230 = PERSON_AND_ORGANIZATION (#2030, #2130);

#5003 = MANAGED_DATA_CREATION (#5002, #2230, #3015, .T., 'preliminary', $);
#5004 = MANAGED_DATA_EXPORT (#5002, #2230, #3017, .F., 'preliminary', $, #5010);
#5010 = STEP_FILE ('dmc test', 'test of data management', 'C:\my
      documents\testcase2.stp', 'STP', #3015, 'STEP Part 21 file', (#2230), (#2230));

#3006 = COORDINATED_UNIVERSAL_TIME_OFFSET (1, $, .AHEAD.);
#3011 = CALENDAR_DATE (1999, 30, 5);
#3014 = LOCAL_TIME (10, 30, 5.90, #3006);
#3015 = DATE_AND_TIME (#3011, #3014);
#3016 = LOCAL_TIME (10, 35, 10.0, #3006);
#3017 = DATE_AND_TIME (#3011, #3016);

new
#5020 = MANAGED_APPLICATION_INSTALLATION (201, 'QSE Space', 'version 7',
      'Analysis and design for steel structures', #2125, 121, 'SCI only copy', #2129);
#2125 = ORGANIZATION ('125', 'Research Engineers (Europe) Ltd', 'Software Engineers');
#2129 = ORGANIZATION ('129', 'Steel Construction Institute', 'Member based Educational
      Institute');
#2029 = PERSON ('29', 'Crowley', 'Andrew', ('John'), ('Dr'), $);
#2229 = PERSON_AND_ORGANIZATION (#2029, #2129);

#5021 = MANAGED_DATA_IMPORT (#5020, #2229, #3023, .F., 'preliminary', $, #5010);
#5010 = STEP_FILE ('dmc test', 'test of data management', 'C:\my
      documents\testcase2.stp', 'STP', #3015, 'STEP Part 21 file', (#2230), (#2230));
#5022 = MANAGED_DATA_MODIFICATION (#5020, #2229, #3025, .F., 'preliminary', $);

#3021 = CALENDAR_DATE (1999, 16, 6);
#3022 = LOCAL_TIME (11, 30, $, #3006);
#3023 = DATE_AND_TIME (#3021, #3022);
#3024 = LOCAL_TIME (17, 30, $, #3006);
#3025 = DATE_AND_TIME (#3021, #3024);
```

The new ‘meta-data’ above states that at half past eleven on the 16th June 1999, Andrew Crowley of the Steel Construction Institute, imported managed data from a STEP file named “testcase2.stp”, into the SCI copy of “QSE Space” (supplied by “Research

Engineers (Europe) Ltd”). Six hours later, he modified the data. All the instances of ‘managed_data_item’ have been placed in three sets of managed data represented by the instances #5003, #5004, and #5021. In addition, some of the managed data has been modified and placed into another set represented by the instance #5022. The unique identifier of each instance in these sets includes the ‘installation_id’ (= 101) and the ‘application_id’ (= 11) derived from the associated ‘managed_application_installation’ (instance #5002). This is the application that originally managed the data, rather than the application that modified the data.

It should be noted that the values of the hash (#) numbers in the physical file are not significant to the comparator logic. What is important is the association between entity instances established by the hash (#) numbers.

10.3.1 Dealing with OPTIONAL attributes

Where the EXPRESS schema defines the attribute to have an OPTIONAL data type, a value may or may not be present in the model, and the following rules apply:

1. Where OPTIONAL attributes are set to NULL in both models, the NULL value is maintained in δ , ϵ , or ζ .
2. Where OPTIONAL attributes have values in α but are set to NULL in β then:
 - δ - the values are set to NULL
 - ϵ - the values in α are maintained
 - ζ - the values are set to NULL
3. Where attributes are set to NULL in α but have values in β , the new values in β are added to δ , ϵ , or ζ .
4. Where OPTIONAL attributes have values in both α and β , the updating mechanism is dependent upon the data type that the attribute has been given in the EXPRESS schema (of LPM/6).

Sample entity instances from STEP Part 21 file

α	#4013 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45113', #5002, #13, (#5003, #5004), .T.); #13 = ELEMENT_CURVE_SIMPLE ('E1', \$, #100, 1, \$, #201, #133);
β	#4013 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45113', #5002, #13, (#5003, #5004, #5021, #5022), .T.); #13 = ELEMENT_CURVE_SIMPLE ('Element 1', 'left column', #100, 1, \$, #201, #133); /* element_name changed element_description added */
δ, ϵ, ζ	#4013 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45113', #5002, #13, (#5003, #5004, #5021, #5022), .T.); #13 = ELEMENT_CURVE_SIMPLE ('Element 1', 'left column', #100, 1, \$, #201, #133); /* element_name changed element_description added */

10.3.2 Dealing with simple and defined data types

Where the attribute has a simple data type (defined by the EXPRESS schema as either a NUMBER, REAL, INTEGER, STRING, BOOLEAN, LOGICAL, or BINARY) the attribute value of the instance in α is replaced by the attribute value of the equivalent instance in β .

Where the attribute has a defined (TYPE) data type the attribute value of the instance in α is replaced by the attribute value of the equivalent instance in β .

It should be noted that comparisons of REAL numbers are based on the precision of the translator program. Thus, when using three digit precision, $4.999 = 5.000 = 5.001$. If the attribute value in β is equal to that in α within the precision of the translator program, then the value is assumed not to have changed.

10.3.3 Dealing with ENTITY data types

Where the attribute has an ENTITY data type, a relationship is formed between this entity and the referred entity. The entity reference (the # number in the physical file) in α is replaced by the entity reference from β .

Sample entity instances from STEP Part 21 file

α	<pre>#4031 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45131', #5002, #31, (#5003, #5004), .T.); #31 = ELEMENT_NODE_CONNECTIVITY (2, 'End Node', #11, #18, \$, #34); #34 = RELEASE_LOGICAL ('FIXED END', \$, .F., .U., .F., .U., .F., .U.);</pre>
β	<pre>#4031 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45131', #5002, #31, (#5003, #5004, #5021, #5022), .T.); #31 = ELEMENT_NODE_CONNECTIVITY (2, 'End Node', #11, #18, #42, #35); /* rigid offset added spring release added */ #35 = RELEASE_SPRING_LINEAR ('Semi-rigid connection', \$, #36, \$, #36, \$, #37, \$); /* new */</pre>
δ, ϵ, ζ	<pre>#4031 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45131', #5002, #31, (#5003, #5004, #5021, #5022), .T.); #31 = ELEMENT_NODE_CONNECTIVITY (2, 'End Node', #11, #18, #42, #35); /* rigid offset added spring release added */ #35 = RELEASE_SPRING_LINEAR ('Semi-rigid connection', \$, #36, \$, #36, \$, #37, \$); /* new */</pre>

10.3.4 Dealing with ENUMERATION data types

Where the attribute has an ENUMERATION data type the attribute value (which is one of the enumeration elements) of the instance in α is replaced by the attribute value of the equivalent instance in β .

Sample entity instances from STEP Part 21 file

α	<pre>#1205 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45101', #5002, #1021, (#5003, #5004), .F.);</pre>
----------	---

```
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,METRE.));
```

```
β #1205 = MANAGED_DATA_ITEM('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45101', #5002,  
    #1021, (#5003, #5004, #5021, #5022), .F.);  
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.CENTI.,.METRE.));
```

```

δ, ε, ζ  #1205 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45101', #5002,
                                     #1021, (#5003, #5004, #5021, #5022), .F.);
                                     #1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.CENTI.,.METRE.));

```

10.3.5 Dealing with SELECT data types

Where the attribute has a SELECT data type, the attribute value (which is one of the select elements) of the instance in α is replaced by the attribute value of the equivalent instance in β following the rules for the underlying base type. It should be noted that the base type of a SELECT type might be different in the two models. In this case, the base type of α is replaced by the base type of β .

Example EXPRESS schema extract

```
TYPE colour_select = SELECT (colour, label, colour_spec);
END TYPE;
```

```
TYPE label = STRING;
END_TYPE;
```

```
TYPE colour = ENUMERATION OF (red, blue, green);
END TYPE;
```

```
TYPE length_measure = REAL;
END_TYPE;
```

```
ENTITY thing;
    thing_colour : colour_select;
    length : length_measure;
END ENTITY;
```

```
ENTITY colour_spec;  
    name : label;  
    description : label;  
END ENTITY;
```

Sample entity instances from STEP Part 21 file

```
/* Instances of data management item not shown for simplicity */
```

α #1 = THING (COLOUR(.RED.), 4.0);
 #2 = THING (LABEL('black'), 5.0);
 #3 = THING (#4, 6.0);
 #4 = COLOUR SPEC ('p34', 'pink');

```

#5 = THING (COLOUR(.BLUE.), 8.0);

β      #1 = THING (COLOUR(.BLUE.), 5.0);
        #2 = THING (COLOUR(.RED.), 5.0);
        #3 = THING (#4, 6.0);
        #4 = COLOUR_SPEC ('p35', 'purple');
        #5 = THING (#6, 8.0);
        #6 = COLOUR_SPEC ('p40', 'yellow');

δ, ε, ζ. #1 = THING (COLOUR(.BLUE.), 5.0);
          #2 = THING (COLOUR(.RED.), 5.0);
          #3 = THING (#5, 6.0);
          #4 = COLOUR_SPEC ('p35', 'purple');
          #5 = THING (#6, 8.0);
          #6 = COLOUR_SPEC ('p40', 'yellow');

```

10.3.6 Dealing with aggregation data types

All aggregation data types have other underlying data types, which may be any of the data types described above: *Simple*, *Named*, or *Aggregation*. The key to understanding how aggregation data types work - and the problems that arise when they are used - is shown as Table D.2 *Understanding the 'Nesting' of EXPRESS Data types*. This diagram is based on the EXPRESS notation and the grammar rules defined in Appendix A.2 of the EXPRESS Language Reference Manual^[16], and on the EXPRESS declaration of an entity as the follows:

```

ENTITY entity_name;
    attribute_name : [OPTIONAL] base_type;
END_ENTITY;

```

It can be seen in Table D.2 that the attribute's base type may be an aggregation, and there is significant looping available to the modeller, which allows the reuse of data types within nested data structures. It is this looping that makes EXPRESS a very powerful modelling language - far more powerful than one would have thought possible if nesting were not taken into consideration.

There are four types of aggregation data type: ARRAY, LIST, BAG, and SET. Each one of the aggregation data types behaves in a different way, which makes it suitable for a particular purpose. In addition to the rules described below for dealing with each of the aggregation data types, the rules for the aggregation's underlying data type (as described above) should also be followed.

10.3.7 Dealing with ARRAYS

An ARRAY is an indexed, fixed size, ordered collection. Because the elements of the ARRAY are indexed, individual elements of the ARRAY must be replaced in the correct order as defined by the index number. For example, the third element in *alpha_model* must be replaced by the third element in *beta_model*.

Further complications are added by the fact that an ARRAY can be declared to have OPTIONAL elements, UNIQUE elements or OPTIONAL UNIQUE elements. The rules

for the comparator operation follow a similar pattern to those of the base type. In the other words, all the values of all the elements in the `ARRAY` of the entity instance in `alpha_model` are replaced by the values of the equivalent elements (i.e. the element with the same index) in the `ARRAY` of the equivalent instance in `beta_model` following the rules described above for the base type.

Example EXPRESS schema extract

```
TYPE text = STRING;
END_TYPE;
```

```
TYPE colour = ENUMERATION OF (red, blue, green);
END_TYPE;
```

```
TYPE length_measure = REAL;
END_TYPE;
```

```
ENTITY thing;
  thing_colours : ARRAY [1:3] OF OPTIONAL colour;
  dimensions : ARRAY [1:3] OF UNIQUE length_measure;
  thing_names : ARRAY [1:2] OF OPTIONAL UNIQUE text;
END_ENTITY;
```

Sample entity instances from STEP Part 21 file

/ Instances of data_management_item not shown for simplicity */*

```
 $\alpha$     #1 = THING ((.RED., $, .GREEN.), (4.0, 5.0, 6.0), ('thing_abc', 'thing_1'));

 $\beta$     #1 = THING (($, .GREEN., $), (8.0, 9.0, 12.0), ('thing_1', 'thing_2'));

 $\delta$     #1 = THING (($, .GREEN., $), (8.0, 9.0, 12.0), ('thing_1', 'thing_2'));

 $\epsilon$     #1 = THING ((.RED., .GREEN., .GREEN.), (8.0, 9.0, 12.0), ('thing_1', 'thing_2'));

 $\zeta$     #1 = THING (($, .GREEN., $), (8.0, 9.0, 12.0), ('thing_1', 'thing_2'));
```

10.3.8 Dealing with LISTS

A LIST is an ordered collection of elements. Further complications are added by the fact that a LIST can be declared to have UNIQUE elements and it can have an undefined upper bound. In most cases where the attribute has a LIST data type, the values of the elements in the LIST of the instance in `alpha_model` are replaced by the values of the elements in the LIST of the equivalent instance in `beta_model`. Because the elements of a LIST are ordered, individual elements must be replaced in the correct sequence; e.g. the third element in `alpha_model` must be replaced by the third element in `beta_model`. If the UNIQUE keyword has been declared, then the LIST must be checked for uniqueness. If the UNIQUE constraint proves FALSE, then an error should be reported.

In the case where the LIST is declared to have an undefined upper bound, the following rules apply:

δ - the values of all the elements in the LIST of the instance in β are appended to the elements in the LIST of the equivalent instance in α . If the UNIQUE keyword has been declared, then only those elements with unique values are added to the LIST.

ϵ - as δ .

ζ - the values of all the elements in the LIST of the instance in α are replaced by the values of the elements in the LIST of the equivalent instance in β .

Example EXPRESS schema extract

```
TYPE colour = ENUMERATION OF (red, blue, green, cyan, magenta, yellow);
END_TYPE;
```

```
TYPE length_measure = REAL;
END_TYPE;
```

```
ENTITY thing;
  thing_colours : LIST [1:?] OF UNIQUE colour;
  dimensions : LIST [1:8] OF length_measure;
END_ENTITY;
```

Sample entity instances from STEP Part 21 file

/* Instances of data_management_item not shown for simplicity */

```
 $\alpha$     #1 = THING ( (.RED., .GREEN., .BLUE.), (4.0, 5.0, 6.2, 7.6, 9.2) );

 $\beta$     #1 = THING ( (.BLUE., .YELLOW.), (8.0, 10.0, 12.0) );

 $\delta$   #1 = THING ( (.RED., .GREEN., .BLUE., .YELLOW.), (8.0, 10.0, 12.0) );

 $\epsilon$   #1 = THING ( (.RED., .GREEN., .BLUE., .YELLOW.), (8.0, 10.0, 12.0) );

 $\zeta$     #1 = THING ( (.BLUE., .GREEN., .RED.), (8.0, 10.0, 12.0) );
```

10.3.9 Dealing with BAGs

A BAG is an unordered collection of elements. Because the elements of the BAG are unordered, they cannot be accessed by their sequence. Normally one would simply replace all the elements of the BAG in *alpha_model* with all the elements of the BAG in the *delta_model*. But if the BAG has been declared to have an undefined upper bound, it is assumed that all elements in *beta_model* are additional to the elements in *alpha_model*.

Example EXPRESS schema extract

```
TYPE colour = ENUMERATION OF (red, blue, green);
END_TYPE;
```

```
TYPE length_measure = REAL;
END_TYPE;
```

```

ENTITY thing;
    thing_colours : BAG [1:?] OF colour;
    dimensions : BAG [1:8] OF length_measue;
END_ENTITY;

```

Sample entity instances from STEP Part 21 file

/ Instances of data_management_item not shown for simplicity */*

```

α    #1 = THING ( (.RED., .GREEN., .BLUE.), (4.0, 5.0, 6.2, 7.6, 9.2) );

β    #1 = THING ( (.CYAN., .YELLOW., .RED.), (8.0, 10.0, 12.0) );

δ    #1 = THING ( (.RED., .GREEN., .BLUE., .CYAN., .YELLOW., .RED.), (8.0, 10.0, 12.0) );

ε    #1 = THING ( (.RED., .GREEN., .BLUE., .CYAN., .YELLOW., .RED.), (8.0, 10.0, 12.0) );

ζ    #1 = THING ( (.CYAN., .YELLOW., .RED.), (8.0, 10.0, 12.0) );

```

10.3.10 Dealing with SETs

A SET is an unordered collection of unique elements. Because the elements of the SET are unordered, they cannot be accessed by their sequence. Normally one would simply replace all the elements of the SET in alpha_model with all the elements of the SET in the delta_model. But if the SET has been declared to have an undefined upper bound, it is assumed that all UNIQUE elements in beta_model are additional to the elements in alpha_model.

Example EXPRESS schema extract

```

TYPE colour = ENUMERATION OF (red, blue, green, cyan, magenta, yellow);
END_TYPE;

```

```

TYPE length_measure = REAL;
END_TYPE;

```

```

ENTITY thing;
    thing_colours : SET [1:?] OF colour;
    dimensions : SET [1:8] OF length_measure;
END_ENTITY;

```

Sample entity instances from STEP Part 21 file

/ Instances of data_management_item not shown for simplicity */*

```

α    #1 = THING ( (.RED., .GREEN., .BLUE.), (4.0, 5.0, 6.2, 7.6, 9.2) );

β    #1 = THING ( (.CYAN., .YELLOW., .RED.), (8.0, 10.0, 12.0) );

```

δ #1 = THING ((.RED., .GREEN., .BLUE., .CYAN., .YELLOW.), (8.0, 10.0, 12.0));
 ϵ #1 = THING ((.RED., .GREEN., .BLUE., .CYAN., .YELLOW.), (8.0, 10.0, 12.0));
 ζ #1 = THING ((.CYAN., .YELLOW., .RED.), (8.0, 10.0, 12.0));

10.3.11 Dealing with multi-dimensional aggregation types

Aggregation data types in EXPRESS are one-dimensional. When two-dimensional constructs are required (e.g. a matrix) an aggregation data type is used as the data type of an aggregation data type. An example of such an EXPRESS construct - taken from LPM/6 - is shown below.

```

ENTITY element_point_stationary_mass
SUBTYPE OF(element_point);
  masses : ARRAY [1:3] OF REAL;
  moments_of_inertia : ARRAY [1:3] OF ARRAY [1:3] OF REAL;
END_ENTITY;
  
```

An 'element_point_stationary_mass' is a type of point element used to represent a nodal response matrix of type stationary mass. The stationary mass matrix is populated as follows:

$$\begin{bmatrix}
 m_x & & & & & \\
 & m_y & & & & \\
 & & m_z & & & \\
 & & & i_{xx} & i_{xy} & i_{xz} \\
 & & & i_{xy} & i_{yy} & i_{yz} \\
 & & & i_{xz} & i_{yz} & i_{zz}
 \end{bmatrix}$$

The 'masses' provide the numerical values of the point masses in the different directions. This is populated as follows:

- Array item 1 is the mass acting in the x direction, denoted m_x in the matrix above.
- Array item 2 is the mass acting in the y direction, denoted m_y in the matrix above.
- Array item 3 is the mass acting in the z direction, denoted m_z in the matrix above.

The 'moments_of_inertia' provide the numerical values of the moments of inertia of a point (denoted i_{xx} to i_{zz} in the matrix above) in the different directions (the values of the inertias are set to 0.0 if unwanted).

The problem arises because data types cannot be given a unique identifier. Consequently, one can never be certain which element of the aggregation has been modified or even if the aggregation is being appended.

The solution lies in the way in which the EXPRESS language is used to define the schema, and the rules for comparator operations. Consider the construct:

```
ENTITY entity_name;
    attribute_name : aggregation_type_1 [m:n] OF aggregation_type_2 [p:q] OF base_type;
END_ENTITY;
```

Each element of 'aggregation_type_1' is itself an 'aggregation_type_2'. Assuming that the base_type is not an aggregation_type then attribute has a two dimensional aggregation data type. For multi-dimensional aggregation data types, the rules for comparator operations (summarized in Table 10.1) must be followed in sequence with reference to Table D.2. The rules detailed here remain valid even when the dimensionality of the aggregation is greater than two.

10.3.12 Dealing with DERIVE, INVERSE attributes

It should be noted that the rules for incremental data import apply only to the explicit attributes of the entity, and **not** to those attributes declared after the DERIVE, or INVERSE clause of the ENTITY declaration. These (implicit) attributes must be recalculated every time a model is updated to ensure referential integrity. It is possible that the comparator operations will render the model invalid. Therefore, the model must be validated after it has been updated to ensure that the UNIQUE and WHERE rules have not been violated. A PMR or an IDI Translator program should contain functions to perform these procedures.

10.3.13 Dealing with 'complex entity instances'

When an entity is declared in the EXPRESS schema to be a SUBTYPE of another entity, a 'complex entity instance' is created in the exchange structure, which involves attributes from more than one entity-type declaration. Where the entity types are the same in both beta_model and alpha_model, the rules for incremental data import described above should be followed for each attribute of each entity giving rise to the complex entity instance. If different 'branches' of the SUPERTYPE-SUBTYPE 'tree' have been populated, it is still assumed that the matching unique identifiers in alpha_model and beta_model are referring to the same real world thing. There are several possibilities as shown in the examples below. The fundamental principle is that 'Update & Maintain' operations should aim to maintain as much information as possible, while 'Update & Replace' should aim to eliminate duplicated information.

Example EXPRESS schema extract

```
ENTITY aa SUPERTYPE OF (bb ANDOR cc);
    attrib_a: STRING;
END_ENTITY;
```

```
ENTITY bb SUBTYPE OF (aa);
    attrib_b: INTEGER;
END_ENTITY;
```

```
ENTITY cc SUBTYPE OF (aa);
    attrib_c: REAL;
END_ENTITY;
```

Sample entity instances from STEP Part 21 file

a) The same entity type in both models

α #1 = BB ('sample string', 15);

β #11 = BB ('another string', 66);

δ, ϵ, ζ #11 = BB ('another string', 66);

b) A more specialized instance in β

α #1 = AA ('sample string');

β #11 = BB ('another string', 66);

δ, ϵ, ζ #11 = BB ('another string', 66);

c) A less specialized instance in β

α #1 = BB ('sample string', 99);

β #11 = AA ('another string');

δ #11 = BB ('another string');

ϵ #11 = BB ('another string', 99);

ζ #11 = BB ('another string');

d) An instance of different branch of SUBTYPE-SUPERTYPE tree in β

α #1 = BB ('sample string', 55);

β #11 = CC ('another string', 3.0);

δ #11 = CC ('another string', 3.0);

ϵ #11 = (AA('another string') BB(55) CC(3.0));

ζ #11 = CC ('another string', 3.0);

Sample entity instances from STEP Part 21 file for an LPM/6 construct

α #4201 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45201', #5002, #201, (#5003, #5004), .T.);

/* Other instances of managed_data_item not shown for simplicity */

/* Section profile passed by attribute value */

#201 = SECTION_PROFILE_I_TYPE (1, 'W14x16x500', 'Wide flange beam to ASTM', 'Plastic', 10, .F., #202, #203, #204, #205, #206, \$, \$, \$);

```
#202 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(498.0), #1021);
#203 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(432.0), #1021);
#204 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(55.6), #1021);
#205 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(88.9), #1021);
#206 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(280.0), #1021);
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
```

β

```
#4201 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45201', #5002,
#201, (#5003, #5004, #5021, #5022), .T.);
/* Other instances of managed_data_item not shown for simplicity */
/* Section profile passed by reference */
#301 = ITEM_REFERENCE_STANDARD ('W14x500', #302);
#302 = ITEM_REF_SOURCE_STANDARD ('ASTM', 'ASTM_A6M', 1994, $);
#304 = ITEM_REFERENCE_ASSIGNED (#301, #303);
#303 = SECTION_PROFILE (1, 'W', 'Wide flange beam to ASTM', 'Plastic', 10, .F.);
```

δ

```
#4201 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45201', #5002,
#201, (#5003, #5004, #5021, #5022), .T.);
/* Other instances of managed_data_item not shown for simplicity */
/* Section profile passed by reference */
#301 = ITEM_REFERENCE_STANDARD ('W14x500', #302);
#302 = ITEM_REF_SOURCE_STANDARD ('ASTM', 'ASTM_A6M', 1994, $);
#304 = ITEM_REFERENCE_ASSIGNED (#301, #303);
#303 = SECTION_PROFILE (1, 'W', 'Wide flange beam to ASTM', 'Plastic', 10, .F.);
```

ε

```
#4201 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF45201', #5002,
#201, (#5003, #5004, #5021, #5022), .T.);
/* Other instances of managed_data_item not shown for simplicity */
/* Section profile passed by reference and attribute value */
#301 = ITEM_REFERENCE_STANDARD ('W14x500', #302);
#302 = ITEM_REF_SOURCE_STANDARD ('ASTM', 'ASTM_A6M', 1994, $);
#304 = ITEM_REFERENCE_ASSIGNED (#301, #303);
#303 = SECTION_PROFILE_I_TYPE (1, 'W', 'Wide flange beam to ASTM', 'Plastic', 10, .F.,
#202, #203, #204, #205, #206, $, $, $);
#202 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(498.0), #1021);
#203 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(432.0), #1021);
#204 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(55.6), #1021);
#205 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(88.9), #1021);
#206 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
(POSITIVE_LENGTH_MEASURE(280.0), #1021);
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
```

```

ζ #4201 = MANAGED_DATA_ITEM (201, #5002, #201, (#5003, #5004, #5021, #5022), .T.);
/* Other instances of managed_data_item not shown for simplicity */
/* Section profile passed by reference */
#301 = ITEM_REFERENCE_STANDARD ('W14x500', #302);
#302 = ITEM_REF_SOURCE_STANDARD ('ASTM', 'ASTM_A6M', 1994, $);
#304 = ITEM_REFERENCE_ASSIGNED (#301, #303);
#303 = SECTION_PROFILE (1, 'W', 'Wide flange beam to ASTM', 'Plastic', 10, .F.);

```

10.3.14 Dealing with the Part 21 'Scope Structure'

Where information sets are submitted to the comparator tool via a STEP Part 21 file, it is possible that the file will contain some '&SCOPE' declarations. The scope structure is part of the syntax of the encoding of STEP Part 21 and is explained in Section F.3 of APPENDIX F. In simple terms, it defines existence relationships between entity instances where an entity instance cannot exist without the presence of another entity instance that references it. (This goes beyond the existence dependencies that have been defined in the EXPRESS schema.) It is possible that the data management tool of a DMC translator has employed the 'scope structure' when assigning the meta-data to the data. In this case, a unique identifier contained in an instance of `managed_data_item` may apply to several instances in the physical file; the fundamental principal is that the collection of instances, taken together, represents a single real world concept.

Sample entity instances from STEP Part 21 file for an LPM/6 construct

```

#4201 = MANAGED_DATA_ITEM ('1A81FD96-D7A8-47d3-8DF7-BEEA6FF41001', #5002,
    #101, (#5003, #5004), .T.);
/* This instance provides the meta-data for an instance of design_part, which requires all of
the instances between &SCOPE and ENDScope */

#101 = &SCOPE
    #601 = PART_PRISMATIC_SIMPLE (1, 'PP1', 'columns main component', $, .ROLLED.,
        'piece mark 1', #201, #701, $, $);
    #701 = LENGTH_MEASURE_WITH_UNIT (10250.0, #1021);
    #1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI., .METRE.));
    #201 = SECTION_PROFILE_I_TYPE (1, 'W14x16x500', 'Wide flange beam to ASTM',
        'Plastic', 10, .T., #202, #203, #204, #205, #206, $, $, $);
    #202 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
        (POSITIVE_LENGTH_MEASURE(498.0), #1021);
    #203 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
        (POSITIVE_LENGTH_MEASURE(432.0), #1021);
    #204 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
        (POSITIVE_LENGTH_MEASURE(55.6), #1021);
    #205 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
        (POSITIVE_LENGTH_MEASURE(88.9), #1021);
    #206 = POSITIVE_LENGTH_MEASURE_WITH_UNIT
        (POSITIVE_LENGTH_MEASURE(280.0), #1021);
ENDSCOPE DESIGN_PART ('left column main component', #601, (#401), $);

```

For comparator operations, the collection of entity instances contained by the scope structure is treated as a single instance. The updating rules follow those for a complex entity instance; i.e. each dependent entity instance is regarded as a component of an ANDOR SUPERTYPE-SUBTYPE construct. If any attribute of any dependent entity instance is modified, then the whole scope structure is considered to be modified and should be updated accordingly.

11 REFERENCES

1. BJORK, B.-C.
Requirements and Information Structures for Building Product Data Models
Technical Research Centre of Finland, VTT Publications, Espoo, 1995
2. BOMAN, M., BUBENKO, J. AND JOHANNESSON, P.
Conceptual Modelling (2nd edition)
Dept. of Computer and Systems sciences, Stockholm University, Stockholm, 1991
3. BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., & MALER, E.
Extensible Markup Language (XML) 1.0 (Second Edition)
W3C Recommendation 6 October 2000
<http://www.w3.org/TR/REC-xml>
4. BRODIE, M., MYOPOULOS, J. AND SCHMIDT, J. (EDS)
On Conceptual Modelling - Perspectives from Artificial Intelligence, Databases and Programming Languages
Springer-Verlag, New York, 1985
5. CROWLEY, A.J.
The Development and Implementation of a Product Model for Constructional Steelwork
PhD Thesis, University of Leeds, UK, 1998
6. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 1 - Overview
SCI Publication P265, The Steel Construction Institute, UK, 2003
7. CROWLEY, A.J., WARD, M.A. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 3 - The Information Requirements
SCI Publication P267, The Steel Construction Institute, UK, (in preparation)
8. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 4 - The Logical Product Model (LPM/6)
SCI Publication P268, The Steel Construction Institute, UK, 2003
9. CROWLEY, A.J. SMITH, A.M. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 5 - Conformance Requirements
SCI Publication P269, The Steel Construction Institute, UK, 2003
10. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 6 - Worked Examples
SCI Publication P270, The Steel Construction Institute, UK, (in preparation)
11. DACOM
Information Modelling Manual IDEF1X
D.Appleton Company Ltd, 1985
12. GREINER, R. & SALZGEBER, G.
A Structured Approach to Design, Analysis and Code Check of Constructional Steelwork:
Engineering Part, Version 1 (Deliverable of the EUREKA 130 CIMsteel Overall Design
and Analysis Working Group)
Technical University of Graz, Austria, 1996
13. HAMBURG, S.E. & HOLLAND, M.V.
Leaping ahead with EDI, pp. 42-48 in *Modern Steel Construction*, Vol. 39, No. 1

American Institute of Steel Construction, Inc., Chicago, USA, February 1999

14. ISO/DTR 9007: 1985
Concepts and terminology for the conceptual schema and information base
ISO / TC97, 1985
15. ISO 10303-1: 1994
Industrial automation systems - Product data representation and exchange
Part 1: Overview and Fundamental Principles
ISO/IEC, Geneva, Switzerland, 1994
16. ISO 10303-11: 1994
Industrial automation systems - Product data representation and exchange
Part 11: The EXPRESS Language Reference Manual
ISO/IEC, Geneva, Switzerland, 1994
17. ISO 10303-21: 2002
Industrial automation systems - Product data representation and exchange
Part 21: Clear text encoding of the exchange structure
ISO/IEC, Geneva, Switzerland, 2002
(Incorporates Technical Corrigendum 1 published 1996-08-15)
18. ISO 10303-22: 1998
Industrial automation systems - Product data representation and exchange
Part 22: Standard Data Access Interface
ISO/IEC, Geneva, Switzerland, 1998
19. ISO 10303-23: 2000
Industrial automation systems - Product data representation and exchange
Part 23: C++ Programming Language Binding to the Standard Data Access Interface
ISO/IEC, Geneva, Switzerland, 2000
20. ISO 10303-24: 2001
Industrial automation systems - Product data representation and exchange
Part 24: C Language Late Binding to the Standard Data Access Interface
ISO/IEC, Geneva, Switzerland, 2001
21. ISO/CD 10303-26: 1996(E)
Industrial automation systems - Product data representation and exchange
Part 26: Interface definition language binding to the standard data access interface
ISO TC184/SC4/WG11 N019, March 1997
<http://www.nist.gov/sc4/step/parts/part026/current/part26.pdf>
22. ISO/TS 10303-27: 2000
Industrial automation systems and integration - Product data representation and exchange -
Part 27: Implementation methods: Java™ programming language binding to the standard
data access interface with Internet/Intranet extensions
ISO/IEC, Geneva, Switzerland, 2000
23. ISO/TS 10303-28: 2002(E)
Industrial automation systems - Product data representation and exchange
Part 28: Implementation methods: XML representations of EXPRESS schemas and data
ISO/IEC, Geneva, Switzerland, 2002
24. ISO 10303-31: 1994
Industrial automation systems - Product data representation and exchange
Part 31: Conformance Testing Methodology & Framework: General Concepts
ISO/IEC, Geneva, Switzerland, 1994
25. ISO 10303-41: 2000
Industrial automation systems - Product data representation and exchange
Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000

26. ISO 10303-42: 2000
Industrial automation systems - Product data representation and exchange
Part 42: Integrated Generic Resources: Geometric & Topological Representation
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
27. ISO 10303-43: 2000
Industrial automation systems - Product data representation and exchange
Part 43: Integrated Generic Resources: Representation Structures
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
28. ISO 10303-44: 2000
Industrial automation systems - Product data representation and exchange
Part 44: Integrated Generic Resources: Product Structure Configuration
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
29. ISO 10303-45: 1998
Industrial automation systems - Product data representation and exchange
Part 45: Integrated Generic Resources: Materials
ISO/IEC, Geneva, Switzerland, 1998
30. ISO 10303-201: 1994
Industrial automation systems - Product data representation and exchange
Part 201: Application protocol: Explicit Draughting
ISO/IEC, Geneva, Switzerland, 1994
31. ISO 10303-203: 1994
Industrial automation systems - Product data representation and exchange
Part 203: Application protocol: Configuration Controlled Design
ISO/IEC, Geneva, Switzerland, 1994
32. ISO WD 10303-221 (N194): 1995
Industrial automation systems - Product data representation and exchange
Part 221: Application protocol: Process Plant Functional Data and its Schematic Representation
ISO/IEC, Geneva, Switzerland, 1995
33. ISO FDIS 10303-225 (N710): 1997
Industrial automation systems - Product data representation and exchange
Part 225: Application protocol: Building Elements using Explicit Shape Representation
ISO/IEC, Geneva, Switzerland, 1997
34. ISO WD 10303-227 (N367): 1995
Industrial automation systems - Product data representation and exchange
Part 227: Application protocol: Plant Spatial Configuration
ISO/IEC, Geneva, Switzerland, 1995
35. ISO/WD 10303-230 (N551): 1996
Industrial automation systems - Product data representation and exchange
Part 230: Application protocol: Building Structural Frame: Steelwork
ISO TC184/SC4/WG3 (T12), 1996
36. ISO/IEC 11578: 1996
Information technology - Open Systems Interconnection - Remote Procedure Call (RPC)
ISO/IEC, Geneva, Switzerland, 1996
37. KERR, R.
Knowledge-Based manufacturing management: applications of artificial intelligence to the effective management of manufacturing companies
Adison-Wesley, Singapore, 1991
38. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
IDEF0 (ICAM Definition Language 0)
Federal Information Processing Standards Publication 183

- Integration Definition for Function Modeling (IDEF0)
FIPS PUB183, NIST, 1993
39. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
IDEF1X (ICAM Definition Language 1 Extended)
Federal Information Processing Standards Publication 184
Integration Definition for Information Modeling (IDEF1X)
FIPS PUB184, NIST, 1993
40. MICROSOFT INC
COM and ActiveX Object Services Microsoft Visual Studio Online Documentation
Microsoft Developer Network (<http://msdn.microsoft.com/>)
41. SCHENCK, D. & WILSON, P.
Information Modeling the EXPRESS Way
Oxford University Press, Oxford, 1994
42. THE OPEN GROUP
DCE 1.1: Remote Procedure Call, Open Group Technical Standard, Document number
C706, August 1997. (<http://www.opengroup.org/publications/catalog/c706.htm>)
43. WIRTH, N.
What can we do about the unnecessary diversity of notation for syntactic definition?
Communications of the Association for Computing Machinery, Vol. 20, No 11, Nov. 1977

APPENDIX A IMPLEMENTATION TOOLS

The following Section on modelling tools is included to illustrate the link between the modelling technology and the implementation technology. It also gives some guidance to those software developers who may wish to model their own applications data structures using EXPRESS modelling tools.

The Section on implementation technology is included to illustrate how the currently available STEP toolkits have been used to achieve successful data exchange in CIMsteel demonstrations, and how commercial CIS compatible translators can be developed.

A.1 Modelling tools

EXPRESS editors

Almost any kind of text editor or word processor can be used to create and edit an EXPRESS source. A language sensitive editor will have built in knowledge of EXPRESS syntax, and might also incorporate layout and capitalisation management. These can be useful to those unfamiliar with EXPRESS syntax.

EXPRESS parsers

An EXPRESS parser is an important tool in a STEP toolkit. It should detect and report every EXPRESS syntax error and never tag correct EXPRESS syntax as being in error. Accurate and meaningful error reporting is essential. A parser is often the front end to an EXPRESS compiler.

EXPRESS visualizers

Many people find it convenient to understand and work with graphical rather than textual models. A visualizer should be able to import and export EXPRESS source to and from a graphical model. It should be possible to edit the graphical model both in terms of layout and content. The tool should automatically take care of page splitting and inter-page connectors. Graphical output should be in a form that can be incorporated into a text document. EXPRESS-G (see APPENDIX E) is the favoured graphical notation used by the those developing STEP Application Protocols and Integrated Resources, although IDEF1X (see APPENDIX C) is sometimes used.

A.2 Implementation tools

EXPRESS compilers

An EXPRESS compiler should produce code which can be used in a computer program or database. It should provide a variety of reports to help with documentation and debugging. A typical EXPRESS compiler converts the EXPRESS into a language form for a particular implementation. The output is usually plain text reports and high-level implementation code such as C++ or SQL. The availability of commercial EXPRESS compilers means that it is not necessary for a translator developer to deal with such issues.

Most commercial EXPRESS compilers will meet the following criteria:

- **Accuracy:** Full support for the International Standard version of EXPRESS, with full error reporting.
- **Code Generation:** Generation of high-level code that enables applications to read, write, and share data models that conform to the EXPRESS schema.
- **Platform Portability:** Generation of code for a variety of platforms and operating systems.
- **Validation:** Generation of code to support WHERE, INVERSE, and UNIQUE constraints.

A.3 STEP file parsers

Full support for the STEP file format^[17] is a basic requirement of all CIS translators (see E.2). A STEP toolkit will provide this functionality. It will either automatically generate the necessary code to read and write STEP files conforming to the compiled EXPRESS model, or this facility will be provided by a separate library which uses the output from the EXPRESS compiler for its configuration. The availability of commercial STEP file parsers and tools that read and write STEP files mean that it is not necessary for a translator developer to deal with such issues.

Data manipulation tools

STEP toolkits will provide libraries of functions for manipulating data conforming to an EXPRESS schema. It will either automatically generate the necessary code to create and access data conforming to a specific compiled EXPRESS model (Early Bindings^[19]), or this facility will be provided by a separate library which has generic data manipulation functions built in. These functions may be used with any EXPRESS schema that has to be compiled for that library (Late Bindings^[20]).

Typically, data is read from a STEP file into a ‘repository’. The ‘repository’ could be anything from relational database, to a simple linked-list. Without some kind of standard, the way that the data is manipulated within the repository will vary depending on which toolkit is used. This could clearly lead to problems if the translator developer decides to change toolkits at a future date.

To overcome this problem STEP is currently defining a standard for an API to STEP data repositories. This is called the STEP Data Access Interface (SDAI)^[18], and the initial specification includes language bindings to C, and C++^[19, 20]. (See APPENDIX G.)

A CIS translator does not have to support SDAI to be CIS compatible. However, it is recommended that vendors choose a toolkit that supports the latest specification. Using a STEP toolkit which supports the SDAI specification will allow easier migration to full SDAI compatibility when it becomes a full International Standard. It will also allow translators to change toolkits more easily as the market for STEP tools develops.

APPENDIX B IDEF0

The following is based on the Functional Modelling Manual (IDEF0) produced by SofTech Inc; Part II, Volume IV of the Integrated Computer-Aided Manufacturing Architecture documentation, (AFWAL-TR-81-4023, 1981).

An Application Activity Model (AAM) is used to represent the activities (or processes) that are involved in creating or using the information within the subject areas. The CIS/2 AAM is documented in the *Information Requirements*^[7]. Example pages of the AAM are shown in Figures B.3 and B.4. The AAM is represented using IDEF0.

IDEF0 is a diagramming method used to describe systems. It is a top-down, from general to specific method: from a single diagram representing an entire system to more detailed diagrams explaining subsections. The IDEF0 methodology leads to collections of successively more detailed diagrams and text pages explaining each diagram. It is also known as SADT.

The IDEF0 modelling methodology has a set of design rules:

- An Activity is an imperative (i.e. Do Something)
- An Activity may summarise other activities (i.e. Abstract)
- A flow represents captured information (e.g. Bill of Materials, Drawing)
- The flow is in the direction of the arrow head
- A Control governs when and how an activity occurs (e.g. Building regulations).
- An Input is converted into an output by an activity (i.e. Transformation)
- A Mechanism is used during an activity
- An Output is produced from an Input (or a control) during an activity

B.1 The IDEF0 notation

The IDEF0 notation is as follows:

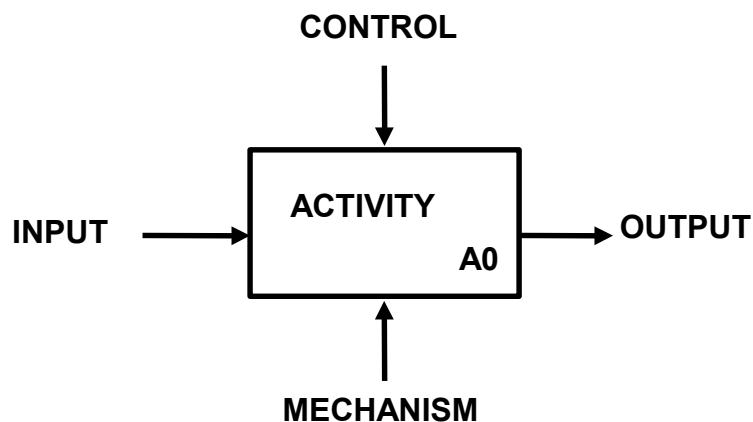


Figure B.1 *The Basics of IDEF0*

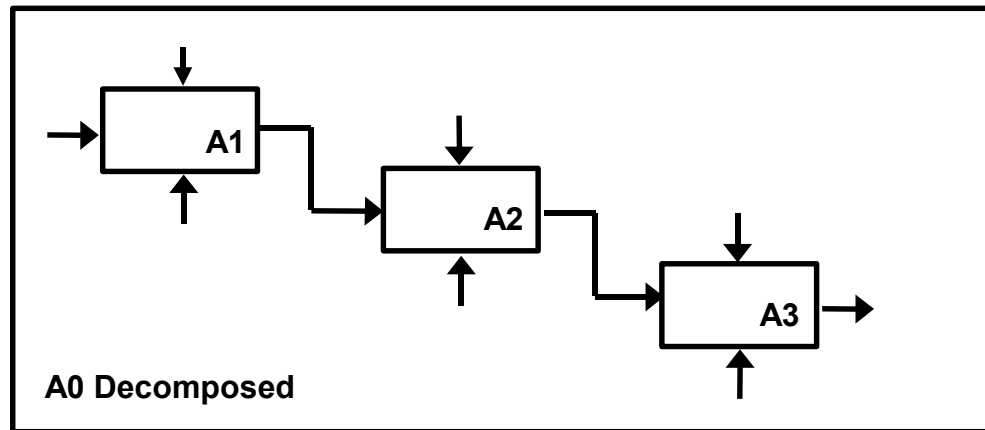


Figure B.2 *Decomposing a high level activity*

Within an IDEF0 model, information flow is shown as a number of ICOM arrows. These ICOM arrows may be Inputs, Controls, Outputs, or Mechanisms for the activities. The fundamental principle of the IDEF0 methodology is that an Input is converted into an Output by an Activity. A Control governs when and how an activity occurs, while a Mechanism is something that is used (but not consumed) during an activity. A component manufacturer could view this as a way of representing the system whereby raw materials (the Input) are processed (the Activity) to produce a product (the Output) using some manufacturing plant (the Mechanism) which is controlled by a production schedule (the Control).

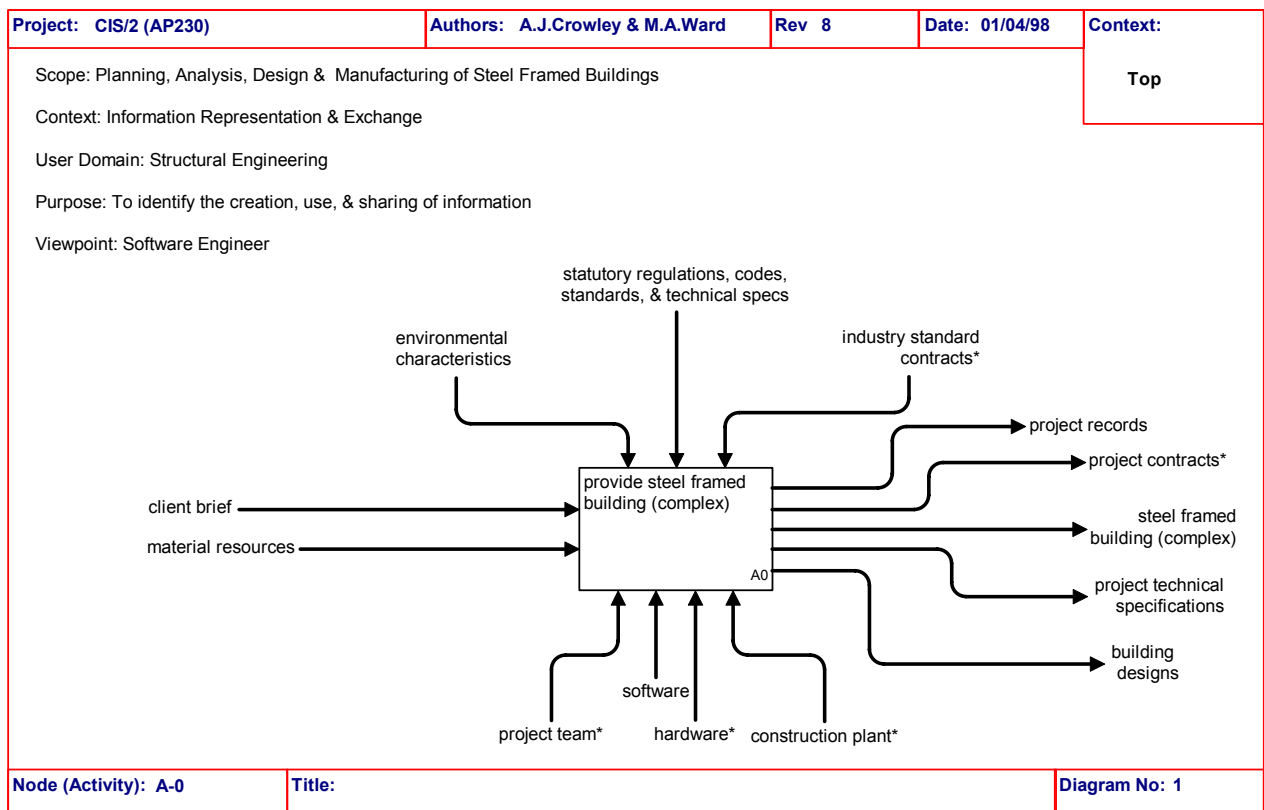


Figure B.3 *The top level diagram for the CIS/2 Activity Model*

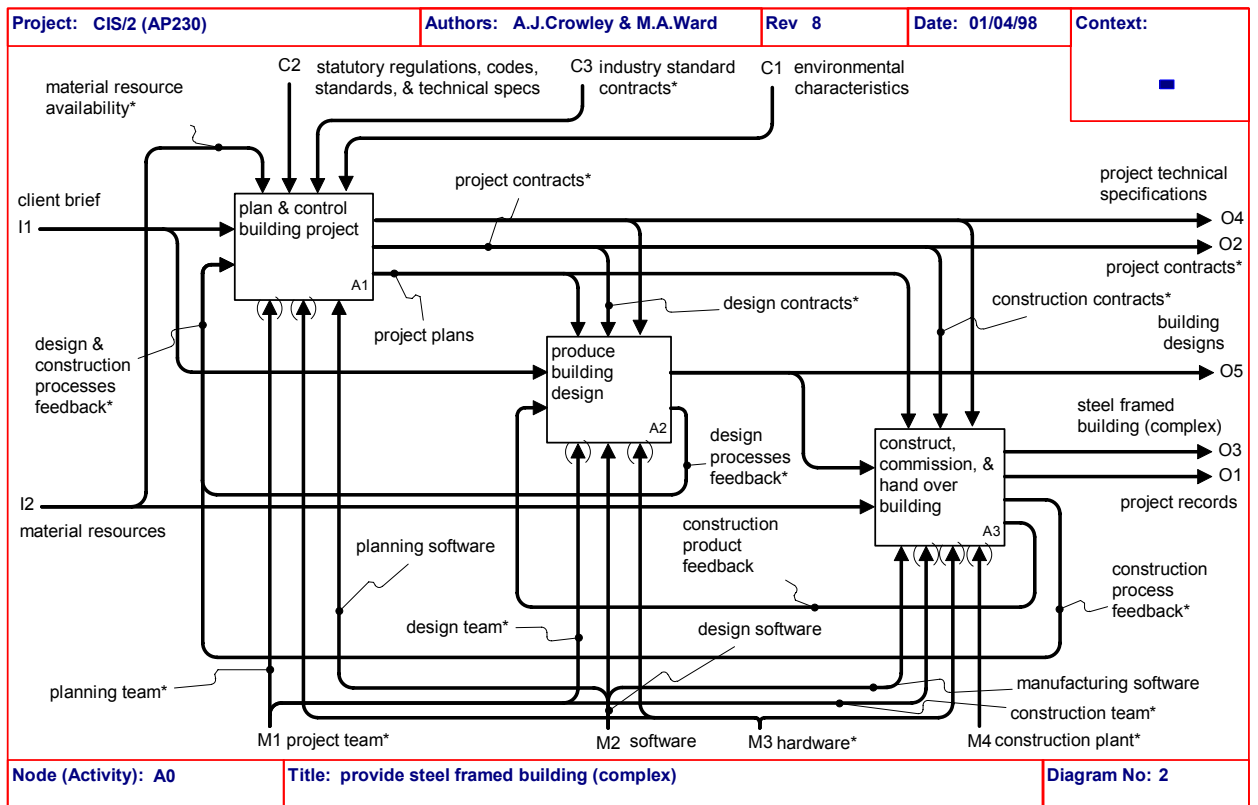


Figure B.4 *The first decomposition of the CIS/2 Activity Model*

APPENDIX C IDEF1X

According to the ‘Information Modelling Manual’^[11], IDEF1X is a graphical modelling notation, which is similar to most other Entity-Relationship notations. The IDEF1X methodology defines the logical structure of data in terms of entities, attributes of entities, and relationships between entities. A relationship is a verb-phrase that connects one entity with another entity, and specifies the cardinality between those entities. The main advantage of IDEF1X is that because it was initially intended for the design of relational database schema, it is a well-understood methodology with good supporting software tools.

In simple terms an IDEF1X model consists of glossaries plus model diagrams composed of:

- Boxes (Entities): Representing ‘things’ about which data is kept, e.g. people, places, events.
- Names within the boxes: Representing ‘characteristics or attributes’ of these things.
- Lines Linking Boxes: Representing ‘relationships’ between these things.

C.1 Entities

An entity represents a set of real or abstract things (people, objects, places, ideas, etc.) which have common attributes or characteristics. Each entity is represented by a box and assigned a unique name. In a complete model, each entity should appear only once. Entities are of two types, Independent, and Dependent as shown in Figure C.1. Independent (Parent) entities are drawn as square cornered boxes and **do not** depend on any other entity for their existence, or identification. Dependent (Child) entities are drawn as round cornered boxes and **do** depend on another entity for their existence, and identification.

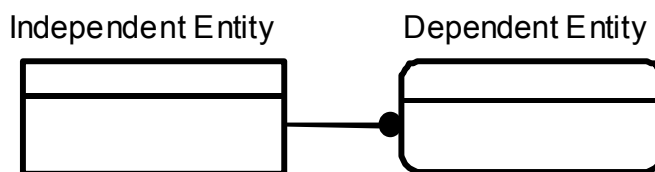


Figure C.1 *Independent and Dependent Entities in IDEF1X*

C.2 Attributes

An attribute represents a particular property or characteristic of an entity. Each entity has one (or more) of its attributes selected to be the Primary Key which must uniquely identify each entity instance. The Primary Key attributes are always inherited by a Dependent entity. The Primary Key attributes are written above a horizontal line that divides the box with other attributes listed below. Inherited Primary Key attributes become known as Foreign Keys. These may be used as either a total or a portion of the Primary Keys in a Dependent entity, or as non-key attributes within an Independent entity.

C.3 Relationships

A specific relationship is an association between two entities in which each instance of the first entity is associated with zero, one, or more instances of the second entity, and each instance of the second entity is associated with exactly one instance of the first entity.

If an instance of the child entity is identified by its association with the parent entity, then the relationship is referred to as an **Identifying Relationship**. In a specific relationship, the Primary Key of the parent entity becomes a **Foreign Key** attribute (FK) of the child entity. In an identifying relationship, the Foreign Key becomes the Primary Key, or part of the Primary Key, of the child entity. Thus, the child is a **Dependent** entity; i.e. it depends on its parent for its existence. An **Identifying Relationship** is represented by a solid line, as shown on the left-hand side of Figure C.2.

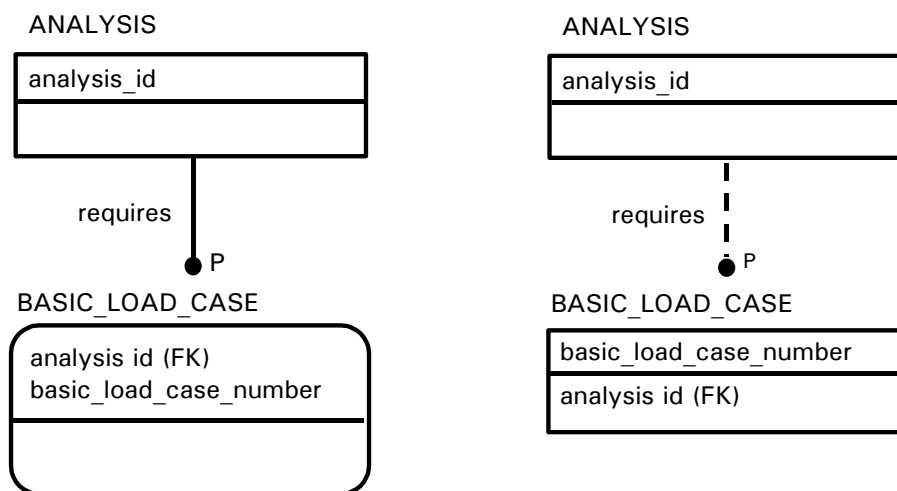


Figure C.2 *Identifying and Non-Identifying Relationships in IDEF1X*

If every instance of the child entity can be uniquely identified without knowing the identity of the parent entity, then the relationship is known as a **Non-identifying Relationship**, and is represented by a broken line, as shown on the right hand side of Figure C.2. Thus, the child does not depend on a parent for its existence.

C.4 Cardinality

The connection relationship may be further defined by specifying the cardinality of the relationship. Cardinality is the specification of how many child instances may exist for each parent instance. Within IDEF1X, the following relationship cardinalities can be expressed:

- — Each parent entity instance may have zero, one or more associated child entity instances
- P ● — Each parent entity instance must have at least one or more associated child entity instances
- Z ● — Each parent entity instance can have zero or at most one associated child entity instances

n — Each parent entity instance is associated with some exact number (n) of child entity instances

An intrinsic aspect of the IDEF1X modelling methodology is the imposition of the third normal form. This results in the resolution of non-specific (many-to-many) relationships. Thus, a fully normalized IDEF1X model will only contain relationships with cardinality defined in one direction.

C.5 Categorisation Relationships

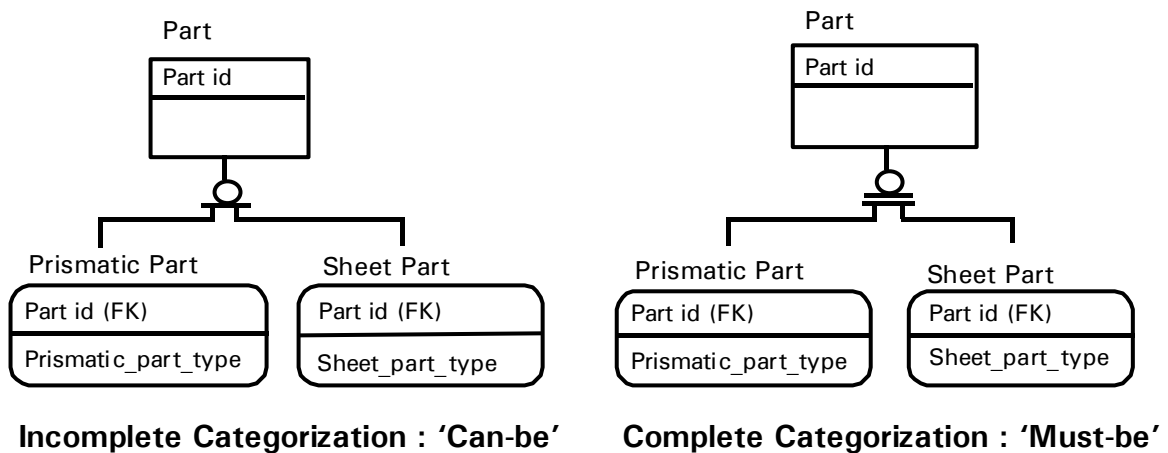


Figure C.3 *Incomplete & Complete Categorization Relationships in IDEF1X*

A categorisation relationship is used to show that a generic entity can be classified into one or more category entities. Both the generic and category entities have the same Primary Key attributes (they represent the same real world thing). Categorisation relationships are defined as being either complete or incomplete. A **complete** categorisation relationship is defined when all of the possible categories are known, and has a double underline to the circle, while an **incomplete** relationship (with an incomplete set of categories) has a single underline, as shown in Figure C.3.

APPENDIX D EXPRESS

EXPRESS is a language that provides an alphanumeric representation whose basic constructs are the *schema*, the *type*, the *entity* and the *rule*. EXPRESS has a lot in common with Object Oriented (OO) programming languages, but strictly speaking, it is not one. It does not have ‘methods’, but it does have *functions*, *rules* and *procedures*, and each of these can be associated with executable statements. Although EXPRESS is not an OO programming language, it is possible to use it as an OO data modelling language (i.e. a data definition language for creating models for implementation in an OODBMS).

According to Schenck & Wilson^[41], EXPRESS is an object-flavoured information model specification language that was initially developed in order to enable the writing of formal information models describing mechanical products. EXPRESS was created specifically to define the information that is consumed or produced throughout a product’s life cycle. EXPRESS itself does not provide for the definition of database formats, file formats, or transfer syntax; this is provided by parts of STEP which give mappings from EXPRESS schemata to implementation forms. For example, STEP Part 21^[17] defines the format of the exchange file. EXPRESS data models are used to generate the data structures held within the STEP physical file, using the mappings defined in the standard. However, the STEP physical file format is totally independent of EXPRESS.

D.1 Summary of the EXPRESS Language

In simple terms, the main features of the EXPRESS language are:

- i. **the containing structure**, called a *schema*; which can be seen as a boundary of the area of interest described. Different schemas can reference each other (using USE or REFERENCE clauses).
- ii. **the definitions of objects**, called *entities*; whose properties are represented as attributes and constraints. EXPRESS allows the definition of entities as SUBTYPES of other entities, where a SUBTYPE entity is a specialization of its SUPERTYPE. This establishes an inheritance relationship, in which the SUBTYPE inherits the characteristics (attributes and constraints). This leads to the inclusion of an entity in an inheritance graph.
- iii. **the attributes of the objects**; which can be seen as relationships between the entity and a *type*. The types can be either:
 - **simple data types** (BINARY, BOOLEAN, LOGICAL, STRING, NUMBER, INTEGER, REAL)
 - **named data type** (either an entity data type or a defined data type - i.e. declared by a type definition)
 - **aggregation data types** (LIST, SET, BAG and ARRAY with the cardinality defined)
 - **constructed data type** (SELECT, ENUMERATION).
 - **optional, derived, or inverse** (in addition to the above).

D.2 Defining entities in EXPRESS

An entity is defined as a set of attributes, each of which has a data type and a cardinality. The relationship between entities is implicit in the entity data type of an attribute. There is no formal modelling methodology associated with EXPRESS, and there are no requirements for normalisation. Thus, there is a degree of flexibility within the EXPRESS language that enables the same concept to be represented in a number of different ways.

In EXPRESS, there is no distinction between an entity representing a real object and an entity representing a relationship. EXPRESS does not have the concept of ‘parent’ or ‘child’ entities. Neither does it have the concept of specific or non-specific relationships. Existence dependency is defined by the cardinality of the relationship in the forward and inverse directions. EXPRESS does not support the notion of Keys.

D.2.1 SUPERTYPES and SUBTYPES

The SUBTYPE/SUPERTYPE construct is the mechanism for specifying classifications in an EXPRESS information model. Since the SUBTYPE is a more specific kind of its SUPERTYPE, every instance of a SUBTYPE is an instance of its SUPERTYPE(s). An entity is a SUBTYPE if, and only if, it contains a SUBTYPE clause. A SUBTYPE entity inherits the attributes and constraints of the more general entities occurring above it in the SUBTYPE/SUPERTYPE hierarchy. This establishes an inheritance relationship, in which the SUBTYPE inherits the characteristics (attributes and constraints). This leads to the inclusion of an entity in an inheritance graph. An example of a SUBTYPE/SUPERTYPE hierarchy is shown below.

Example entity definition in EXPRESS

```
ENTITY structural_frame_item
```

```
  SUPERTYPE OF (ONEOF (structural_frame_product, structural_frame_process,
    feature, joint_system, material, section_profile));
```

```
    item_number : INTEGER;
```

```
    item_name : label;
```

```
    item_description : OPTIONAL text;
```

```
END_ENTITY;
```

```
ENTITY structural_frame_product
```

```
  SUPERTYPE OF (ONE OF (assembly, coating, part, fastener, fastener_mechanism,
    weld_mechanism, chemical_mechanism) ANDOR
    structural_frame_product_with_material)
```

```
  SUBTYPE OF (structural_frame_item);
```

```
    life_cycle_stage : OPTIONAL label;
```

```
END_ENTITY;
```

```
ENTITY part
```

```
  SUPERTYPE OF (ONEOF(part_prismatic, part_sheet, part_complex) ANDOR part_derived)
```

```
  SUBTYPE OF (structural_frame_product);
```

```
    fabrication_method : fabrication_type;
```

```
    manufacturers_ref : OPTIONAL text;
```

```
END_ENTITY;
```

ENTITY material

SUPERTYPE OF (ONEOF(material_anisotropic, material_isotropic,
material_orthotropic) ANDOR material_constituent)

SUBTYPE OF (structural_frame_item);

END_ENTITY;

D.3 EXPRESS data types

In EXPRESS every attribute, local variable or formal parameter has an associated data type. Types can be classified as **simple**, **named**, **aggregation**, or **constructed** (enumeration or select) data types.

D.3.1 Simple data types

Simple data types are the basic types provided by the EXPRESS language. They include:

- NUMBER - (Supertype of Real and Integer)
- REAL - composed of a mantissa (including a decimal point) and an optional exponent e.g. 1.e6, 3.5E-5, 356.64
- INTEGER - composed entirely of digits
- STRING - composed of a sequence of characters enclosed by apostrophes
- BOOLEAN - TRUE or FALSE
- LOGICAL - TRUE, FALSE, or UNKNOWN
- BINARY - 0 or 1

D.3.2 Named data types

Named data types are the data types that the modeller declares. They can be either **entity** or **defined** data types.

Entity data types

Entity data types are object types declared by entity declarations. Any entity declared in a schema can be used as the data type of an attribute, local variable or formal parameter. Using an entity as an attribute's data type establishes a relationship between two entities. An example of an entity data type used to establish a relationship is shown below.

```
ENTITY POINT;
  x, y, z : REAL;
END_ENTITY;
```

```
ENTITY LINE;
  p1, p2 : POINT;
END_ENTITY;
```

The LINE entity has two attributes named 'p1' and 'p2'. The data type of each of these attributes is defined by the entity POINT.

Defined data types

A defined data type is a user extension to the set of **simple** data types and is created in a **TYPE** declaration. A defined data type can be used as any other data type by referencing the name given to it. An example of a defined data type declared by a **TYPE** assigned by the modeller is shown below.

```
TYPE volume = REAL;
END_TYPE;
```

```
ENTITY PART;
    bulk: volume;
END_ENTITY;
```

The attribute named 'bulk' is defined by the type assignment of 'volume' as a real number. However, the use of the defined data type 'volume' helps to clarify the meaning and context of that real number.

D.3.3 Aggregation data types

Aggregation types are used to represent ordered or unordered collections of elements of some basic type. An example of an aggregation type is shown below.

```
ENTITY STRUC_MEMBER
    mbr_connections: OPTIONAL SET [1:?] OF STRUC_CONNECTION;
END_ENTITY;
```

The example shows a SET aggregation type, but there are others - ARRAY, BAG, LIST. (Aggregation types are discussed in more detail later.)

D.3.4 Constructed data types

Constructed data types can be either **ENUMERATION** or **SELECT** data types.

ENUMERATION data type

An enumeration data type is an ordered set of values represented by names. The values of the enumeration-type are designated by **enumeration_ids** and are referred to as enumeration items. Each enumeration item belongs only to the data type that defines it, and must be unique within that type definition. An example enumeration type is shown below.

```
TYPE ELMNT_BASIC_LD_TYPE = ENUMERATION OF
    (CONC_1D_ELMNT_LD, DISTRIB_1D_ELMNT_LD, UNDEFINED);
END_TYPE;
```

SELECT data type

A select data type defines a named collection of other types. This collection is called the select list. An instance of a select data type is an instance of one of the types specified in the type list. This allows an attribute or variable to be one of several types. If the type of an attribute is a select type, then it is defined to be one of the types in the select list.

D.3.5 INVERSE attributes

Every attribute that has an entity as its data type establishes an entity relationship. An **inverse attribute** captures the relationship between the entity being referred to, and the named attribute of the referencing entity. Inverse attributes are declared following the INVERSE keyword.

D.4 Domain rules and where clauses

Domain rules specify constraints that restrict a defined data type. Domain rules follow the **WHERE** keyword. Each domain rule evaluates to a logical (TRUE, FALSE, UNKNOWN) value. A domain rule is asserted when the expression evaluates to TRUE; and is violated when the expression evaluates to FALSE. For example, a defined data type could be created which constrained the underlying values to be positive integers.

```
TYPE positive = INTEGER;
WHERE
    not_negative: SELF > 0;
END_TYPE;
```

Any attribute, local variable or formal parameter declared to be of type 'positive' is then constrained to allow only positive integer values.

D.5 Relationships and cardinality in EXPRESS

As relationships are an important aspect of modelling in EXPRESS, this Section discusses how they are made (in both IDEF1X and EXPRESS) and how they are constrained, and how important 'relationships' are in an apparently 'object-oriented' modelling language like EXPRESS. It must be emphasised that the modeller must consider all the possible relationships that are formed when one entity is declared to be the data type of another entity.

The first example is the 'one-to-many' relationship. Consider the EXPRESS specification shown below and illustrated in IDEF1X in Figure D.1.

```
ENTITY first_entity;
relating_attribute : second_entity;
first_attribute : STRING;
END_ENTITY;

ENTITY second_entity;
second_attribute : STRING;
END_ENTITY;
```

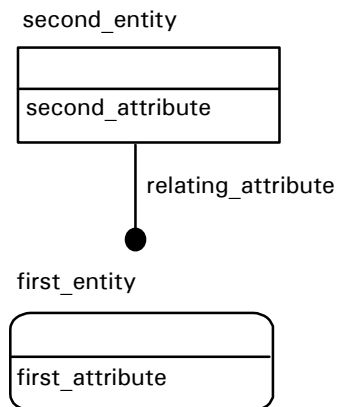


Figure D.1 Representation of a 'one-to-many' relationship

The above EXPRESS specification is interpreted to mean that entities 'first_entity' and 'second_entity' are related through 'relating_attribute'. The cardinality of 'first_entity' with respect to 'second_entity' in this case is [1:1] i.e. every instance of 'first_entity' requires one instance of 'second_entity'. (It should be noted that the specification of the upper and lower bounds of the aggregation is known as the 'bounds spec'.) The default inverse cardinality (as given by the 'bounds spec' [0:?]) is not usually made explicit. Thus, in the above case, an instance of 'second_entity' may be associated with zero, one or more instances of 'first_entity'. It would appear that the relationship has been drawn 'upside-down'. However, it must be remembered that relationships are drawn from parent to child in IDEF1X but from 'child to parent' in EXPRESS (as far as there exist parent and child entities in EXPRESS - see earlier discussion).

It should also be noted that a reference between entities in an EXPRESS information model is not a pointer (although in an information base it could be implemented as one). In particular, an inverse attribute is not a pointer; it is a cardinality constraint. Its presence in an entity, though, does have the side effect of enabling access to the other entity.

D.5.1 Cardinality and dependency

Certain cases of cardinality **imply** dependency; i.e. one entity cannot exist (or be instanced) without another. In order to derive the dependencies from the possible combinations of cardinality consider the more general case of the 'many-to-many' relationship.

This is shown in the EXPRESS specification below and illustrated in IDEF1X in Figure D.2:

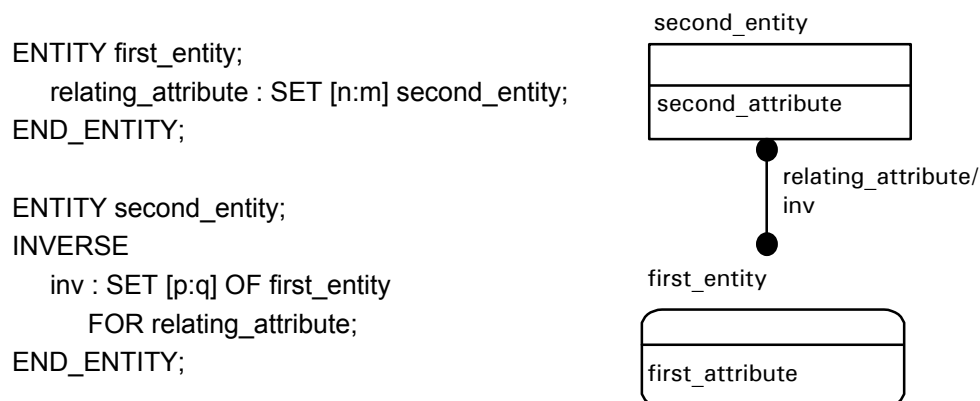


Figure D.2 Representation of a "many-to-many" relationship

The above specification is the more generalized version of the previous EXPRESS specification, and is interpreted to mean that the entities 'first_entity' and 'second_entity' are related through the attribute 'relating_attribute'. The cardinality of 'first_entity' with respect to 'second_entity' is [n:m] where n and m are lower and upper bounds of the cardinality of the relationship in the forward direction, while p and q are lower and upper bounds of the cardinality of the relationship in the inverse direction.

In theory, there are an infinite variety of conditions of cardinality. In practice we are only interested in four basic cases: i.e. where the lower bound takes the value 0 or 1, while the upper bounds can take the value 1 or ? (unbounded).

- [1:1] one
- [0:1] zero or one
- [1:?] one or many
- [0:?] zero, one or many

In order to understand the different cases of cardinality and how they should be interpreted, each of the four different cases of forward cardinality and each of the four cases of inverse cardinality may be written in ‘structured English’ as follows.

1. *Forward Cardinality [1:1] $n = m = 1$*

Every instance of ‘first_entity’ requires one (and only one) instance of ‘second_entity’, thus ‘first_entity’ is existence dependent on ‘second_entity’.

2. *Forward Cardinality [0:1] $n = 0, m = 1$*

An instance of ‘first_entity’ may be associated with one (and only one) instance of ‘second_entity’, but ‘first_entity’ is independent of ‘second_entity’.

3. *Forward Cardinality [1:?] $n = 1, m = ?$*

Every instance of ‘first_entity’ requires at least one instance of ‘second_entity’, thus ‘first_entity’ is existence dependent on ‘second_entity’, but there may be more than one instance of ‘second_entity’ associated with an instance of ‘first_entity’.

4. *Forward Cardinality [0:?] $n = 0, m = ?$*

An instance of ‘first_entity’ may be associated with one or more instances of ‘second_entity’, but ‘first_entity’ is independent of ‘second_entity’, and there may be more than one instance of ‘second_entity’ associated with an instance of ‘first_entity’.

5. *Inverse Cardinality [1:1] $p = q = 1$*

Every instance of ‘second_entity’ requires one (and only one) instance of ‘first_entity’, thus ‘second_entity’ is existence dependent on ‘first_entity’.

6. *Inverse Cardinality [0:1] $p = 0, q = 1$*

An instance of ‘second_entity’ may be associated with one (and only one) instance of ‘first_entity’, but ‘second_entity’ is independent of ‘first_entity’.

7. *Inverse Cardinality [1:?] $p = 1, q = ?$*

Every instance of ‘second_entity’ requires at least one instance of ‘first_entity’, thus ‘second_entity’ is existence dependent on ‘first_entity’, but there may be more than one instance of ‘first_entity’ associated with an instance of ‘second_entity’.

8. *Inverse Cardinality [0:?] $p = 0, q = ?$*

An instance of ‘second_entity’ may be associated with one or more instances of ‘first_entity’, but ‘second_entity’ is independent of ‘first_entity’, and there may be more than one instance of ‘first_entity’ associated an instance of ‘second_entity’.

Once these basic principles have been established, the cardinality matrix shown in Table D.1 can be made by putting these cardinality combinations together. Simplifying the constraint matrix, it can be seen that the following rules emerge:

- if $n=1$ there is forward dependency
- if $p=1$ there is inverse dependency
- if $n=p=1$ then there is have mutual dependency
- if $n=p=0$ then there is mutual independence; i.e. no dependency.

Table D.1 *Derived dependency matrix*

Inverse cardinality [p:q]	Forward cardinality [n:m]			
	[1:1]	[0:1]	[1:?]	[0:?]
[1:1]	Mutually dependent	inverse dependency	mutually dependent	inverse dependency
[0:1]	Forward dependency	none	forward dependency	none
[1:?]	Mutually dependent	inverse dependency	mutually dependent	inverse dependency
[0:?]	Forward dependency	none	forward dependency	None

D.5.2 Simple data types in relationships

As discussed above, relationships between entities are formed by an attribute of one entity referencing the other entity. It does this by using the second entity as the data type of the referencing attribute. The second entity participates in the relationship as either a simple or aggregation data type, as defined by the *EXPRESS Language Reference Manual*^[16]. The example on cardinality only shows a SET aggregation data type, but there are others.

Consider the EXPRESS specification shown below:

```
ENTITY first_entity;
    relating_attribute : second_entity;
    first_attribute : STRING;
END_ENTITY;
```

```
ENTITY second_entity;
    second_attribute : STRING;
END_ENTITY;
```

Here a 'second_entity' is used as a simple data type. A 'bounds spec' cannot be supplied though it is implied that the forward cardinality is [1:1]. (It should be noted that the default inverse cardinality is [0:?].)

D.5.3 Aggregation data types in relationships

Aggregation types are used to represent ordered or unordered collections of elements of some basic type. There are four types of aggregate data type defined in EXPRESS; i.e. ARRAY, BAG, LIST, and SET.

ARRAY data type

An ARRAY is a fixed size ordered collection. It is indexed by a sequence of integers. The data type definition specifies whether or not an instance may have duplicate elements. (See clause 8.2.1 of ISO 10303-11: 1994.^[16])

An array may contain duplicate entities unless the UNIQUE keyword is present. UNIQUE indicates that each element of that array is different from every other element in the same array instance. An array rather than a list should be used if indexing is important and a fixed number of elements required. The bounds spec provides the index of the first and last elements, thus the upper bound cannot be specified by a “?”.

LIST data type

A LIST is a sequence of elements that can be accessed according to their position. The number of elements in a list may vary, and can be constrained by data type definition. A list may contain duplicate entities unless the UNIQUE keyword is present. (See clause 8.2.2 of ISO 10303-11: 1994.)

A LIST data type has as its domain sequences of like elements. The optional upper and lower bounds [n:m], which are integer-valued expressions, define the minimum and maximum number of elements that can be held in an instance of a LIST data type.

BAG data type

A BAG is an unordered collection in which duplication is allowed. The number of elements in a BAG may vary, and can be constrained by the data type definition. (See clause 8.2.3 of ISO 10303-11: 1994.) A BAG data type has as its domain unordered collections of like elements. The optional lower and upper bounds, which are integer-valued expressions, define the minimum and maximum number of elements that can be held in a collection defined by BAG data type.

SET data type

A SET is an unordered collection in which duplicate elements are **not** allowed. The number of elements in a SET may vary, and can be constrained by the data type definition. (See clause 8.2.4 of ISO 10303-11: 1994.)

A SET data type has as its domain unordered collections of like elements. The SET data type is a specialization of the BAG data type. The optional lower and upper bounds, which are integer-valued expressions, define the minimum and maximum number of elements that can be held in a collection defined by a SET data type. A collection defined by a SET data type shall not contain two or more elements that are instance equal.

Example

a_set_of_points : SET OF point;

The attribute named `a_set_of_points` can contain zero or more points. Each point instance is required to be different from every other point in the set. If no limits are stated then the limits `[0:?]` can be assumed. If the value is required to have no more than 15 points, the specification can provide an upper bound, as in:

`a_set_of_points : SET [0:15] OF point;`

Multi-dimensional aggregate data types

The bounds spec `[n:m]` is used to determine the size of the collection, rather than the dimensionality. EXPRESS aggregations are one-dimensional. Objects that are usually considered to have multiple dimensions (such as mathematical matrices) can be represented by an aggregation data type whose base type is another aggregation data type. Aggregation data types may thus be nested to an arbitrary depth, allowing any number of dimensions to be represented. For example, one could define a `LIST [1:3] OF ARRAY [5:10] OF INTEGER`, which would in effect have two dimensions. How this ‘nesting’ works is shown in Table D.2.

Table D.2 *Understanding the ‘Nesting’ of EXPRESS Data types*

Component		(go to)		
1 ENTITY <code>entity_name</code> ; <code>attribute_name</code> : [OPTIONAL] <code>base_type</code> (2); END_ENTITY;				
2 <code>base_type</code>				
3 <code>simple_type</code>	BINARY			
	BOOLEAN			
	INTEGER			
	LOGICAL			
	NUMBER			
	REAL			
	STRING			
4 <code>named_type</code>	Entity_name	(1)		
	Defined_type	<code>simple_type</code> (3)		
		<code>constructed_types</code>	<code>Enumeration_type</code>	ENUMERATION
			<code>Select_type</code>	<code>named_type</code> (4)
		<code>aggregation_type</code> (5)		
5 <code>Aggregation_type</code>	array_type	ARRAY [m:n] OF [OPTIONAL] [UNIQUE] <code>base_type</code> (2)	Only the ARRAY type is able to have its elements declared as OPTIONAL. The ARRAY type is also able to have its elements declared as UNIQUE The LIST type is able to have its elements declared as UNIQUE	
	list_type	LIST [m:n] OF [UNIQUE] <code>base_type</code> (2)		
	bag_type	BAG [m:n] OF <code>base_type</code> (2)		
	set_type	SET [m:n] OF <code>base_type</code> (2)		

Example

complex_list : LIST [0:12] OF UNIQUE ARRAY [1:10] OF INTEGER;

This example defines a list of arrays. The list can contain zero to twelve arrays. Each array shall be different from every other array in the list, as specified by the **UNIQUE** keyword. There are zero, 10, 2×10 , ... , 12×10 integer values associated with the attribute named complex_list depending on the number of list elements.

D.5.4 Uniqueness in relationships

In general, there are two ways of specifying uniqueness:

- i. as part of the entity declaration
- ii. as part of the attribute's data type definition

In the first case, the **UNIQUE** rule applies to the entity (or at least one of its attributes). In other words, the domain of the **UNIQUE** rule is the entity. In the second case, the **UNIQUE** rule applies to (part of) the attribute. In other words, the domain of the **UNIQUE** rule is the attribute. This declaration within a cardinality specification adds additional complexity to the relationship.

Implicit uniqueness

The cardinality constraints defined in the forward and inverse directions can **imply** uniqueness. For example, when the upper bound of the inverse cardinality is equal to one (i.e. $q=1$) there is implied uniqueness in the relationship. It should be remembered that the cardinality in the **INVERSE** declaration constrains the relationship in the inverse direction (i.e. 'second_entity' to 'first_entity'), whereas a **UNIQUE** declaration constrains the relationship in the forward direction (i.e. 'first_entity' to 'second_entity'). It should also be noted that a **UNIQUE** declaration **cannot** be put in an **INVERSE** declaration.

Collective and distributive references

In EXPRESS, an attribute with an aggregation type represents a **collective** rather than a **distributive** reference. A collective reference is a single reference to a group of things, whereas a distributive reference is multiple references to single things. The semantic distinction is more obvious when the **UNIQUE** keyword is used in conjunction with an aggregate type.

Example

```
ENTITY garage;
  owned_cars : SET OF car;
  UNIQUE
    UR1 : owned_cars;
END_ENTITY;
```

In the example above, the attribute 'owned_cars' on its own could be considered either as a single reference to a set of cars (collective reference) or as multiple references to individual cars (distributive reference). The keyword **UNIQUE** emphasizes the semantic difference, such that the attribute is interpreted as a collective reference. Each car can then be owned by many garages, for it is a given set of cars which is uniquely owned whereas a given car can be in many sets.

Example

```
ENTITY x_ent;  
  sectors: ARRAY [1:10] OF ARRAY [11:14] OF UNIQUE a_attr;  
END_ENTITY;
```

The first array has 10 elements of data type ARRAY [11:14] OF UNIQUE 'a_attr'. There are a total of 40 elements of data type 'a_attr' in the attribute sectors. Within each ARRAY [11:14], no duplicates may occur; however, the same instance may occur in two different ARRAY [11:14] instances within a single 'sector'.

APPENDIX E EXPRESS-G

EXPRESS-G is a formal graphical notation for the display of data specifications defined in the EXPRESS language. The notation only supports a subset of the EXPRESS language. However, since the STEP community has endorsed its use in AP documentation, the Product Modeller must know the notation and capabilities of EXPRESS-G.

E.1 EXPRESS-G Notation

As there are many examples of EXPRESS-G diagrams in the CIS, this Section gives a brief overview of the EXPRESS-G notation. EXPRESS-G uses the following conventions defined in Annex D of the *EXPRESS Language Reference Manual*^[16].

The definition of types or entities is depicted by boxes with several possibilities as shown in Table E.1.

Table E.1 *Representing definitions in EXPRESS-G*

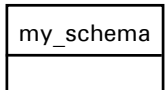

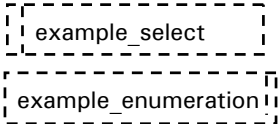
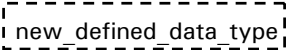
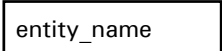
EXPRESS	Graphical symbols	Example
Schema	Rectangular solid box divided in two sub-boxes by a horizontal line. In the lower part is located the name of the schema	
simple data type	Rectangular solid box with a double vertical line at the right end of the box. The label within the box is the name of the data type	
Constructed data types	Rectangular dashed box. SELECT data type: double vertical line on the left ENUMERATION data type: double vertical line on the right	
defined data type	Rectangular dashed box. The label within the box is the name of the data type	
entity data type	Rectangular solid box. The label within the box is the name of the entity	

Table E.2 *References in EXPRESS-G*

EXPRESS	Graphical symbols	Example
page reference onto the current page	Relationship line terminated by a rounded box which contains a page number and the reference number. If several pages should be referenced, the box may contain a parenthesised list of the page numbers where references originated.	
page reference onto another page	Relationship line terminated by a rounded box, which contains a page number and the reference number.	
Definition referenced from another schema	a round box enclosing the name of the definition qualified by the schema name. The REFERENCE statement is represented by a dashed enclosing rectangular box.	
Definition used from another schema	a round box enclosing the name of the definition qualified by the schema name. The USE statement is represented by a solid enclosing rectangular box.	

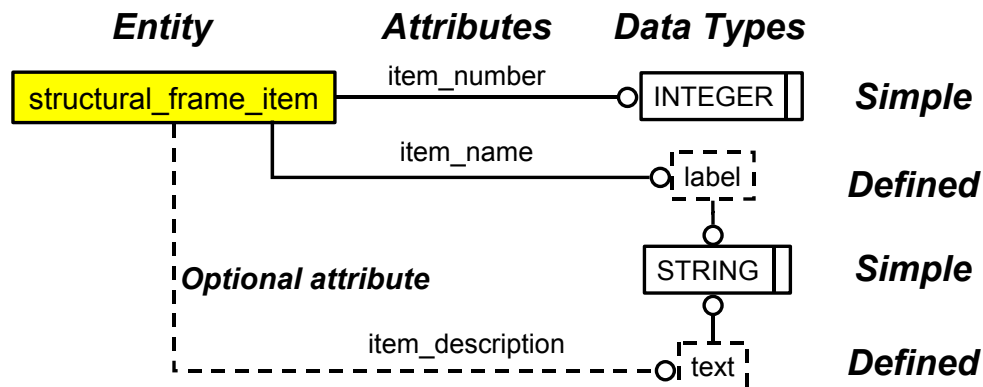
Table E.3 *Representing Relationships in EXPRESS-G*

EXPRESS	Graphical symbols	Example
Optional attribute	Dashed line	-----
Attribute non optional	Solid thin line	_____
Inheritance relationship (SUBTYPE/SUPERTYPE)	Solid thick line An <i>ONEOF</i> relation may be indicated by the digit 1 placed at a branching junction.	
Preference sense of a relationship is	Open circle at the end of the line and before the second term of the association.	
Cardinality of relationship with aggregation data types	On the relationship line for the attribute. The first letter of the aggregation data type (A, B, L, or S) is used. The default cardinality is one for a non-optional attribute, and zero-or-one for an optional attribute.	
Inverse attribute	Characters INV enclosed in parentheses	
Derived attribute	Characters DER enclosed in parentheses	
Constraints:	An asterisk flags the entities or attributes involved in a constraining clause *	*entity_involved_in_a_rule

E.2 Examples of EXPRESS-G as used in LPM/6

Some examples of how EXPRESS-G is used to illustrate LPM/6 are shown below. The equivalent EXPRESS construct is shown alongside each EXPRESS-G diagram

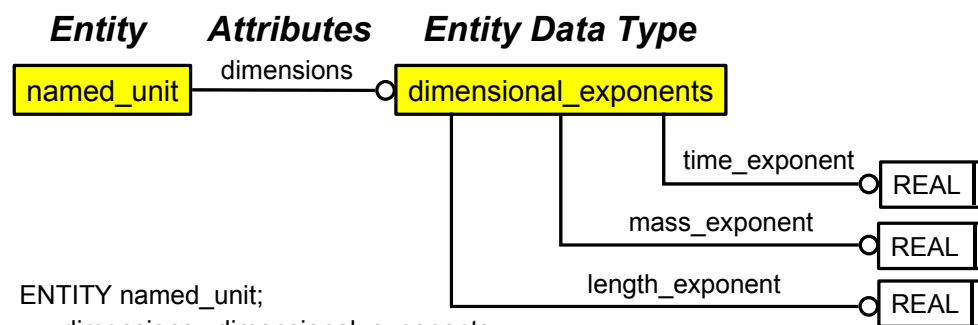
E.2.1 Simple and defined data types



```
ENTITY structural_frame_item;
    item_number : INTEGER;
    item_name : label;
    item_description : OPTIONAL text;
END ENTITY;
```

Figure E.1 *Representing Simple and defined data types in EXPRESS-G*

E.2.2 Entity data types



```
ENTITY named_unit;
    dimensions : dimensional_exponents;
END ENTITY;
```

```
ENTITY dimensional_exponents;
    length_exponent : REAL;
    mass_exponent : REAL;
    time_exponent : REAL;
END ENTITY;
```

Figure E.2 *Representing Entity data types in EXPRESS-G*

E.2.3 SUPERTYPES and SUBTYPES

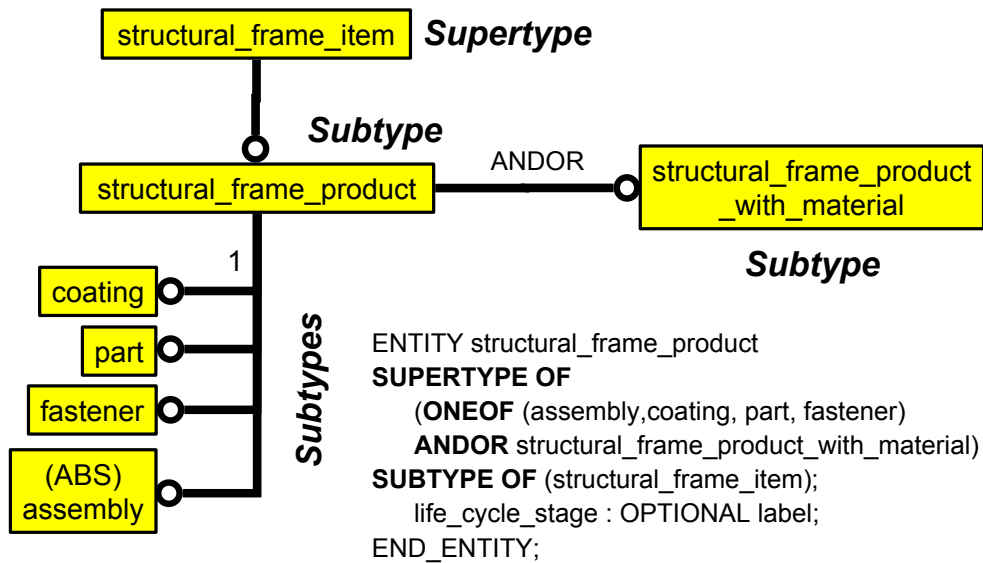


Figure E.3 Representing SUPERTYPES and SUBTYPES in EXPRESS-G

E.2.4 ENUMERATION data types

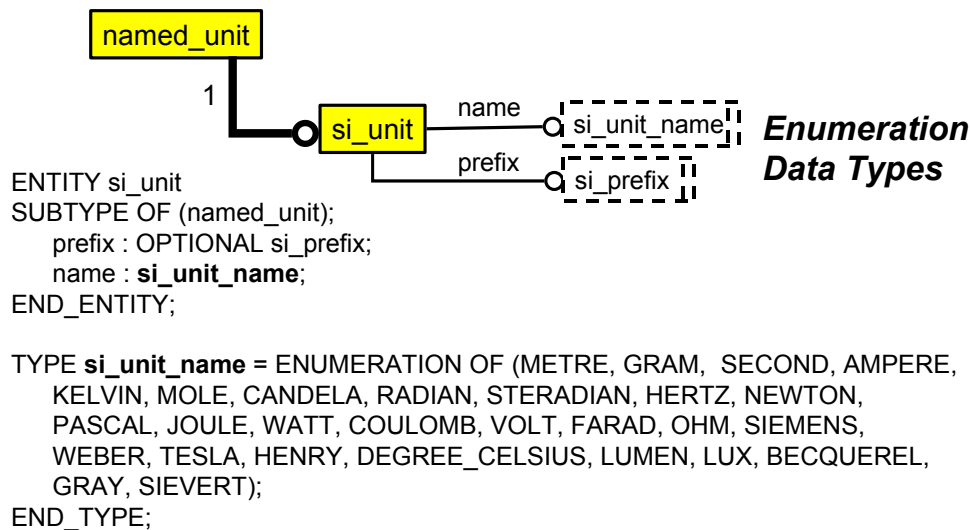


Figure E.4 Representing ENUMERATION data types in EXPRESS-G

E.2.5 SELECT data types

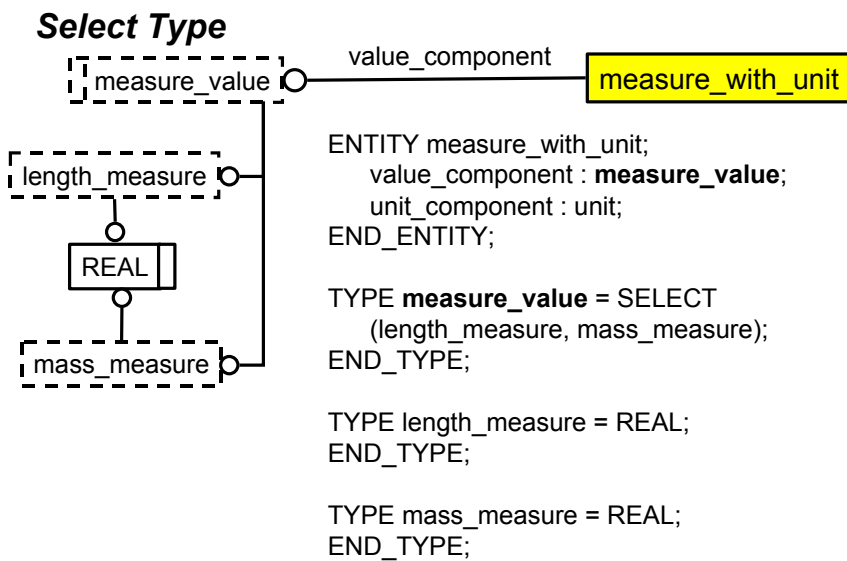


Figure E.5 Representing *SELECT* data types in EXPRESS-G (1)

SELECT data types may reference defined types (as shown above), or other entities (as shown below).

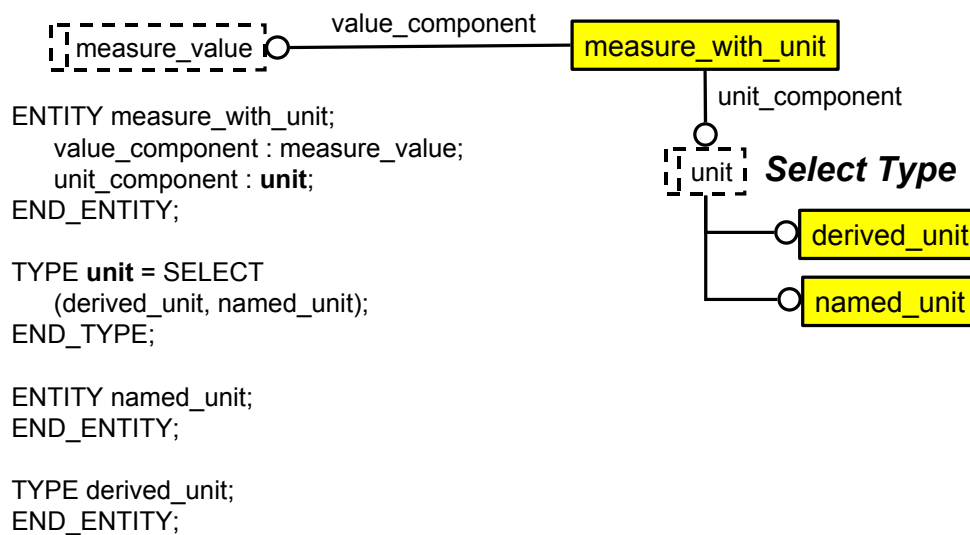
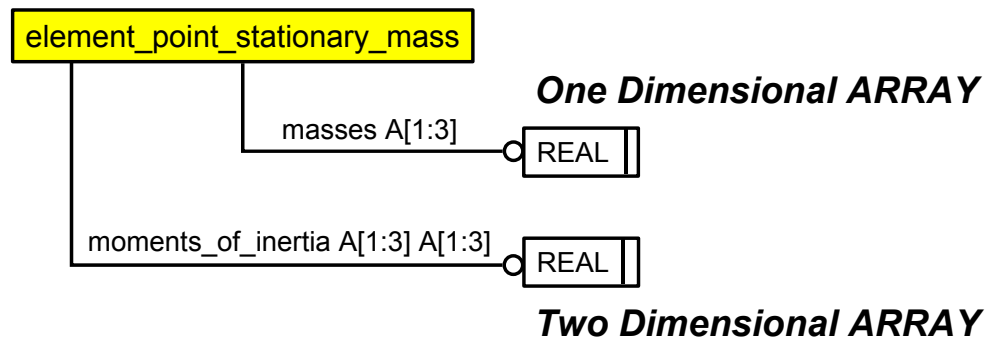


Figure E.6 Representing *SELECT* data types in EXPRESS-G (2)

E.2.6 Aggregation data types



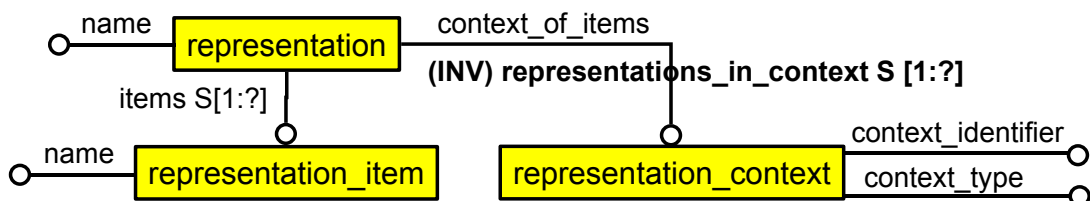
```

ENTITY element_point_stationary_mass
  masses : ARRAY [1:3] OF REAL;
  moments_of_inertia : ARRAY [1:3] OF ARRAY [1:3] OF REAL;
END_ENTITY;

```

Figure E.7 Representing aggregation data types in EXPRESS-G

E.2.7 INVERSE clauses



```

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

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

```

Figure E.8 Representing INVERSE clauses in EXPRESS-G

APPENDIX F STEP FILE STRUCTURE

The CIS specifies that the physical file format shall be in accordance with the 2002 International Standard issue STEP exchange file format^[17] (which incorporates the 1996 ‘Technical Corrigendum’). STEP file exchange implementations simply rely on conformance to an agreed EXPRESS schema and the use of an agreed physical file format.

F.1 The exchange structure

The exchange structure is described by an unambiguous, context-free grammar to facilitate parsing by software. The grammar is expressed in Wirth Syntax Notation^[43]. The form of the product data in the exchange structure is specified by using a mapping from the EXPRESS language^[16].

The exchange structure is a sequential file using a clear text encoding. It consists of two sections: the header section - providing data relating to the exchange structure itself, and the data section - providing the data to be transferred. The special token ‘ISO-10303-21;’ is used to open an exchange structure, and the special token ‘END-ISO-10303-21;’ is used to close an exchange structure. The special token ‘HEADER;’ and ‘DATA;’ is used to open the header and data section respectively. The special token ‘ENDSEC;’ is used to close both sections.

F.1.1 The header section

Three header section entities are specified and one instance of each is required to occur in the header section of every exchange structure. The header section entities are **file_description**, **file_name**, and **file_schema**, and they appear in that order. (They are fully defined in header_section_schema in clause 9.2 of ISO 10303-21: 1994.)

EXPRESS specification

The EXPRESS specification for the header section schema is as follows:

SCHEMA header_section_schema;

ENTITY file_description;
 description: LIST [1:?] OF STRING (256);
 implementation_level: STRING (256);
 END_ENTITY;

ENTITY file_name;
 name: STRING (256);
 time_stamp: STRING (256);
 author: LIST [1:?] OF STRING (256);
 organization: LIST [1:?] OF STRING (256);
 preprocessor_version: STRING (256);
 originating_system: STRING (256);
 authorization: STRING (256);
 END_ENTITY;

```
ENTITY file_schema;
  schema_identifiers: LIST [1:1] OF schema_name;
END_ENTITY;
```

```
TYPE schema_name = STRING (1024);
END_TYPE;
```

An example of a populated header schema as encoded in the exchange structure with comments is shown below. The token ‘/*’ opens a comment – and the token ‘*/’ closes the comment.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION ( ('{cimsteel logical product model version (6) object (1) structural-
  frame-schema(1)}',                                /* OBJECT IDENTIFIER */
  'CC110', 'CC170', 'CC118', 'CC009'),                /* CONFORMANCE CLASSES */
  '2;1');                                             /* IMPLEMENTATION LEVEL */
FILE_NAME('dmc1.stp',                                /* FILENAME */
  '2003-4-28 T16:10:45',                              /* CREATION DATE */
  ('Andrew Crowley'),                                /* AUTHOR */
  ('Steel Construction Institute'),                    /* ORGANIZATION */
  '1.5.1 (compiled 17:44 11/24/1998)',                /* PROCESSOR VERSION */
  'NIST EXPRESSO',                                    /* PROCESSOR */
  'AJC');                                             /* AUTHORIZATION */
FILE_SCHEMA (('STRUCTURAL_FRAME_SCHEMA'));
ENDSEC;

DATA;
....
ENDSEC;

END-ISO-10303-21;
```

It should be noted that none of the attributes are optional; a value must be given for each of the attributes of the three entities of the header schema. Furthermore, the data in the DATA section of the physical file is meaningless without the appropriate schema. Thus, the FILE_SCHEMA must provide the name of the appropriate schema as defined by the EXPRESS schema declaration. For CIS/2, the name of the schema is ‘STRUCTURAL_FRAME_SCHEMA’. This will remain constant for all CIS/2 data exchange activities.

In the example above, the ‘Object Identifier’ (provided in *Volume 4*^[8]) has been placed in the first list element of the attribute ‘description’ of the entity ‘file_description’. It is recommended that implementations of CIS/2 use this convention, as STEP Part 21 does not provide a place for the ‘Object Identifier’ object id in the encoded file. The ‘Object Identifier’ is intended to provide an unambiguous identification of the schema amongst other schemas and other information objects in an open information system.

The example above also uses a convention that should be adopted by implementations of CIS/2; i.e. it places the list of Conformance Classes (CCs) in the second and subsequent list elements of the attribute ‘description’ of the entity ‘file_description’. The data provided in the data section is thereby specified as being conforming with those CCs. Although this is not a Conformance Requirement of CIS/2, software vendors are strongly recommended to follow this practice, as it will speed up the interrogation of a physical file by a receiving import translator.

F.1.2 The data section

The data section contains instances of entities that correspond to the EXPRESS schema governing the exchange structure in the header section. Six simple data types are used in exchange structures: INTEGER, REAL, STRING, ENTITY instance name (i.e. the #number), ENUMERATION, and BINARY.

Each instance should be represented at most once in the data section; distinct entity instances shall have distinct entity instance names (i.e. the #number). The entity instances need not be ordered in the exchange structure. An entity instance may be referenced before its name occurs as an identifier in the exchange structure. Examples of these instances, together with the corresponding EXPRESS mapping, are given in the next Section.

F.2 Mapping of EXPRESS data types to the exchange structure

F.2.1 Optional values

Where an Optional attribute has not been given a value, it is encoded as a dollar sign “\$”.

F.2.2 Boolean

The EXPRESS element of BOOLEAN is treated as a pre-defined enumerated data type with a value encoded as “T” (for TRUE) and “F” (for FALSE).

F.2.3 Logical

The EXPRESS element of LOGICAL is treated as a pre-defined enumerated data type with a value encoded as “T” (for TRUE), “F” (for FALSE) and “U” (for UNKNOWN).

Example entity definition in EXPRESS

```
ENTITY widget;
  A : OPTIONAL REAL;
  B : BOOLEAN;
  C : LOGICAL;
END_ENTITY;
```

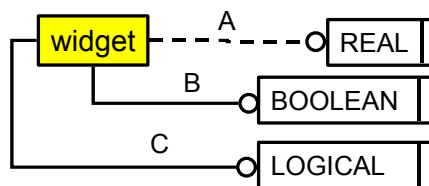


Figure F.1 Example entity definition represented EXPRESS-G

Sample entity instance in data section

#1 = WIDGET (\$, .T., .F.);

F.2.4 Aggregation data types

The EXPRESS elements of LIST is mapped to the exchange structure as a list data type. If an entire list is optional and does not exist, the list is encoded by a dollar sign “\$”. If the list is empty, the list is encoded as a left parenthesis “(” followed by a right parenthesis “)”.

Example entity definition in EXPRESS

	Comments
ENTITY widget;	
attribute1: LIST [0:?] OF INTEGER;	-- A
attribute2: LIST [1:?] OF INTEGER;	-- B
attribute3: OPTIONAL LIST [1:?] OF INTEGER;	-- C
attribute4: REAL;	-- D
END_ENTITY;	

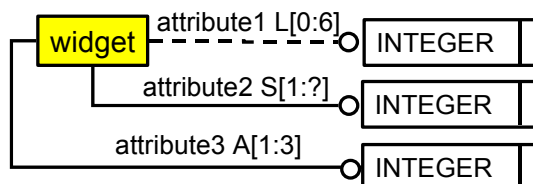


Figure F.2 Example entity definition represented EXPRESS-G

Sample entity instance in data section:

#4 = WIDGET((), (1,2,4), \$, 2.56);

Comments

A: ‘attribute1’ is an empty list (list with zero elements) in this instance.

B: ‘attribute2’ contains three elements in this instance.

C: ‘attribute3’ does not have a value in this instance.

D: ‘attribute4’ has a value of 2.56 in this instance.

The EXPRESS elements of aggregation data types (ARRAY, SET, & BAG) are all mapped in a similar way.

F.2.5 Enumeration

The actual value in an instance of the ENUMERATION is one of the enumerated values in the EXPRESS schema, in capital letters and delimited by full stops.

Example entity definition in EXPRESS

	Comments
TYPE primary_colour = ENUMERATION OF (red, green, blue);	
END_TYPE;	
ENTITY widget;	
p_colour : primary_colour;	-- A
END_ENTITY;	

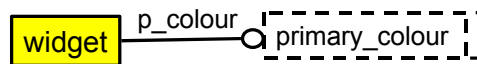


Figure F.3 *Example entity definition represented EXPRESS-G*

Sample entity instance in data section

#2 = WIDGET(.RED.);	-- A
---------------------	------

Comments

A: The value of the attribute 'p_colour' in this entity instance is RED.

F.2.6 Select data types

An EXPRESS select data type defines a list of data types, called the 'select-list', whose values are valid instances of the select data type. An instance of a select data type should be a value of at least one of the data types in the select-list. For instances of simple defined types and enumeration data types, the KEYWORD identifies the data type of the instance (which will be one of the types in the select-list).

If the data type in the select list is itself a select data type, then the encoding rules are applied recursively. Only instances of entity data types, simple data types and enumeration data types can actually be encoded.

Where a select data type references a SUPERTYPE entity, the select-list includes all of the SUBTYPE entities of that SUPERTYPE entity type, since a SUBTYPE is an instance of a SUPERTYPE entity.

Note, this rule is defined in the 1996 'Technical Corrigendum' and differs from the original 1994 edition of ISO 10303-21.

Example entity definition in EXPRESS

```

ENTITY measure_with_unit
  SUPERTYPE OF (ONE OF
    (length_measure_with_unit,
     mass_measure_with_unit,
     time_measure_with_unit));
  value_component : measure_value;
  
```

```

    unit_component : unit;
WHERE
    WR41T1A : valid_units (SELF);
END_ENTITY;

TYPE unit = SELECT (named_unit, derived_unit);
END_TYPE;

TYPE measure_value = SELECT
    (length_measure, mass_measure, time_measure, ... );
END_TYPE;

ENTITY named_unit
    SUPERTYPE OF ((ONEOF(length_unit, mass_unit, time_unit, ... )) ANDOR
        (ONEOF(si_unit, conversion_based_unit, context_dependent_unit)));
    dimensions : dimensional_exponents;
END_ENTITY;

ENTITY derived_unit;
    elements : SET [1:?] OF derived_unit_element;
WHERE
    WR41T1 : ( SIZEOF ( elements ) > 1 ) OR
        (( SIZEOF ( elements ) = 1 ) AND ( elements[1].exponent <> 1.0 ));
END_ENTITY;

```

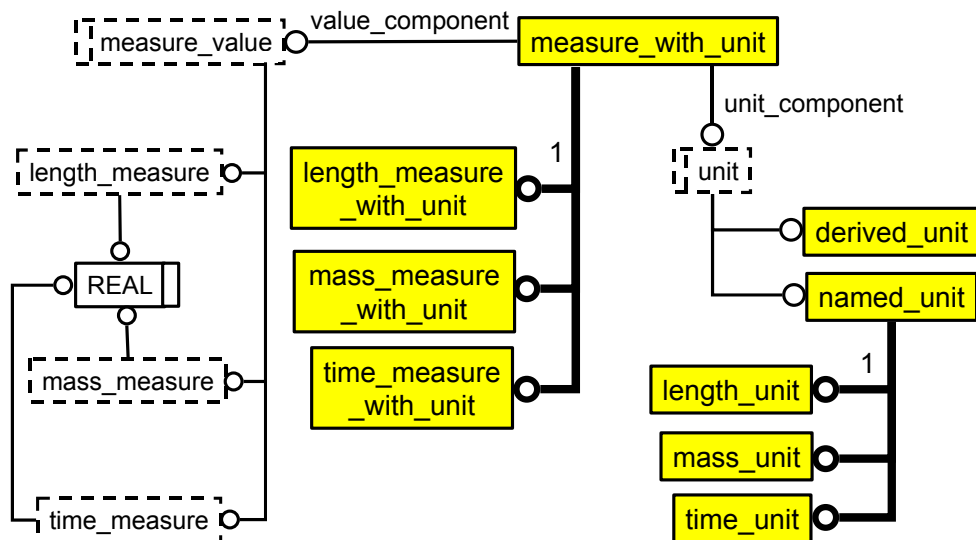


Figure F.4 Example entity definition represented EXPRESS-G

Sample entity instance in data section

```
#202 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(498.0), #1021);
```

```
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
```

```
#216 = INERTIA_MEASURE_WITH_UNIT (NUMERIC_MEASURE(342200.0), #1151);
```

```
#1149 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.CENTI.,.METRE.));
#1150 = DERIVED_UNIT_ELEMENT (#1149, 4.0);
#1151 = INERTIA_UNIT ((#1150));
```

F.2.7 Derived attributes

The derived attributes of an entity shall not be mapped to the exchange structure, except where an attribute in a SUBTYPE redefines an attribute in a SUPERTYPE (see later).

F.2.8 Entities with other entities as attributes

If an entity instance is specified as an attribute of a second (referencing) entity instance, the first (referenced) entity instance shall be mapped to the exchange structure as an entity instance name (i.e. the #number).

Example entity definition in EXPRESS

```
ENTITY widget;
  widget_material : material;
END_ENTITY;
```

```
ENTITY material;
  name : STRING;
END_ENTITY;
```

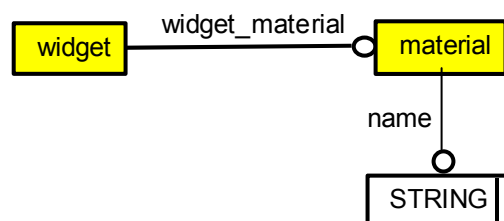


Figure F.5 Example entity definition represented EXPRESS-G

Sample entity instance in data section

```
#1 = WIDGET (#3);
#3 = MATERIAL ('steel');
```

F.2.9 Subtypes & SUPERTYPES

Entities defined as SUBTYPES or SUPERTYPES of other entities shall be mapped to the exchange structure using one of two mapping rules, internal mapping or external mapping. One mapping rule applies to each instance of a SUBTYPE entity. The selection of which mapping rule applies depends on the conformance class chosen for the implementation.

Instances of an entity having a SUBTYPE clause in the EXPRESS entity declaration are known as 'complex entity instance', in that they involve attributes from more than one entity-type declaration. The treatment of complex entity instances is formally defined in clause 11.2.5 of STEP Part 21, and depends upon which of two STEP 'implementation conformance class' is being used:

- If conformance class 1 is being used, a complex entity instance which has a linear inheritance graph (and thus can be given the entity name of the lowest SUBTYPE), must mapped to the exchange structure using the internal mapping rules. All other complex entity instances, which will have a branched inheritance graph (typically arising from an ANDOR SUPERTYPE declaration) and cannot be given a single entity name, must mapped to the exchange structure using the external mapping rules.
- If conformance class 2 is being used, all complex entity instances are mapped using the external mapping rules.

Which implementation conformance class has been employed in a particular Part 21 exchange file is specified by the value assigned to the attribute 'implementation_level' of the entity 'file_description' in the header section. This will be set either to character '2;1' or '2;2', as shown below:

Example header entity definition in EXPRESS

	Comments
ENTITY file_description;	
description: LIST [1:?] OF STRING (256);	
implementation_level: STRING (256);	-- A
END_ENTITY;	

Sample entity instance in data section

ISO-10303-21;	
HEADER;	
FILE_DESCRIPTION (
('{cimsteel logical product model version (6) object (1) structural-frame-schema(1)}',	
'CC110', 'CC170', 'CC118', 'CC009'),	
'2;1');	-- A

Comments

A: The example indicates that the file has been mapped in accordance with implementation conformance class 1 of version 2 of STEP Part 21.

F.2.10 Internal mapping

The order in which the inherited and explicit attributes appear in the exchange structure shall be determined as follows:

- i. All inherited attributes shall appear sequentially prior to the explicit attributes of any entity.
- ii. The attributes of a SUPERTYPE entity shall be inherited in the order they appear in the SUPERTYPE entity itself.
- iii. If the SUPERTYPE entity is itself a SUBTYPE of another entity, then the attributes of the higher SUPERTYPE entity shall be inherited first.
- iv. When multiple SUPERTYPE entities are specified, the attributes of SUPERTYPE entities shall be processed in the order specified in the SUBTYPE OF expression.

Example of internal mapping

	Comments
ENTITY aa SUPERTYPE OF (ONEOF(bb, cc)); attrib_a: STRING; END_ENTITY;	-- A
ENTITY bb SUBTYPE OF (aa); attrib_b: INTEGER; END_ENTITY;	-- B
ENTITY cc SUBTYPE OF (aa); attrib_c: REAL; END_ENTITY;	-- C
ENTITY dd; attrib_d: aa; END_ENTITY;	-- D

Sample entity instance in data section

	Comments
#1 = AA ('SAMPLE STRING');	-- A
#2 = BB ('ABC', 666);	-- B
#3 = CC ('XYZ', 99.99);	-- C
#4 = DD (#1);	-- D
#5 = EE (#2);	-- D
#6 = FF (#3);	-- D

Comments

- A: Since it was not an abstract SUPERTYPE, the SUPERTYPE entity 'aa' can be instantiated in an exchange structure. Note that it contains only its attribute 'attrib_a' when it is instantiated.
- B: The entity 'bb' is a SUBTYPE of 'aa' and therefore its instances will contain the attributes of both 'aa' and 'bb'.
- C: The entity 'cc' is a SUBTYPE of 'aa' and therefore its instances will contain the attributes of both 'aa' and 'cc'.
- D: The entity 'dd' references entity 'aa' as an attribute. Therefore, an instance of entity 'dd' may reference any of #1, #2, or #3.

F.2.11 External mapping

The formal description of external mapping is not reproduced here (see 11.2.5.3 of ISO 10303-21: 2002), but the syntax is illustrated below.

Example of external mapping

	Comments
ENTITY aa SUPERTYPE OF (bb ANDOR cc); attrib_a: STRING; END_ENTITY;	-- A
ENTITY bb SUBTYPE OF (aa); attrib_b: INTEGER; END_ENTITY;	-- B
ENTITY cc SUBTYPE OF (aa); attrib_c: REAL; END_ENTITY;	-- C
ENTITY dd; attrib_d: aa; END_ENTITY;	-- D

Sample entity instance in data section

	Comments
#1 = BB ('SAMPLE STRING', 15);	-- A
#2 = CC ('SS', 3.0);	-- B
#3 = (AA('ASTRID')BB(17)CC(4.0));	-- C
#4 = DD (#1);	-- D
#5 = DD (#2);	-- D
#6 = DD (#3);	-- D
#7 = AA ('ABC');	-- E

Comments

- A: #1 is an instance of 'aa' and 'bb' combined ('bb' inherits an attribute from its SUPERTYPE 'aa'). Internal mapping applies because the inheritance graph is linear.
- B: #2 is an instance of 'aa' and 'cc' combined ('cc' inherits an attribute from its SUPERTYPE 'aa'). Internal mapping applies because the inheritance graph is linear.
- C: #3 is an instance of 'aa', 'bb' and 'cc' combined (external mapping has been used thus the attribute inherited from the SUPERTYPE 'aa' is shown next to the SUPERTYPE name).
- D: #4, #5, and #6 are all instances of the entity 'dd', which references entity 'aa' as an attribute. An instance of entity 'dd' may, therefore, legally reference any of #1, #2, #3. ('dd' does not create a complex entity instance.)
- E: #7 is an instance of the non-abstract SUPERTYPE 'aa'. Internal mapping applies because the evaluated set contains only one member.

Example entity definition in EXPRESS

```

ENTITY named_unit
  SUPERTYPE OF (ONE OF (length_unit, mass_unit, time_unit)
    ANDOR (ONEOF (si_unit, conversion_based_unit)));
  dimensions : dimensional_exponents;
END_ENTITY;

ENTITY length_unit ;
  SUBTYPE OF (named_unit);
END_ENTITY;

ENTITY si_unit ;
  SUBTYPE OF (named_unit);
  prefix : OPTIONAL si_prefix;
  name : si_unit_name;
  DERIVE
    SELF.named_unit.dimensions : dimensional_exponents := dimensions_for_si_unit
    (SELF.name);
END_ENTITY;

```

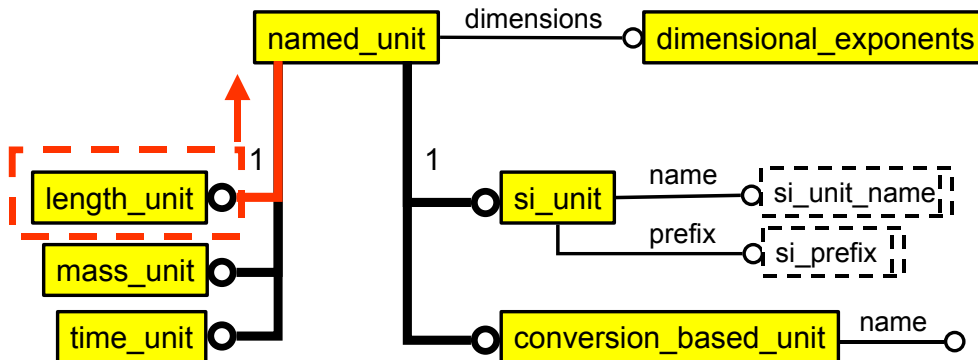


Figure F.6 Example of internal mapping represented in EXPRESS-G

Sample entity instance in data section (internal mapping)

```

/* Length Unit (mm) */
#1021 = LENGTH_UNIT(#1022);
#1022 = DIMENSIONAL_EXPONENTS (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

```

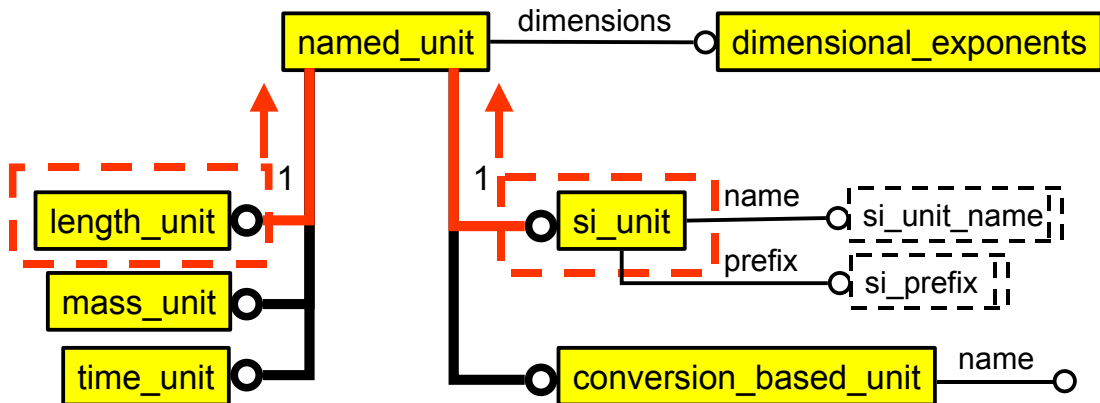


Figure F.7 Example of external mapping represented in EXPRESS-G

Sample entity instance in data section (external mapping)

```
/* Length Unit (mm) */
#1020 = SI_UNIT (*, .MILLI., .METRE.);
#1021 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.MILLI., .METRE.));
```

F.2.12 Entities with attributes redefined as an effect of a DERIVE

If a SUBTYPE entity redefines an attribute of its SUPERTYPE using the DERIVE clause, that attribute in the SUPERTYPE shall be encoded with an asterisk “*”.

Example entity definition in EXPRESS

```
ENTITY point
SUPERTYPE OF (ONE OF (point_on_curve) );
x: REAL;
y: REAL;
z: REAL;
END_ENTITY;
```

```
ENTITY point_on_curve
SUBTYPE OF (point);
u: REAL;
c: curve;
DERIVE
SELF\point.x: REAL := fx(u, c);
SELF\point.y: REAL := fy(u,c);
SELF\point.z: REAL := fz(u,c);
END_ENTITY;
```

```
ENTITY curve;
attr: STRING;
END_ENTITY;
```

Sample entity instance in data section

	Comments
#1 = CURVE ('curve_attribute');	
#2 = POINT_ON_CURVE (*, *, *, 0.55, #1);	-- A
#3 = POINT (2.0, 3.0, 4.0);	-- B

Comments

- A: Because a SUBTYPE with a derived attribute is used here, the attributes 'x', 'y', and 'z' are all replaced by asterisks.
- B: Because point is not an ABSTRACT SUPERTYPE, it is possible to have an instance of point in the exchange structure. The attributes 'x', 'y', and 'z' appear as normal.

F.2.13 Attributes redefined as an effect of a SUBTYPE declaration

A SUPERTYPE attribute may be redefined by a SUBTYPE entity if the attribute name in the SUBTYPE is the same as in the SUPERTYPE. In this case, there is no effect no the encoding - the value of the redefined attribute in the SUBTYPE shall be encoded as an attribute of the SUPERTYPE. The redeclared attribute shall be ignored, that is, it shall not be considered an attribute of the SUBTYPE for encoding purposes.

Note, this rule is defined in the 1996 'Technical Corrigendum' and differs from the original 1994 edition of ISO 10303-21.

Example entity definition in EXPRESS

```
ENTITY aaa SUPERTYPE OF (ONE OF (bbb, ccc) );
  a1: NUMBER;
  a2: curve;
END_ENTITY;
```

```
ENTITY bbb SUBTYPE OF (aaa);
  SELF\aaa.a1: INTEGER;
  b: REAL;
END_ENTITY;
```

```
ENTITY ccc SUBTYPE OF (aaa);
  SELF\aaa.a2: line;
END_ENTITY;
```

Sample entity instance in data section

	Comments
#1 = LINE (...);	
#2 = CURVE (...);	
#3 = BBB (1, #2, 0.5);	-- A
#4 = CCC (1.5, #1);	-- A

Comments

A: For instances #3 and #4 the encoding is the same as if there were no redeclared attributes in the entities 'bbb' and 'ccc'.

F.2.14 Unmapped constructs

The following EXPRESS constructs do not map into the exchange structure:

WHERE rules

INVERSE attributes

SCHEMA declaration (the schema name appears in the header section)

CONSTANTS

RULEs

Remarks

F.3 The Scope structure

The Scope structure within entity instances is a mechanism for providing a scope of reference and for defining existence relationships among entity instances. An EXPRESS schema specifies entity data types and establishes relationships among entity data types (where the data type is another entity). In addition to these relationships, the exchange structure can convey existence dependencies among entity instances.

An entity instance 'B' is said to be 'existence-dependent' on an entity instance 'A', if 'B' can exist only when 'A' exists. Existence dependence may be independent of the data types of instances 'A' and 'B', and it may be independent of any reference between the instances 'A' and 'B'.

Sample instance in data section of physical file

```
#1001 = &SCOPE
#1 = CARTESIAN_POINT ('origin point 1', (0.0, 0.0, 0.0));
#501 = REPRESENTATION ('coordinate system', (#1, #201), #502);
#502 = (GEOMETRIC_REPRESENTATION_CONTEXT(3)
  GLOBAL_UNIT_ASSIGNED_CONTEXT((#1021))
  REPRESENTATION_CONTEXT('context for coord system, 'units for coordinates'));
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
#201 = AXIS2_PLACEMENT_3D ('test placement', #1, #301, #302);
#301 = DIRECTION ('z vector', (0.0, 0.0, 1.0));
#302 = DIRECTION ('x vector', (1.0, 0.0, 0.0));
ENDSCOPE COORD_SYSTEM_CARTESIAN_3D ('my coord system', 'my use', 'right
handed', 3, #201);
```

Entity instance #1001 is of type 'coord_system_cartesian_3d' is defined by an instance of 'axis2_placement_3d' and a 'cartesian_point'. (Both of these geometric constructs also require a context.) These instances (#201 and #1) are in the scope of instance #1001 and it can be inferred that they are existence dependent on instance #1001. The 'direction'

instances that define the ‘axis2_placement_3d’ are also in the scope of #1001 and are dependent on it for their existence.

Sample instance in data section of physical file (2)

```
#1 = &SCOPE
#601 = PART_PRISMATIC_SIMPLE (1, 'PP1', 'columns main component', $, .ROLLED.,
    'piece mark 1', $, #201, #701, $, $);
#701 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(10250.0), #1021);
#1021 = (LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
#201 = SECTION_PROFILE_I_TYPE (1, 'W14x16x500', 'Wide flange beam to ASTM',
    'Plastic', $, .F., #202, #203, #204, #205, #206);
#202 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(498.0), #1021);
#203 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(432.0), #1021);
#204 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(55.6), #1021);
#205 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(88.9), #1021);
#206 = LENGTH_MEASURE_WITH_UNIT (LENGTH_MEASURE(280.0), #1021);
ENDSCOPE /#601, #1021/ DESIGN_PART ('left column main component', #601, (#401), $);

#401 = ASSEMBLY_DESIGN_STRUCTURAL_MEMBER_LINEAR (1, 'M1', 'left hand
    column', 'preliminary', 1, .LOW., .F., .F., (), (), .T., .COMBINED_MEMBER.,
    .PRIMARY_MEMBER., .COLUMN.);

#412 = DESIGN_PART ('right column main component', #601, (#402), $);

#402 = ASSEMBLY_DESIGN_STRUCTURAL_MEMBER_LINEAR (2, 'M2', 'right hand
    column', 'preliminary', 2, .LOW., .F., .F., (), (), .T., .COMBINED_MEMBER.,
    .PRIMARY_MEMBER., .COLUMN.);
```

Instances #601 (‘part_prismatic_simple’) and #1021 (‘length_unit’) are exported from the SCOPE structure of ‘design_part’ entity instance #1. This makes it possible for entity instance #601 to be referenced in an attribute of the second ‘design_part’ entity instance #412, even though instance #412 is existence-dependent on ‘design_part’ instance #1, but is the subject of an ‘independent’ second ‘design_part’ object.

F.4 Use of STEP Part 21 for the CIS

When implementing the CIS for exchanging files between applications, vendors should produce translators that are capable of reading or writing physical files in accordance with STEP Part 21^[17]. The requirements of the physical file format have been briefly described above, and are described in more detail in STEP Part 21. There are, however, a few limitations imposed for the CIS as explained in the following Sections.

F.4.1 Conformance

CIS implementation is required to be in accordance with conformance class 1.

F.4.2 Scope structure

The scope structure described in clause 10.3 of Part 21 and Section F.3 is not used in Basic CIS implementations. More advanced implementations may use the scope structure to simplify the management of data instances. Where a DMC CIS Translator supports

the scope structure, this must be stated in the translator's documentation. Although it is not a conformance requirement for CIS Translators to support the scope structure, it is a requirement that this capability must be explicitly declared when submitting an application for conformance testing.

F.4.3 User-defined entity instances

User-defined entity instances - described in clause 10.4 - are not used (or allowed) in CIS implementations.

F.4.4 Populating the HEADER section

There are two sections to a physical file: the HEADER and the DATA section. The HEADER section is populated using a STEP toolkit, while the DATA section must be read with the EXPRESS schema (the data in the DATA section is meaningless without reference to the appropriate schema). The HEADER section states the name of the schema used to configure the physical file. **For CIS/2, this will be 'STRUCTURAL_FRAME_SCHEMA'.**

CIS implementations are required to place the value of the 'Object Identifier' (provided in *Volume 4⁽⁸⁾*) in the first list element of the attribute 'description' of the entity 'file_description'.

Although it is not a Conformance Requirement of CIS/2, software vendors are strongly recommended to place the list of the Conformance Classes (CCs) relevant to the data in the data section in the second and subsequent list elements of the attribute 'description' of the entity 'file_description'. The data provided in the data section is thereby specified as being conforming with those CCs.

Sample header entity instance in header section

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (
    ('{cimsteel logical product model version (6) object (1) structural-frame-schema(1)}',
    'CC110', 'CC170', 'CC118', 'CC009'), '2;1');
FILE_NAME('dmc1.stp', '1999-5-28 T18:15:13', ('Andrew Crowley'), ('Steel Construction
    Institute'), '1.5.1 (compiled 17:44 11/24/1998)', 'NIST EXPRESSO', 'AJC');
FILE_SCHEMA (('STRUCTURAL_FRAME_SCHEMA'));
ENDSEC;
```

APPENDIX G STANDARD DATA ACCESS INTERFACE (SDAI)

The following is based upon the current edition of STEP Part 22^[18].

G.1 Introduction

Part 22 of STEP specifies a **Standard Data Access Interface** (SDAI), an application programming interface (API) to data that has been defined using EXPRESS^[16]. Together with standard data definitions, it will facilitate integration of different software components from different vendors.

Application software may be written using specific data access methods for specific data storage technologies. The use of a different data storage technology or data access method may require application software to be significantly modified, because each data storage technology has its own specific set of operations which it performs upon the data it maintains. This set of operations is referred to as a **data access interface**. If data storage technologies use the same standard data access interface, then application software can be written independently of all data storage systems.

The SDAI defines operations that give the application programmer the capability to manipulate data through this interface exactly as it is described in the defining schema or schemas.

G.2 SDAI Overview

G.2.1 Operations and the session state

Once an SDAI session has been initiated, the SDAI operations manipulate instances of entity types defined in application schemas and SDAI schemas. The session in which these operations take place has a state. Information concerning the session and its state is available as instances of the **SDAI session schema** and **SDAI population schema** during the session. Each makes a set of SDAI operations available. Some operations have the effect of changing the state of a session. STEP Part 22 offers the following operations of interest to developers of CIS/2 translators:

1. event recording
2. start/end transaction, commit, abort
3. validate global/where rule, uniqueness, reference
4. get (complex) entity definition
5. create/copy/delete entity instance
6. put/get/test attribute
7. save/undo changes
8. find entity instance users
9. create aggregate, get/put/remove current member

G.2.2 Repositories, schema instances, transactions and SDAI-models

The SDAI defines an interface between an application and the environment in which entity instances exist. Two aspects of that environment are known as **repositories** and as **schema instances**. Repositories are data storage facilities. Schema instances are logical collections of **SDAI-models** from which a set of instances can be derived. This set of entity instances defines the domain over which references between entity instances and global rule validation are supported. Schema instances must be created within a repository.

It should be noted that a repository may be implemented in memory, as a single database, multiple databases, a single file, a collection of files, or any other method.

Entity instances are created within SDAI-models created within a repository. The entity instances making up each SDAI-model are based on a single EXPRESS schema with the interface specifications resolved. Entity instances stored in one SDAI-model may refer to entity instances stored in another SDAI-model, provided that there exists a schema instance with which both SDAI-models are associated. The two SDAI-models must be based upon the same EXPRESS schema or be based upon two EXPRESS schemas that were defined as having domain equivalent constructs in order to be associated with the same schema instance. One SDAI model may be associated with more than one schema instance.

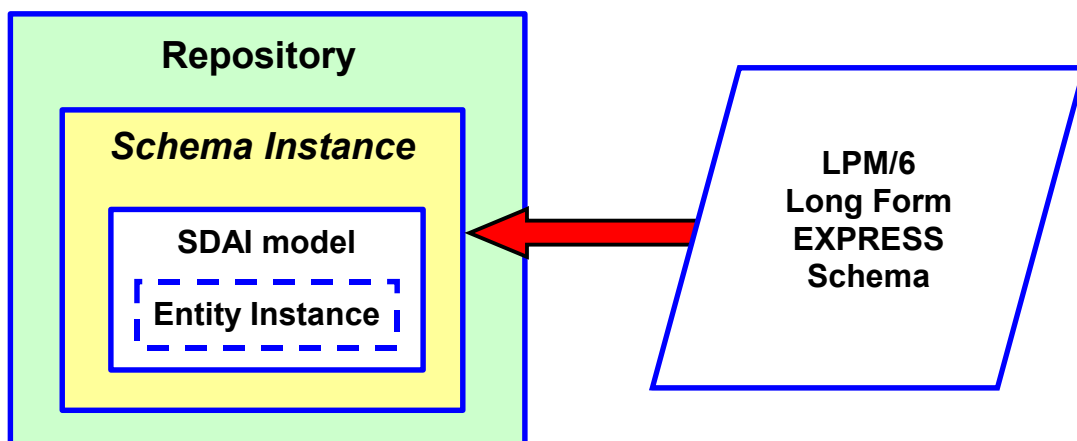


Figure G.1 *Repositories, schema instances, and SDAI-models*

G.2.3 Transactions and access modes

A **transaction** consists of a series of operations whose effect may be saved or undone as a unit. Within a transaction, access to specific repositories and SDAI-models is managed. Transactions and access to SDAI-models have **modes** associated with them: read-write and read-only. The read-write mode allows the application to access and change the instances in the SDAI-models. The read-only mode allows the application to access the instances in the SDAI-models but does not allow them to be changed. This specification prohibits an SDAI-model being accessed in read-write mode within a transaction that was initiated as read-only.

G.2.4 Dictionary and session data

The SDAI session schema describes the structure of an SDAI session. The SDAI population schema describes the structure of population data used to manage the

application data created by a user based upon an application schema. These SDAI schemas include definitions of such things as sessions, transactions, schema instances, repositories, and SDAI-models.

In order to allow applications to have access to the information about the schema defining their data the SDAI provides a data dictionary. The structure of this dictionary is defined by an EXPRESS schema: the SDAI dictionary schema. Figure G.2 describes the relationships between application data, dictionary and session data, the application program and the SDAI operations in simplified terms.

The entity or application instance data access operations do not provide support for the modification of session, population or dictionary data. Changes to this data may occur as the consequence of special operations whose purpose is to manage the SDAI environment. These operations are the only mechanism provided by the SDAI to modify any of the session, population or dictionary data. The basic data access operations (i.e., entity instance operations) are used to read the session, population and dictionary instances in the same manner as application instances. Not all operations are required to support access to the population data, as these instances are not required to exist within an SDAI-model.

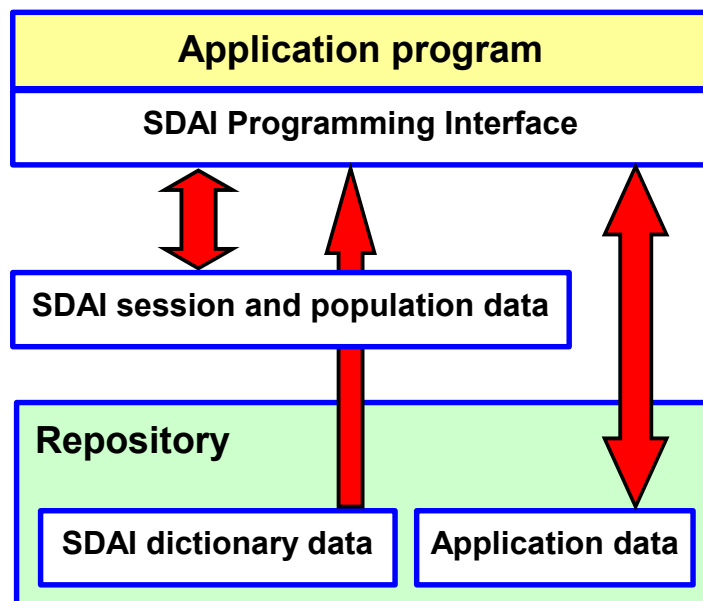


Figure G.2 *Dictionary and session data*

G.2.5 SDAI data type schemas

The SDAI data type schema defines the relationship between the types of entity instances that exist in an SDAI implementation. Support of the SDAI data type schema hierarchy is required of all implementations. Characteristics of these instances are not specified in this schema, as they are implementation-specific details. However, characteristics of these instances, and others, are described in the SDAI parameter data schema.

The SDAI parameter data schema describes, in abstract terms, the various types of instance data that are referenced by the interface, and its sole purpose is to help clarify the specification of the SDAI operations. The SDAI parameter data schema need not be instantiated within an SDAI implementation. It declares several of the same entities as

found in the SDAI data type schema but adds characteristics useful for further describing the behaviour of instances of those entities.

G.3 Early & Late Bindings

As SDAI was developed to be independent of any programming language, it includes the concept of late and early bindings for the programming languages shown in Table G.1. The relative merits of early and late bindings are shown in Table G.2.

Table G.1 *Early & Late Bindings for various Programming Languages*

Language	SDAI Early Bindings	SDAI Late Bindings
C + +	STEP Part 23	STEP Part 23 ^[19]
C	-	STEP Part 24 ^[20]
IDL	STEP Part 26	STEP Part 26 ^[21]
JAVA	STEP Part 27	STEP Part 27 ^[22]
XML	STEP Part 28	STEP Part 28 ^[23]

Table G.2 *Early Bindings vs Late Bindings*

	Early Bindings	Late Bindings
Compilation result	data structures & specific bindings	schema & generic bindings
Advantages	convenience and speed	independent of schema more than one schema independent of technology
Disadvantages	implementation schema specific	slow schema has to be loaded at runtime

G.4 Example SDAI C programs

The following example programs demonstrate the use of the most common SDAI operations that developers of CIS/2 translators need to know. These are summarized in Table G.3.

Table G.3: *SDAI operations required for CIS/2 translators*

sdaiOpenSession	Open session
sdaiTypeChecking	Activate type checking
sdaiOpenRepositoryBN	Open repository
sdaiCreateModelBN	Create the model
sdaiOpenModelBN	Open Repository Model
sdaiGetEntityExtentBN	Get all instances of an entity
sdaiCreateIterator	Create an Iterator to traverse an aggregation
sdaiBeginning	Go to the beginning of an aggregation
sdaiCreateInstanceBN	Create an instance of an entity
sdaiNext	Get an element of the aggregation
sdaiGetAggrByIterator	Retrieve the instance of the entity
sdaiGetAttrBN	Get an attribute of an instance
sdaiDeleteIterator	Get rid of the iterator
sdaiPutAttrBN	Create attribute
sdaiCloseModel	Close the repository Model
sdaiDeleteModel	Delete the model
sdaiCloseRepository	Close repository
sdaiCloseSession	Close session

G.4.1 Example Program for an Export translator.

/*

This program uses the EDM Toolkit to create a Model called “model” and an instance of an entity “calendar_date”, it then fills the instance with attributes and closes the model, and exports the model to a STEP physical file.

*/

/* include files */

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <memory.h>

#include "sdai.h"

/*This file is generic to all STEP toolkits with SDAI Late C bindings */

```

char IniFile[_MAX_PATH];
#define USR_ERROR      1L
#define SUCCESS        0L
#define FATAL          -1L
enum {false, true} ;

/* Function to print an error message and terminate */
void printErrorAndTerminate(char *message, long errorCode)
{
    printf("\n%s Error: %s",message,edmiGetErrorText(errorCode));
    sdaiCloseSession();
    exit(1);
}

void MainOutput(char *str)
{
    printf("%s", str);
}

void main()
{
    char HeaderName[30] = "Header";
    long          rstat;
    SdaiSession    sessionId;
    SdaiRepository repository;
    SdaiModel       model_key;
    SdaiInstance    date;
    long            nWarnings,nErrors;
    long            realPrecision = 8;
    long            errCode, Rc, occur, ret_type;
    long            cacheSize = 0;
    Rc = 0L;
    occur = 1L;
    ret_type = 0L;

    /* Open an already created database */
    rstat = edmiOpenDatabase("c:\\db\\", "CIMSTEEL", "CIS");
    if (rstat)
        printErrorAndTerminate("\nError when opening the database.",rstat);

```

```

/* Open session */
sessionId = sdaiOpenSession();
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Error Code is %d \n", Rc);
        Rc = USR_ERROR;
        printf("Got to number sdaiOpenSession\n");
    }

#ifdef TRACE
    (void) sdaiDebug(DB_DEFINE, DB_CALLS | DB_ARGS | DB_RETURNS |
                    DB_ERRORS | DB_STDOUT, 0, "");
    (void) sdaiDebug(DB_START, 0, 0, "");
#endif
(void) sdaiUpdateInverse(sdaiOFF);

/* Activate type checking and data value checking in all write EDM/SDAI functions */
(void) sdaiTypeChecking(sdaiON);

/* Open repository */
repository = sdaiOpenRepositoryBN("ModelRepository");
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Cannot open repository. Error Code is %d \n", Rc);
        Rc = USR_ERROR;
    }

/* Create the model */
sdaiCreateModelBN ( repository, "model", "generic_lpm");
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Cannot create the repository Model. Error Code is %d \n", Rc);
        Rc = USR_ERROR;
    }

/* Open Repository Model */
model_key = sdaiOpenModelBN (repository, "model", sdaiRW);
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Cannot open the repository Model. Error Code is %d \n", Rc);
        Rc = USR_ERROR;
    }

```

```

    }

    /*Create an instance of the calendar_date entity*/
    date = sdaiCreateInstanceBN(model_key,"calendar_date");

    /*Create integer attribute */
    sdaiPutAttrBN (date,"day_component", sdaiINTEGER, 10);

    /* Create integer attribute */
    sdaiPutAttrBN (date,"month_component", sdaiINTEGER, 12);

    /* Create integer attribute */
    sdaiPutAttrBN (date,"year_component", sdaiINTEGER, 1996);

    /* Close the repository Model */
    sdaiCloseModel(model_key);
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Cannot Close rep Model. Error Code is %d \n", Rc);
        Rc = USR_ERROR;
    }

    /* Activate the STEP Formatter to produce a new STEP file */
    rstat = edmiWriteStepFile("ModelRepository",
        NULL,
        "model",
        "Write.stp",
        "fmtDia.txt",
        realPrecision,
        0,
        &nWarnings,
        &nErrors,
        &errCode);

    if (rstat) {
        printErrorAndTerminate("\nError when writing the data to a STEP file:", rstat);
    }

    /* Delete the model */
    sdaiDeleteModel(model_key);

```

```

/* Close repository */
sdaiCloseRepository(repository);

/* Close session and terminate */
(void) sdaiCloseSession();

/* Close database */
edmiCloseDatabase("CIS");
}

```

G.4.2 Example Program for an Import translator.

```

/*

This program uses the EDM Toolkit to read a STEP file to a model called "model" and
then opens the model, takes one of the "calendar_date" entity instances within it, and
reads the value of one of its attributes - "day_component".

*/

/* include files */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>
#include "sdai.h"

/* This file is generic to all STEP toolkits with SDAI Late C bindings */

char IniFile[_MAX_PATH];
#define USR_ERROR      1L
#define SUCCESS        0L
#define FATAL          -1L
enum {false, true} ;

/* Function to print an error message and terminate */
void printErrorAndTerminate(char *message, long errorCode)
{
    printf("\n%s Error: %s",message,edmiGetErrorText(errorCode));
    sdaiCloseSession();
    exit(1);
}

void MainOutput(char *str)
{

```

```

        printf("%s", str);
    }
void main()
{
    char HeaderName[30] = "Header";
    long                rstat;
    SdaiSession          sessionId;
    SdaiRepository        repository;
    SdaiModel             model_key;
    SdaiAggr              repAggr;
    SdaiInstance          replInstance;
    SdaiIterator           replterator;
    SdaiInteger            integerValue;
    Long                  Rc, occur, ret_type;
    long                  cacheSize = 0;
    SdaiErrorCode          sdaiError;
    char spLog[255];
                                /* Scanner/parser log filename */
    Rc = 0L;
    occur = 1L;
    ret_type = 0L;

    /* Open an already created database */
    rstat = edmiOpenDatabase("c:\\db\\", "CIMSTEEL", "CIS");
    if (rstat)
        printErrorAndTerminate("\nError when opening the database.",rstat);

    /* Open session */
    sessionId = sdaiOpenSession();
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Error Code is %d \n", Rc);
        Rc = USR_ERROR;
        printf("Got to number sdaiOpenSession\n");
    }

    #ifdef TRACE
    (void) sdaiDebug(DB_DEFINE, DB_CALLS |
        DB_ARGS | DB_RETURNS | DB_ERRORS | DB_STDOUT, 0, "");
    (void) sdaiDebug(DB_START, 0, 0, "");
    #endif
    (void) sdaiUpdateInverse(sdaiOFF);

```



```
/* Activate type checking and data value checking in all write EDM/SDAI functions */
```

```
(void) sdaiTypeChecking(sdaiON);
```

```
/* Open repository */
```

```
repository = sdaiOpenRepositoryBN("ModelRepository");
```

```
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
```

```
    {
```

```
        printf("Error Code is %d \n", Rc);
```

```
        Rc = USR_ERROR;
```

```
        printf("Got to number sdaiOpenRepositoryBN\n");
```

```
    }
```

```
/* Read the STEP file */
```

```
rstat = edmiReadStepFile("a:\\model.stp",
```

```
    NULL,
```

```
    NULL,
```

```
    "ModelRepository",
```

```
    HeaderName,
```

```
    "model",
```

```
    "LPM",
```

```
    cacheSize,
```

```
    SREAD_MAX_CHECKING_LEVEL,
```

```
    &sdaiError);
```

```
if (rstat) {
```

```
    printErrorAndTerminate("\nError when reading the data from a STEP file:", rstat);
```

```
}
```

```
/* Open Repository Model */
```

```
model_key = sdaiOpenModelBN (repository, "model", sdaiRW);
```

```
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
```

```
    {
```

```
        printf("Cannot Open the repository Model. Error Code is %d \n", Rc);
```

```
    }
```

```
    /* Initialise the Aggregator */
```

```
    repAggr = 0;
```

```
/* Get all instances of the entity and put them in an Aggregation type */
```

```
repAggr = sdaiGetEntityExtentBN(model_key, "calendar_date");
```

```

/* Create an Iterator type to traverse the Aggregation type */
replterator = sdaiCreateliterator(repAggr);

/*Go to the beginning */
sdaiBeginning(replterator);

/* Get an element of the aggregation */
sdaiNext(replterator);

/* Retrieve the instance of the entity */
(void) sdaiGetAggrByliterator(replterator, sdaiINSTANCE, &replInstance);

/* Get the day_component of the instance */
if ( sdaiTestAttrBN(replInstance, "day_component") == sdaiTRUE )
    sdaiGetAttrBN ( replInstance, "day_component", sdaiINTEGER, &integerValue );

/*Get rid of the iterator */
sdaiDeleteliterator(replterator);

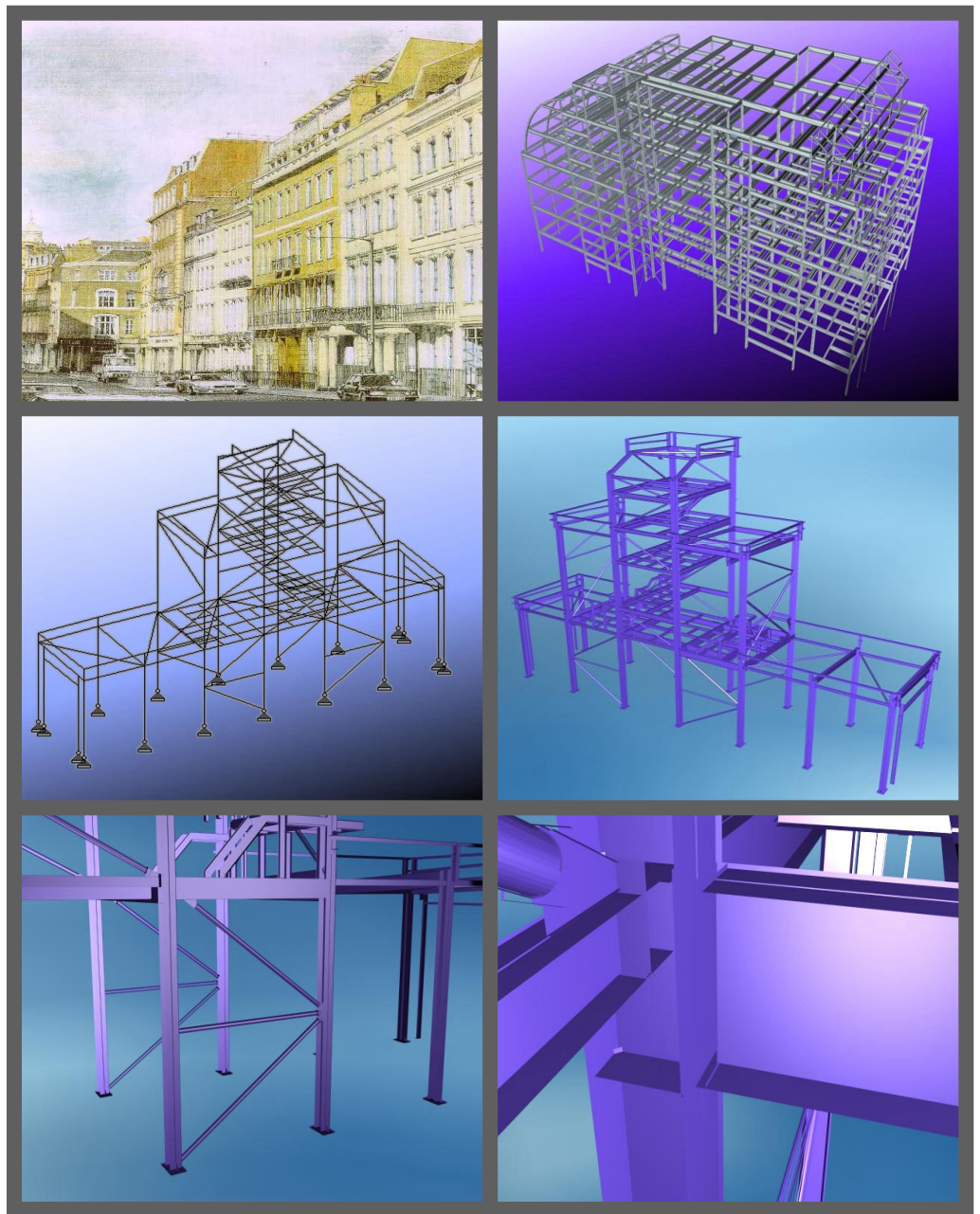
/* Close the repository Model */
sdaiCloseModel(model_key);
    if ((Rc = sdaiErrorQuery()) != SUCCESS)
    {
        printf("Cannot Close rep Model. Error Code is %d \n", Rc);
        Rc = USR_ERROR;
    }

/* Close repository */
sdaiCloseRepository(repository);
strcpy(spLog,"splog");

/* Close session and terminate */
(void) sdaiCloseSession();

/* Close database */
edmiCloseDatabase("CIS");
}

```



The Steel Construction Institute

CIMsteel Integration Standards Release 2: Second Edition

Volume 4 The Logical Product Model (LPM/6)

CIS/2.1



The Steel Construction Institute

The Steel Construction Institute is an independent, technical, member-based organization, dedicated to the development and promotion of the effective use of steel in construction. Founded in 1986, the SCI now has over 600 corporate members in 37 countries. The SCI's research and development activities cover many aspects of steel construction including multi-storey construction, industrial buildings, light gauge steel framing systems, stainless steel, fire engineering, bridge and civil engineering, offshore and hazard engineering, structural analysis systems, environmental engineering and information technology. The results of these projects are fed back to SCI members via a comprehensive package of benefits. Membership is open to all organizations and individuals that are concerned with the use of steel in construction. Members include designers, contractors, suppliers, fabricators, academics and government departments in the United Kingdom, elsewhere in Europe and in countries around the world. The SCI is financed by subscriptions from its members, revenue from research contracts and consultancy services, publication sales and course fees.

A Membership Information Pack is available free on request from:

The Membership and Development Manager, The Steel Construction Institute, Silwood Park, Ascot, Berkshire, SL5 7QN, UK.

Telephone: +44 (0)1344 623345, Fax: +44 (0)1344 622944.



School of Civil Engineering

The School of Civil Engineering in the University of Leeds is the largest in the United Kingdom, and has excellent contacts with the construction industry and professional institutions. Its staff members have a wide range of experience in the practice of engineering, planning, architecture, design, construction and management. The School is committed to Quality Assurance of its teaching. Contacts with industry include an Advisory Committee of senior engineers and architects in practice, an industrial tutor scheme for undergraduates involvement of practising specialists in student's projects, and lectures by eminent engineers, architects and landscape architects. Industry also participates by giving sponsorships, prizes, providing vacation work and welcoming its graduates on completion of their degree. The emphasis on multi-discipline work, design projects, communication skills and teamwork seems to make Leeds graduates particularly useful.

The Computer Aided Engineering (CAE) group is a multi-disciplinary research group concerned with the application of Information Technologies to the overall engineering process. The group gained an international reputation through its innovative work applying the concepts of product modelling to achieve a more efficient information flow in the construction sector.

For details of current research or a prospectus, please contact:

The School Secretary, School of Civil Engineering, University of Leeds, Leeds, LS2 9JT, UK. Telephone: +44 (0)113 343 2248, Fax: +44 (0)113 343 2265.

CIMsteel Integration Standards

Release 2: Second Edition

Volume 4 –

The Logical Product Model (LPM/6)

A J Crowley BEng, PhD

A S Watson BTech, PhD, CEng, MICE

Published by:

The Steel Construction Institute
Silwood Park
Ascot
Berkshire
SL5 7QN

Tel: 01344 623345

Fax: 01344 622944

URL: <http://www.steel-sci.org/>

In association with:

Computer Aided Engineering Group
School of Civil Engineering
The University of Leeds
Leeds
LS2 9JT

Tel: 0113 343 2282

Fax: 0113 343 2265

URL: <http://www.leeds.ac.uk/civil/>

This publication is derived from deliverables of the CIMsteel Project

© 2000, 2003 The University of Leeds

© 2000, 2003 The Steel Construction Institute

The University of Leeds and the Steel Construction Institute wish to acknowledge the valuable contribution from other CIMsteel Collaborators, and from other parties, in the generation of this material.

Apart from any fair dealing for the purposes of research or private study or criticism or review, as permitted under the Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the UK Copyright Licensing Agency, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organisation outside the UK.

Enquiries concerning reproduction outside the terms stated here should be sent to the publishers, The Steel Construction Institute, at the address given on the title page.

Although care has been taken to ensure, to the best of our knowledge, that all data and information contained herein are accurate to the extent that they relate to either matters of fact or accepted practice or matters of opinion at the time of publication, The Steel Construction Institute, The University of Leeds, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data and/or information or any loss or damage arising from or related to their use.

Publications supplied to the Members of the Institute at a discount are not for resale by them.

Publication Number: SCI-P-268

ISBN 1 85942 102 4 (1st Edition)


British Library Cataloguing-in-Publication Data. (1st Edition)

A catalogue record for this book is available from the British Library. (1st Edition)

Extracts of International Standards included in this publication have been reproduced by kind permission of the British Standards Institute. Full copies of these International Standards are available from national standards bodies.

Front cover images have been reproduced by kind permission of Whitby Bird & Partners, Rowen Structures Ltd, Taywood Engineering Ltd, and Philip Quantrill (Structural Engineers) Ltd.

This publication is supported by a companion web site at <http://www.cis2.org/>

 [®] is a Registered Trademark

FOREWORD

2nd Edition

This publication is one of a series of SCI publications that document the 2nd Edition of the 2nd release of the CIMsteel Integration Standards (CIS/2.1). This publication documents version six of the *CIMsteel Logical Product Model (LPM/6)*. The Logical Product Model is an interpretation of the information requirements (described elsewhere), which encapsulates the concepts used in the domain of Structural Engineering in a form that can be implemented in computer systems.

In the 3 years since the first edition of CIS/2 was published, CIS/2 users have raised 103 issues. Most of these have been trivial, but the 25 ‘Major Technical’ issues warranted the development of a second edition of CIS/2. Further, second editions of the STEP Generic Resources^[23, 24, 25, 26] used in LPM/5 were published in 2000; the revised constructs contained therein are incorporated into LPM/6.

In summary, 124 EXPRESS constructs have been added to LPM/6 for the 2nd Edition of CIS/2. A further 112 have been modified. Many of the changes to the Logical Product Model have been made to accommodate the information requirements of MIS systems, including:

- Drawing files
- Cost codes
- CNC files
- Sequences and lots

Other changes were required to represent:

- Design loading
- Cambered assemblies
- Various types of Welds
- Fasteners with position
- Pock marks
- Managed data without data history

This 2nd Edition should be seen as a ‘point release’, rather than a whole ‘new’ release of the CIMsteel Integration Standards. Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

The development of the second edition was coordinated by the CIS/2 International Technical Committee (ITC); the body responsible for improving and maintaining CIS/2 as an open standard. Chaired by Chuck Eastman, the ITC comprises The Steel Construction Institute and Leeds University (the joint custodians of CIS/2), together with The American Institute of Steel Construction (AISC), Georgia Institute of Technology and the National Institute of Standards and Technology (NIST). The successful deployment of CIS/2 technology over the past years has been due to the efforts of software vendors who have developed commercially viable CIS/2 translators.

Consequently, several of these companies are represented on the ITC, including: Bentley, CSC (UK) Ltd, CSI, Design Data, Fabtrol, Intergraph, RAM International, and Tekla.

The authors would like to thank the following for their input to the 2nd Edition:

- Bob Lipman and Kent Reed of NIST
- Chuck Eastman, Joon You, Frank Wang, Donghoon Yang, and Jae Min Lee of Georgia Institute of Technology.

The development of the second edition was sponsored by the American Institute of Steel Construction (AISC).

Previous Editions

The first release of the CIMsteel Integration Standards (CIS/1) and the draft release of CIS/2 were the deliverables of the Pan-European Eureka EU130 CIMsteel Project. Over a ten-year period, this extensive research and development project involved seventy organizations in eight countries; representing a wide cross section of the constructional steelwork industry, including designers, fabricators, software vendors, universities, research and trade organizations. In addition to the contributions of the individual collaborators, the CIMsteel project also received financial support from:

- Forschungsförderungsfonds für die gewerbliche Wirtschaft (FFF) in Austria
- Erhvervsfremme Styrelsen, Industrie-ministeriet in Denmark
- Ministère de l'Industrie, Département Communication et Commerce Extérieur in France
- Istituto Mobiliare Italiano in Italy
- Senter Den Haag in The Netherlands
- Department of Trade and Industry in the United Kingdom

The principal authors of this publication are Andrew Crowley of the SCI and Alastair Watson of the University of Leeds. The development of this publication was funded by the SCI, Corus Group plc (formerly British Steel plc) and the University of Leeds.

The authors would like to thank their (former) colleagues in the Computer Aided Engineering (CAE) Group within the School of Civil Engineering at the University of Leeds for their work on the CIMsteel Project; in particular Steven Bennett, Dimitris Christodoulakis, Paul Hirst, Nashett El-Kaddah, George Kamparis, Gareth Knowles, Peter Riley, Alan Smith, and Mike Ward. The authors would also like to acknowledge the valuable outputs of the CIMsteel 'Overall Design and Analysis Working Group'. This group was led by Richard Greiner of the Technical University of Graz, Austria, and included representatives from Leeds and Nottingham Universities, QSE, CSC, SCI, Ove Arup, Taywood from the UK, Ramboll (Denmark), TDV (Austria), CTICM (France), Sidercad (Italy) and FCSA (Finland).

The CIS are the result of considerable efforts by many persons representing CIMsteel collaborators, associate collaborators and other interested parties. The principal developers of the CIS were Leeds University & SCI (UK), CTICM & LGCH (France), TNO (Netherlands), Ramboll (Denmark), Italsiel & Sidercad (Italy). Inputs from beyond Europe, particularly from the USA and Japan are acknowledged. The issues raised by the international review team of AP230 have been invaluable in the development of CIS/2.

AISC endorsement of CIS/2

CIS/2 has been endorsed by the American Institute of Steel Construction (AISC) as the standard for the electronic exchange of structural steel project information for the North American structural steel design and construction industry^[13].

During 1998, the Electronic Data Interchange (EDI) Review Team of the AISC evaluated data transfer standards with a view to adopting one. On December 7th 1998, their recommendation of CIS/2 was approved by the AISC Board of Directors as part of the AISC Business Plan for Standardizing the Electronic Exchange of Structural Steel Project Information. Phase I of the AISC Business Plan (now completed) included the public endorsement of CIS/2 as the standard for the electronic exchange of structural steel project information for the entire U.S. structural steel design and construction industry, as well as the recommendation that it be adopted as an international standard. Phase II of the AISC Business Plan includes several activities that will promote and support CIS/2 and its implementation.

AISC, headquartered in Chicago, is a not-for-profit organization established in 1921 to serve the structural steel industry in the United States. The organization's mission is to promote the use of structural steel through research activities, market development, education, codes and specifications, technical assistance, quality certification and standardization. AISC maintains the specification for the design of structural steel framing in the U.S. It has a long tradition of more than 75 years of providing assurance and service to the steel construction industry by providing reliable information.

For further details, please contact:

Director of Information Technology,
American Institute of Steel Construction,
One East Wacker Drive, Suite 3100, Chicago, IL 60601-2001, USA.
Telephone: +1 312 670 5413, Fax: +1 312 670 5403

CONTENTS

	Page No.
FOREWORD	III
2 nd Edition	iii
Previous Editions	iv
AISC endorsement of CIS/2	v
CONTENTS	VI
List of Figures	viii
SUMMARY	XII
1 SCHEMA DECLARATION	1
1.1 Object identifier	1
2 SCHEMA INTERFACE DECLARATIONS	2
2.1 Schema Interface declarations for types	2
2.2 Schema Interface declarations for entities	4
2.3 Schema Interface declarations for functions	10
2.4 Schema Interface declarations for rules	12
2.5 Schema Interface declarations for constants	12
3 LPM/6 DECOMPOSITION	13
3.1 Decomposition concepts and assumptions	13
3.2 Decomposition type definitions	14
3.3 Decomposition entity definitions	22
4 LPM/6 PROJECT DEFINITION	62
4.1 Project Definition concepts and assumptions	62
4.2 Project Definition type definitions	62
4.3 Project Definition entity definitions	63
5 LPM/6 STRUCTURAL MODELLING	102
5.1 Structural Modelling concepts and assumptions	102
5.2 Structural Modelling type definitions	103
5.3 Structural Modelling entity definitions	108
6 LPM/6 LOADING	173
6.1 Loading concepts and assumptions	173
6.2 Loading type definitions	173
6.3 Loading entity definitions	178
7 LPM/6 STRUCTURAL RESPONSE	222
7.1 Structural Response concepts and assumptions	222
7.2 Structural Response type definitions	222
7.3 Structural Response entity definitions	223

8	LPM/6 PARTS	260
8.1	Parts concepts and assumptions	260
8.2	Parts type definitions	260
8.3	Parts entity definitions	261
9	LPM/6 FEATURES	286
9.1	Features concepts and assumptions	286
9.2	Features type definitions	286
9.3	Features entity definitions	288
10	LPM/6 JOINT SYSTEMS	330
10.1	Joint Systems concepts and assumptions	330
10.2	Joint Systems type definitions	330
10.3	Joint Systems entity definitions	342
11	LPM/6 MATERIALS	401
11.1	Materials concepts and assumptions	401
11.2	Materials type definitions	401
11.3	Materials entity definitions	402
12	LPM/6 LOCATION	422
12.1	Location concepts and assumptions	422
12.2	Location type definitions	425
12.3	Location entity definitions	426
13	LPM/6 GROUPING	463
13.1	Grouping concepts and assumptions	463
13.2	Grouping type definitions	463
13.3	Grouping entity definitions	469
14	LPM/6 GEOMETRY	485
14.1	Geometry concepts and assumptions	485
14.2	Geometry type definitions	489
14.3	Geometry entity definitions	494
15	LPM/6 UNITS & MEASURES	539
15.1	Units & Measures concepts and assumptions	539
15.2	Units & Measures type definitions	543
15.3	Units & Measures entity definitions	549
16	LPM/6 ITEM REFERENCES	581
16.1	Item References concepts and assumptions	581
16.2	Item References type definitions	581
16.3	Item References entity definitions	582
16.4	Item References function definitions	608
17	LPM/6 PRODUCT DEFINITION	613

17.1	Product Definition concepts and assumptions	613
17.2	Product Definition type definitions	613
17.3	Product Definition entity definitions	614
18	LPM/6 PROCESS DEFINITION	618
18.1	Process Definition concepts and assumptions	618
18.2	Process Definition type definitions	618
18.3	Process Definition entity definitions	628
19	LPM/6 DATA MANAGEMENT	652
19.1	Data Management concepts and assumptions	652
19.2	Data Management type definitions	653
19.3	Data Management entity definitions	656
19.4	Data Management function definitions	672
20	REFERENCES	674
APPENDIX A	LPM/6 EXPRESS SCHEMA (LONG FORM)	677
APPENDIX B	LPM/6 EXPRESS-G DIAGRAMS	899
APPENDIX C	NEW CONSTRUCTS IN 2 ND EDITION OF CIS/2	981
APPENDIX D	MODIFIED CONSTRUCTS IN 2 ND EDITION OF CIS/2	985
INDEX		989

List of Figures

Figure 3.1	<i>Example composition of a steel framed building</i>	13
Figure 3.2	<i>Examples of internal structural connections</i>	28
Figure 3.3	<i>Example of a structural frame</i>	29
Figure 3.4	<i>Defining an absolute camber</i>	37
Figure 4.1	<i>Defining a grid_level at the origin of the grid (zero altitude)</i>	73
Figure 4.2	<i>Defining a gridline (at the origin of the grid: $x = y = 0$)</i>	79
Figure 5.1	<i>An example of a simple 2D analysis model</i>	102
Figure 5.2	<i>The sign conventions used in LPM/6</i>	103
Figure 5.3	<i>Types of simple volume elements</i>	105
Figure 5.4	<i>Examples of boundary conditions</i>	119
Figure 5.5	<i>The moment-rotation relationship associated with of non-linear spring boundary condition</i>	127
Figure 5.6	<i>Example of an element_curve_complex</i>	133
Figure 5.7	<i>The relationship between a beta angle and an orientation vector for a beam element (Example 1)</i>	134
Figure 5.8	<i>The relationship between a beta angle and an orientation vector for a column element (Example 1)</i>	135
Figure 5.9	<i>Defining simple curve elements</i>	137

Figure 5.10	<i>The relationship between a beta angle and an orientation vector for a beam element (Example 2)</i>	138
Figure 5.11	<i>The relationship between a beta angle and an orientation vector for a column element (Example 2)</i>	138
Figure 5.12	<i>Defining the orientation of a beam element</i>	139
Figure 5.13	<i>Defining the orientation of a column element (Example 1)</i>	140
Figure 5.14	<i>Defining the orientation of a column element (Example 2)</i>	141
Figure 5.15	<i>Defining the orientation of a column element (Example 3)</i>	141
Figure 5.16	<i>Defining element eccentricity for linear elements</i>	142
Figure 5.17	<i>The effect of element_eccentricity on the definition of the locating longitudinal axis of a beam element</i>	143
Figure 5.18	<i>The effect of element_eccentricity on the definition of the locating longitudinal axis of a column element</i>	144
Figure 5.19	<i>Defining element_eccentricity for planar elements</i>	145
Figure 5.20	<i>Defining element_eccentricity for cubic elements</i>	146
Figure 5.21	<i>Defining surface elements</i>	154
Figure 5.22	<i>Examples of element_surface_profiled</i>	157
Figure 5.23	<i>Defining an element_sheet_profiled</i>	158
Figure 5.24	<i>Defining volume elements</i>	160
Figure 5.25	<i>Tri-linear approximation of a non-linear spring release</i>	171
Figure 6.1	<i>Defining a loading combination</i>	213
Figure 7.1	<i>Examples of structural response models</i>	222
Figure 7.2	<i>Analysis results for a surface element (stresses)</i>	230
Figure 7.3	<i>Analysis results for a surface element (tractions)</i>	232
Figure 7.4	<i>Analysis results for a volume element (stress tensor)</i>	233
Figure 8.1	<i>An example of a part_prismatic_complex</i>	268
Figure 8.2	<i>Defining a part with a varying depth (Example 1)</i>	270
Figure 8.3	<i>Defining a part with a varying depth (Example 2)</i>	271
Figure 8.4	<i>Defining a part with a varying depth (Example 3)</i>	271
Figure 8.5	<i>An example of a doubly tapered prismatic part</i>	272
Figure 8.6	<i>Defining an absolute camber</i>	276
Figure 8.7	<i>Example of part_prismatic_simple_castellated</i>	278
Figure 8.8	<i>Dimensions for castellated prismatic parts</i>	280
Figure 8.9	<i>Examples of part_prismatic_simple_curved</i>	280
Figure 8.10	<i>Examples of part_sheet_profiled</i>	285
Figure 9.1	<i>Definition of 'left', 'right', 'top', 'bottom', 'start' and 'end'</i>	288
Figure 9.2	<i>An example of a cutting plane applied to a prismatic part</i>	290
Figure 9.3	<i>Example of a straight edge chamfer applied to a part</i>	291
Figure 9.4	<i>Examples of types of edge chamfer</i>	291
Figure 9.5	<i>Definition of a feature_edge_chamfer_fillet</i>	292
Figure 9.6	<i>Example of a 'rounding' and a 'fillet' applied to an edge</i>	293
Figure 9.7	<i>Example of a feature_edge_chamfer_straight</i>	294
Figure 9.8	<i>Example of a combination a 'straight chamfer' and a 'fillet'</i>	295
Figure 9.9	<i>Example of a feature_surface_simple applied to a sheet part</i>	300
Figure 9.10	<i>Examples of right and left-handed double threads</i>	302

Figure 9.11	<i>An example of a feature_volume_curved</i>	306
Figure 9.12	<i>Defining a circular hole</i>	308
Figure 9.13	<i>Applying an edge chamfer to a hole</i>	309
Figure 9.14	<i>The resulting 'countersunk hole'</i>	309
Figure 9.15	<i>Defining a rectangular hole</i>	311
Figure 9.16	<i>Defining a slotted hole</i>	312
Figure 9.17	<i>Defining a curved slotted hole</i>	314
Figure 9.18	<i>Defining a chamfer (prismatic volume feature)</i>	317
Figure 9.19	<i>A chamfer applied to the start face of a part</i>	317
Figure 9.20	<i>A chamfer applied to the end face of a part</i>	318
Figure 9.21	<i>Defining a 'flange chamfer'</i>	319
Figure 9.22	<i>Defining a 'flange notch'</i>	320
Figure 9.23	<i>Defining a notch (prismatic volume feature)</i>	322
Figure 9.24	<i>Applying a notch to a part</i>	322
Figure 9.25	<i>Combining a notch and a skewed end</i>	324
Figure 9.26	<i>Applying a skewed end to a part (Example 1)</i>	325
Figure 9.27	<i>Applying a doubly skewed end to a part</i>	326
Figure 9.28	<i>Defining the skew angles for a doubly skewed end</i>	326
Figure 10.1:	<i>Alignment for intermittent welds</i>	331
Figure 10.2:	<i>Example of weld_surface_shape</i>	340
Figure 10.3:	<i>Example of a fastener_mechanism showing origin</i>	347
Figure 10.4:	<i>Fastener_mechanism with repositioned origin</i>	348
Figure 10.5	<i>An example of a 'countersunk fastener'</i>	354
Figure 10.6	<i>An example of a 'curved fastener'</i>	355
Figure 10.7	<i>An example of a 'self-drilling screw'</i>	365
Figure 10.8	<i>An example of a 'tapered screw'</i>	366
Figure 10.9	<i>Example fastener_simple_stud</i>	368
Figure 10.10	<i>An example of a 'tapered washer'</i>	373
Figure 10.11	<i>Example of a joint_system_mechanical</i>	379
Figure 10.12	<i>Example of a joint_system_welded_linear</i>	381
Figure 10.13	<i>Example of a 'plug weld'</i>	383
Figure 10.14	<i>Examples of different weld types and penetrations</i>	386
Figure 10.15:	<i>Examples of weld_mechanism_fillet</i>	388
Figure 10.16:	<i>Example of a weld_mechanism_fillet_intermittent</i>	392
Figure 10.17:	<i>Defining the attributes of weld_mechanism_groove_beveled</i>	395
Figure 11.1	<i>Examples of bi-linear and tri-linear stress-strain characteristics</i>	407
Figure 11.2	<i>Example of the secant modulus</i>	408
Figure 12.1	<i>The hierarchical system of location within LPM/6</i>	422
Figure 12.2	<i>Located parts and located joint systems</i>	423
Figure 12.3	<i>Locating parts within an assembly</i>	424
Figure 12.4	<i>General transformation matrix for 2D coordinate systems</i>	429
Figure 12.5	<i>Nested coordinate systems</i>	431
Figure 12.6	<i>General transformation matrix for rotation about X-axis only</i>	433
Figure 12.7	<i>General transformation matrix for rotation about Y-axis only</i>	433
Figure 12.8	<i>General transformation matrix for rotation about Z-axis only</i>	434

Figure 12.9	<i>A cylindrical point defined in a cylindrical coordinate system</i>	437
Figure 12.10	<i>A spherical point located in a spherical coordinate system</i>	439
Figure 12.11	<i>An example of a located_feature_joint_dependent</i>	451
Figure 12.12	<i>Located parts forming an assembly</i>	456
Figure 13.1	<i>An example of a cyclic group relationship</i>	478
Figure 14.1	<i>Relating topology to geometry</i>	488
Figure 14.2	<i>Defining cardinal points 1-19 on a generic section profile</i>	489
Figure 14.3	<i>Defining cardinal points 21 to 49 on a flanged section profile</i>	492
Figure 14.4	<i>Defining cardinal points 51-59 on a box or hollow section profile</i>	492
Figure 14.5	<i>Defining cardinal points for a section_profile_compound</i>	493
Figure 14.6	<i>Effect of mirroring on the cardinal point positions</i>	496
Figure 14.7	<i>Definition of a section_profile_angle</i>	497
Figure 14.8	<i>Defining centreline section profiles</i>	499
Figure 14.9	<i>Definition of a section_profile_channel</i>	500
Figure 14.10	<i>Definition of a section_profile_circle_hollow</i>	504
Figure 14.11	<i>Examples of section_profile_compound</i>	507
Figure 14.12	<i>Relocating the components of a section_profile_compound</i>	508
Figure 14.13	<i>Examples of section_profile_derived</i>	510
Figure 14.14	<i>Example of an edge defined section profile</i>	512
Figure 14.15	<i>Definition of a section_profile_i_type</i>	513
Figure 14.16	<i>Definition of section_profile_i_type_asymmetric</i>	516
Figure 14.17	<i>Definition of a section_profile_i_type_rail</i>	518
Figure 14.18	<i>Example of a section_profile_rectangle_hollow</i>	522
Figure 14.19	<i>Definition of a section_profile_t_type</i>	525
Figure 14.20	<i>Definition of the origin_offset</i>	529
Figure 14.21	<i>Example of the principal axes used in section_properties_asymmetric</i>	535
Figure 16.1	<i>An example of a cyclic document relationship</i>	585
Figure 18.1	<i>An example of a heating and cooling process</i>	644
Figure 18.2	<i>An example of a quenching process</i>	645

SUMMARY

This publication is one of a series of SCI publications that documents the 2nd Edition of the 2nd release of the CIMsteel Integration Standards (CIS/2.1). This publication documents version six of the *CIMsteel Logical Product Model (LPM/6)*. The Logical Product Model is an interpretation of the *Information Requirements*, which encapsulates the concepts used in the domain of Structural Engineering in a form that can be implemented in computer systems.

The main body of this publication documents the Short Form of LPM/6. It provides illustrated textual descriptions of the entities, with their attributes and formal propositions (rules & constraints), together with references to STEP resources. In general, it does not repeat the definitions of those STEP entities that have been used from the Generic Resources. However, brief explanations of some of the generic constructs are provided to aid their understanding and use with LPM/6. The definitions for the entities that have specifically defined for LPM/6 are presented within one of 17 subject areas. A brief guide to the EXPRESS Language is discussed in the *Implementation Guide*. The philosophy behind LPM/6 and how it was developed are discussed elsewhere.

Appendix A presents the complete LPM/6 that underlies CIS/2 as an EXPRESS schema in its 'long form'; that is, as one single schema with all the inter-schema references resolved. The Long Form contains no definitions or explanations, merely an EXPRESS long form listing of the CIS/2.1 implementation schema.

Appendix B provides a graphical presentation (in EXPRESS-G) of the CIS/2.1 implementation schema (i.e. the Long Form of LPM/6). The notation used in the EXPRESS-G diagrams is discussed in the *Implementation Guide*.

Other documents in this series include:

- *Volume 1 – Overview*^[6]
- *Volume 2 - Implementation Guide*^[7]
- *Volume 3 - The Information Requirements*^[8]
- *Volume 5 - Conformance Requirements*^[9]
- *Volume 6 - Worked Examples*^[10]

1 SCHEMA DECLARATION

The following EXPRESS declaration begins the computer-sensible version of the CIMsteel Logical Product Model version 6 (LPM/6). To ensure that it provides a mechanism for unambiguous data sharing, LPM/6 has been assigned the name 'STRUCTURAL_FRAME_SCHEMA'. This schema shall be implemented in CIS/2-compatible computer systems to support file exchange and data sharing via Release 2 of the CIMsteel Integration Standards (CIS/2).

```
*)
SCHEMA STRUCTURAL_FRAME_SCHEMA;
(*
```

This schema may be implemented in a number of forms, from 'Basic CIS Translators' that support physical file exchange through to advanced database management systems. The details of the different methods of implementation are discussed in the *Implementation Guide*^[7]; while the *Conformance Requirements*^[9] specify what software vendors need to do to make their applications 'CIS/2-compatible'. It is strongly recommended that these publications be consulted before attempting any implementation of this schema.

1.1 Object identifier

In order to provide an unambiguous identification of this schema amongst other schemas and other information objects in an open information system, CIS/2 employs a technique similar to that defined in STEP Part 1. For LPM/6, CIS/2 defines the object identifier as:

```
(*
{cimsteel logical product model version (6) object (1) structural-frame-schema(1)}
*)
```

Notes:

This is not part of the EXPRESS schema specification, merely a way to identify the version of the schema defined here.

Its use and encoding in a STEP Part 21 file are explained in the *Implementation Guide* and the *Conformance Requirements*.

2 SCHEMA INTERFACE DECLARATIONS

The following EXPRESS declarations identify the necessary external references that will be resolved in the long form of LPM/6. All of the external references declared below are described in detail in the STEP ‘integrated resources’ (Part 40 series of ISO 10303^[23, 24, 25, 26, 27]). The short form schema makes use of the schema interfacing capabilities of EXPRESS so that CIS/2-compatible applications are able to use the EXPRESS constructs contained in the STEP ‘integrated resources’.

2.1 Schema Interface declarations for types

*)

REFERENCE FROM basic_attribute_schema (-- STEP Part 41

description_attribute_select,
id_attribute_select,
name_attribute_select,
role_select);

REFERENCE FROM date_time_schema (-- STEP Part 41

ahead_or_behind,
day_in_month_number,
hour_in_day,
minute_in_hour,
month_in_year_number,
second_in_minute,
year_number);

REFERENCE FROM measure_schema (-- STEP Part 41

area_measure,
context_dependent_measure,
count_measure,
descriptive_measure,
length_measure,
mass_measure,
measure_value,
numeric_measure,
parameter_value,
plane_angle_measure,
positive_length_measure,
positive_plane_angle_measure,
positive_ratio_measure,
ratio_measure,
si_prefix,
si_unit_name,
solid_angle_measure,
thermodynamic_temperature_measure,
time_measure,
unit,

volume_measure);

REFERENCE FROM support_resource_schema (-- STEP Part 41
 identifier,
 label,
 text);

REFERENCE FROM geometric_model_schema (-- STEP Part 42
 boolean_operand,
 boolean_operator,
 csg_primitive,
 csg_select,
 geometric_set_select,
 surface_model,
 wireframe_model);

REFERENCE FROM geometry_schema (-- STEP Part 42
 axis2_placement,
 b_spline_curve_form,
 b_spline_surface_form,
 curve_on_surface,
 dimension_count,
 knot_type,
 pcurve_or_surface,
 preferred_surface_curve_representation,
 transition_code,
 trimming_preference,
 trimming_select,
 vector_or_direction);

REFERENCE FROM topology_schema (-- STEP Part 42
 list_of_reversible_topology_item,
 reversible_topology,
 reversible_topology_item,
 set_of_reversible_topology_item,
 shell);

REFERENCE FROM representation_schema (-- STEP Part 43
 founded_item_select,
 transformation);

REFERENCE FROM qualified_measure_schema (-- STEP Part 45
 value_qualifier);
(*

2.2 Schema Interface declarations for entities

*)

USE FROM action_schema (-- STEP Part 41

action,
action_directive,
action_method,
directed_action,
executed_action,
versioned_action_request);

USE FROM approval_schema (-- STEP Part 41

approval,
approval_status);

USE FROM basic_attribute_schema (-- STEP Part 41

description_attribute,
id_attribute,
name_attribute,
object_role,
role_association);

USE FROM certification_schema (-- STEP Part 41

certification,
certification_type);

USE FROM contract_schema (-- STEP Part 41

contract,
contract_type);

USE FROM date_time_schema (-- STEP Part 41

calendar_date
coordinated_universal_time_offset,
date,
date_and_time,
local_time);

USE FROM document_schema (-- STEP Part 41

document,
document_relationship,
document_representation_type,
document_type,
document_usage_constraint,
document_with_class);

USE FROM group_schema (-- STEP Part 41

group,
group_relationship);

```
USE FROM management_resources_schema ( -- STEP Part 41
    group_assignment);
```

```
USE FROM measure_schema ( -- STEP Part 41
    area_measure_with_unit,
    area_unit,
    context_dependent_unit,
    conversion_based_unit,
    derived_unit,
    derived_unit_element,
    dimensional_exponents,
    global_unit_assigned_context,
    length_measure_with_unit,
    length_unit,
    mass_measure_with_unit,
    mass_unit,
    measure_with_unit,
    named_unit,
    plane_angle_measure_with_unit,
    plane_angle_unit,
    ratio_measure_with_unit,
    ratio_unit,
    si_unit,
    solid_angle_measure_with_unit,
    solid_angle_unit,
    thermodynamic_temperature_measure_with_unit,
    thermodynamic_temperature_unit,
    time_measure_with_unit,
    time_unit,
    volume_measure_with_unit,
    volume_unit);
```

```
USE FROM person_organization_schema ( -- STEP Part 41
    address,
    organization,
    organization_relationship,
    organizational_address,
    person,
    person_and_organization,
    person_and_organization_role,
    personal_address);
```

```
USE FROM product_property_representation_schema ( -- STEP Part 41
    shape_representation);
```

```
USE FROM geometric_model_schema ( -- STEP Part 42
```

block,
boolean_result,
box_domain,
boxed_half_space,
brep_2d,
brep_with_voids,
circular_area,
convex_hexahedron,
csg_solid,
cyclide_segment_solid,
eccentric_cone,
edge_based_wireframe_model,
ellipsoid,
elliptic_area,
extruded_area_solid,
extruded_face_solid,
face_based_surface_model,
faceted_brep,
faceted_primitive,
geometric_curve_set,
geometric_set,
geometric_set_replica,
half_space_2d,
half_space_solid,
manifold_solid_brep,
polygonal_area,
primitive_2d,
rectangular_area,
rectangular_pyramid,
revolved_area_solid,
revolved_face_solid,
right_angular_wedge,
right_circular_cone,
right_circular_cylinder,
shell_based_surface_model,
shell_based_wireframe_model,
solid_model,
solid_replica,
sphere,
surface_curve_swept_area_solid,
surface_curve_swept_face_solid,
swept_area_solid,
swept_face_solid,
tetrahedron,
torus,
trimmed_volume);

USE FROM geometry_schema (-- STEP Part 42

axis1_placement,
axis2_placement_2d,
axis2_placement_3d,
b_spline_curve,
b_spline_curve_with_knots,
b_spline_surface,
b_spline_surface_with_knots,
b_spline_volume,
b_spline_volume_with_knots,
bezier_curve,
bezier_surface,
bezier_volume,
block_volume,
boundary_curve,
bounded_curve,
bounded_pcurve,
bounded_surface,
bounded_surface_curve,
cartesian_point,
cartesian_transformation_operator,
cartesian_transformation_operator_2d,
cartesian_transformation_operator_3d,
circle,
clothoid,
composite_curve,
composite_curve_on_surface,
composite_curve_segment,
conic,
conical_surface,
curve,
curve_bounded_surface,
curve_replica,
cylindrical_point,
cylindrical_surface,
cylindrical_volume,
degenerate_pcurve,
degenerate_toroidal_surface,
direction,
eccentric_conical_volume,
elementary_surface,
ellipse,
ellipsoid_volume,
evaluated_degenerate_pcurve,
fixed_reference_swept_surface,
geometric_representation_context,
geometric_representation_item,

hexahedron_volume,
hyperbola,
intersection_curve,
line,
offset_curve_2d,
offset_curve_3d,
offset_surface,
oriented_surface,
outer_boundary_curve,
parabola,
pcurve,
placement,
plane,
point,
point_in_volume,
point_on_curve,
point_on_surface,
point_replica,
polar_point,
polyline,
pyramid_volume,
quasi_uniform_curve,
quasi_uniform_surface,
quasi_uniform_volume,
rational_b_spline_curve,
rational_b_spline_surface,
rational_b_spline_volume,
rectangular_composite_surface,
rectangular_trimmed_surface,
reparametrised_composite_curve_segment,
seam_curve,
spherical_point,
spherical_surface,
spherical_volume,
surface,
surface_curve,
surface_curve_swept_surface,
surface_of_linear_extrusion,
surface_of_revolution,
surface_patch,
surface_replica,
swept_surface,
tetrahedron_volume,
toroidal_surface,
toroidal_volume,
trimmed_curve,
uniform_curve,


```
uniform_surface,
uniform_volume,
vector,
volume,
wedge_volume);
```

USE FROM topology_schema (-- STEP Part 42

```
closed_shell,
connected_edge_set,
connected_face_set,
edge,
edge_curve,
edge_loop,
face,
face_bound,
face_outer_bound,
face_surface,
loop,
open_path,
open_shell,
oriented_closed_shell,
oriented_edge,
oriented_face,
oriented_open_shell,
oriented_path,
path,
poly_loop,
subedge,
subface,
topological_representation_item,
vertex,
vertex_loop,
vertex_point,
vertex_shell,
wire_shell);
```

USE FROM representation_schema (-- STEP Part 43

```
definitional_representation,
founded_item,
functionally_defined_transformation,
global_uncertainty_assigned_context,
item_defined_transformation,
mapped_item,
parametric_representation_context,
representation,
representation_context,
representation_item,
```

```
representation_map,
representation_relationship,
representation_relationship_with_transformation,
uncertainty_measure_with_unit);
```

```
USE FROM qualified_measure_schema ( -- STEP Part 45
    measure_qualification,
    precision_qualifier,
    qualitative_uncertainty,
    standard_uncertainty,
    type_qualifier,
    uncertainty_qualifier);
```

```
USE FROM building_design_schema ( -- STEP Part 225
    truncated_pyramid);
```

(*

2.3 Schema Interface declarations for functions

*)

```
REFERENCE FROM basic_attribute_schema ( -- STEP Part 41
    get_description_value,
    get_id_value,
    get_name_value,
    get_role);
```

```
REFERENCE FROM date_time_schema ( -- STEP Part 41
    leap_year,
    valid_calendar_date,
    valid_time);
```

```
REFERENCE FROM document_schema ( -- STEP Part 41
    acyclic_document_relationship);
```

```
REFERENCE FROM group_schema ( -- STEP Part 41
    acyclic_group_relationship);
```

```
REFERENCE FROM measure_schema ( -- STEP Part 41
    derive_dimensional_exponents,
    dimensions_for_si_unit,
    valid_units);
```

```
REFERENCE FROM person_organization_schema ( -- STEP Part 41
    acyclic_organization_relationship);
```

```
REFERENCE FROM support_resource_schema ( -- STEP Part 41
    bag_to_set);
```

REFERENCE FROM geometric_model_schema (-- STEP Part 42

acyclic_set_replica,
acyclic_solid_replica,
build_transformed_set,
constraints_geometry_shell_based_surface_model,
constraints_geometry_shell_based_wireframe_model);

REFERENCE FROM geometry_schema (-- STEP Part 42

above_plane,
acyclic_curve_replica,
acyclic_point_replica,
acyclic_surface_replica,
associated_surface,
base_axis,
build_2axes,
build_axes,
constraints_composite_curve_on_surface,
constraints_param_b_spline,
constraints_rectangular_composite_surface,
cross_product,
curve_weights_positive,
dimension_of,
dot_product,
first_proj_axis,
get_basis_surface,
list_to_array,
make_array_of_array,
make_array_of_array_of_array,
normalise,
orthogonal_complement,
same_side,
scalar_times_vector,
second_proj_axis,
surface_weights_positive,
vector_difference,
volume_weights_positive);

REFERENCE FROM topology_schema (-- STEP Part 42

boolean_choose,
closed_shell_reversed,
conditional_reverse,
edge_reversed,
face_bound_reversed,
face_reversed,
list_face_loops,
list_loop_edges,

```
list_of_topology_reversed,
list_to_set,
mixed_loop_type_set,
open_shell_reversed,
path_head_to_tail,
path_reversed,
set_of_topology_reversed,
shell_reversed,
topology_reversed);
```

```
REFERENCE FROM representation_schema ( -- STEP Part 43
  acyclic_mapped_representation,
  item_in_context,
  using_items,
  using_representations,
  valid_measure_value);
(*)
```

2.4 Schema Interface declarations for rules

```
*)
REFERENCE FROM geometry_schema ( -- STEP Part 42
  compatible_dimension);
(*)
```

2.5 Schema Interface declarations for constants

```
*)
REFERENCE FROM geometry_schema ( -- STEP Part 42
  dummy_gri);

REFERENCE FROM topology_schema ( -- STEP Part 42
  dummy_tri);
(*)
```

3 LPM/6 DECOMPOSITION

3.1 Decomposition concepts and assumptions

3.1.1 Conceptual overview

The Decomposition subject area includes both the decomposition and composition mechanisms, both of which are fundamental concepts of the LPM. Decomposition describes how the Structure is broken down for the design of structural frames, structural members and structural connections. Its manufacturing counterpart - Composition - describes how the physical parts are joined together to form the complete structure.

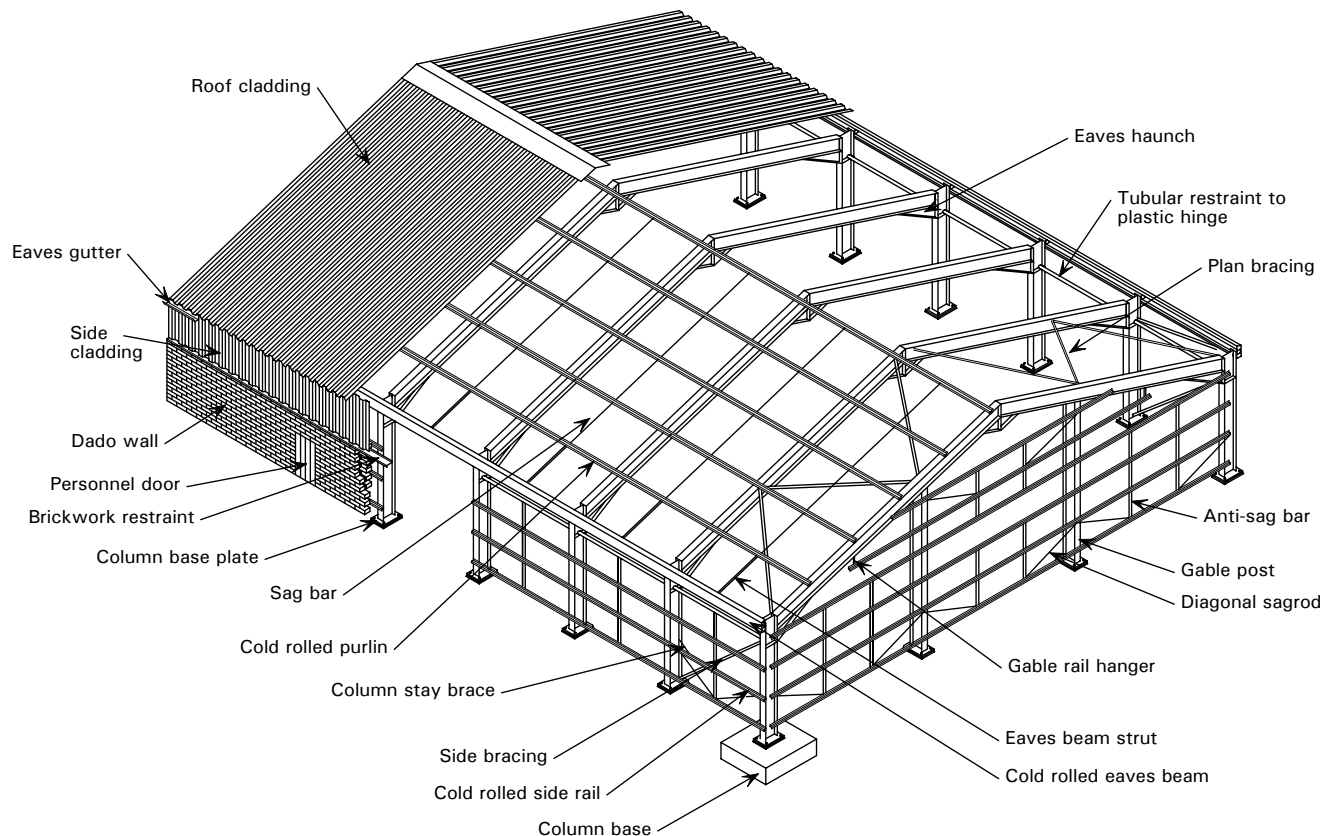


Figure 3.1 *Example composition of a steel framed building*

For the purposes of member and connection design, the structure is represented using design assemblies. These may be decomposed into other (sub) design assemblies. The design assemblies are categorized by their functional role as either structural members, structural connections, or structural frames. The structural members are categorized by their dimensionality as either linear (one-dimensional), planar (two-dimensional) or cubic (three-dimensional). The shape of an assembly may be represented using explicit geometry. Design assemblies ultimately comprise design parts and design joint systems. A design part is a conceptual representation of a basic piece of steel. A design joint system is a conceptual representation of a basic joint system. Design results can be described for structural members and structural connections, and are expressed in terms of the elastic or plastic resistance of the global member or connection, or local components.

For the purposes of detailing, production preparation, and manufacturing, the structure is represented using manufacturing assemblies. These may be composed of other (sub)

manufacturing assemblies. Manufacturing assemblies ultimately comprise located parts and located joint systems. A located part is a representation of a basic physical piece of steel. A located joint system is a representation of a physical joint system (e.g. a bolt group or a weld).

The composition mechanism is hierarchical in nature due to the constraint that an `assembly_manufacturing_child` can belong to only one 'parent' `assembly_manufacturing`. The decomposition mechanism is more complicated, and effectively allows many-to-many relationships to exist between instances of `assembly_design`. An additional ANDOR SUPERTYPE construct in assembly allows it to be given an explicit shape through the reference to `shape_representation`.

3.2 Decomposition type definitions

The following types have been modified in this subject area for the 2nd edition of CIS/2:

- `member_linear_type`
- `member_planar_type`

The following types have been added in this subject area for the 2nd edition of CIS/2:

- `member_beam_role`
- `member_beam_type`
- `member_brace_type`
- `member_cable_type`
- `member_column_type`
- `member_plate_type`
- `member_slab_type`
- `member_wall_type`

3.2.1 buckling_direction

Type definition:

Classifies the direction of the buckling of the member as follows:

- `x_dir` - when considering buckling in the member `x_direction`,
- `y_dir` - when considering buckling in the member `y` direction (for beam with an I section profile - buckling about the minor axis),
- `z_dir` - when considering buckling in the member `z` direction (for beam with an I section profile - buckling about the major axis)

EXPRESS specification:

```
*)
TYPE buckling_direction
= ENUMERATION OF
  (x_dir, y_dir, z_dir);
END_TYPE;
(*
```

Notes:

Unchanged from CIS/1. Unchanged in 2nd Edition

3.2.2 complexity_level**Type definition:**

Classifies the assembly to be of low, medium, or high complexity based on the predicted difficulty of manufacture. The definition of complexity is based upon the CIMsteel document *Design for Manufacturing Guidelines*^[3]. This may be used to aid initial cost estimates. (Used in entity assembly).

EXPRESS specification:

```
*)
TYPE complexity_level
= ENUMERATION OF
    (low, medium, high);
END_TYPE;
(*
```

Notes:

Unchanged from CIS/1. Unchanged in 2nd Edition.

3.2.3 connection_type**Type definition:**

Classifies the type of connection assumed for analysis purposes. Connections can be pinned, semi-rigid, or rigid. Rigid and semi-rigid connections can also be full strength or partial strength. This enumeration may be used by some applications as an initializer.

EXPRESS specification:

```
*)
TYPE connection_type
= ENUMERATION OF
    (pinned,
     semi_rigid_full_str,
     semi_rigid_partial_str,
     rigid_full_str,
     rigid_partial_str);
END_TYPE;
(*
```

Notes:

Known as ansys_connctn_type in CIS/1. Unchanged in 2nd Edition.

3.2.4 frame_continuity**Type definition:**

Classifies the structural frame as either simple, continuous, or semi-continuous, based on the ability of the frame's connections to transfer bending. This enumeration may be used by some applications as an initializer. (Used in entity assembly_design_structural_frame.)

EXPRESS specification:

*)
 TYPE frame_continuity
 = ENUMERATION OF
 (simple, continuous, semi_continuous);
 END_TYPE;
 (*

Notes:

Known as frm_continuity in CIS/1. Unchanged in 2nd Edition.

3.2.5 frame_type

Type definition:

Classifies the type of model assumed for analysis purposes, based on the dimensionality of the frame (i.e. 2D or 3D) and the end fixity of the frame's elements (i.e. pinned or rigid). This enumeration may be used by some applications as an initializer. (Used in entities assembly_design_structural_frame and analysis_model).

EXPRESS specification:

*)
 TYPE frame_type
 = ENUMERATION OF
 (space_frame, space_truss, plane_frame, plane_truss, grillage, undefined);
 END_TYPE;
 (*

Notes:

Known as analys_model_type in CIS/1. Unchanged in 2nd Edition.

3.2.6 member_beam_role

Type definition:

Classifies the beam (1 dimensional member) by its primary role and possible use.

EXPRESS specification:

*)
 TYPE member_beam_role
 = ENUMERATION OF
 (edge_beam,
 eaves_beam,
 gantry_girder,
 joist,
 lintel,
 portal_rafter,
 purlin,
 rafter,
 ring_beam,
 side_rail,
 waling_beam);
 END_TYPE;

(*

Notes:

New for CIS/2 2nd Edition.

3.2.7 member_beam_type

Type definition:

Classifies the beam (1 dimensional member) by its primary form.

EXPRESS specification:

*)

TYPE member_beam_type

= ENUMERATION OF

(box_girder,
fish_bellied_beam,
haunched_beam,
plate_girder,
stub_girder,
tapered_beam);

END_TYPE;

(*

Notes:

New for CIS/2 2nd Edition.

3.2.8 member_brace_type

Type definition:

Classifies the brace (1 dimensional member) by its primary form and possible use in other types of structural member.

EXPRESS specification:

*)

TYPE member_brace_type

= ENUMERATION OF

(cross_brace,
diagonal_brace,
horizontal_brace,
knee_brace,
lateral_brace,
longitudinal_brace,
plan_brace,
raker,
sway_brace,
vertical_brace);

END_TYPE;

(*

Notes:

New for CIS/2 2nd Edition.

3.2.9 member_cable_type

Type definition:

Classifies the cable (1 dimensional member) by its primary form and possible use in other types of structural member.

EXPRESS specification:

```
*)
TYPE member_cable_type
= ENUMERATION OF
    (stay,
     suspension_cable,
     suspension_chain);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

3.2.10 member_class

Type definition:

Classifies the level in the structural hierarchy of the structural member. This hierarchy is based on load transfer: e.g., secondary members are supported by primary members.

EXPRESS specification:

```
*)
TYPE member_class
= ENUMERATION OF
    (primary_member,
     secondary_member,
     tertiary_member,
     undefined_class);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

3.2.11 member_column_type

Type definition:

Classifies the column (1 dimensional member) by its primary form and possible use in other types of structural member.

EXPRESS specification:

```
*)
TYPE member_column_type
= ENUMERATION OF
    (battened_column,
     box_column,
     compound_strut,
```

```
portal_column);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

3.2.12 member_cubic_type**Type definition:**

Classifies the cubic member (i.e. 3 dimensional) by its primary form and possible decomposition into other types of structural member. For example, a floor (a cubic member) is likely to be made up of (planar) slabs and (linear) beams. Similarly, a stair would require (planar) stair elements.

EXPRESS specification:

```
*)
TYPE member_cubic_type
= ENUMERATION OF
    (floor, stair, ramp, structural_core, structural_shell, undefined);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

3.2.13 member_linear_type**Type definition:**

Classifies the linear member (i.e. 1 dimensional) by its primary form and possible use in other types of structural assembly. For example, beams, columns and braces.

EXPRESS specification:

```
*)
TYPE member_linear_type
= ENUMERATION OF
    (beam,
    column,
    truss_element,
    brace,
    spring_element,
    cable,
    pipe,
    wire,
    tie,
    undefined,
    arch,
    beam_column);
END_TYPE;
(*
```

Notes:

New for CIS/2. Modified for 2nd Edition.

3.2.14 member_planar_type

Type definition:

Classifies the planar member (i.e. 2 dimensional) by its primary form and possible use in other types of structural assembly. For example, a slab (a planar member) is likely to be used as part of a (cubic) floor. Similarly, a stair element is likely to be used as part of a (cubic) stair.

EXPRESS specification:

*)

TYPE member_planar_type

= ENUMERATION OF

(wall,
slab,
stair_element,
ramp_element,
undefined,
plate);

END_TYPE;

(*

Notes:

New for CIS/2. Modified in 2nd Edition.

3.2.15 member_plate_type

Type definition:

Classifies the plate (2 dimensional member) by its primary form and its possible use in other types of structural member.

EXPRESS specification:

*)

TYPE member_plate_type

= ENUMERATION OF

(bearing_plate,
diaphragm,
flange,
web);

END_TYPE;

(*

Notes:

New for CIS/2 2nd Edition.

3.2.16 member_role

Type definition:

Declares the principal role of the structural member, i.e. whether the member acts mainly in compression, tension, bending, or a combination of actions.

EXPRESS specification:

```
*)
TYPE member_role
= ENUMERATION OF
    (compression_member,
    tension_member,
    bending_member,
    combined_member,
    undefined_role);
END_TYPE;
(*
```

Notes:

Known as member_role in CIS/1 - enumeration items changed. Unchanged in 2nd Edition.

3.2.17 member_slab_type

Type definition:

Classifies the slab (2 dimensional member) by its form.

EXPRESS specification:

```
*)
TYPE member_slab_type
= ENUMERATION OF
    (flat_slab,
    ribbed_slab,
    solid_slab,
    trough_slab,
    voided_slab,
    waffle_slab);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

3.2.18 member_wall_type

Type definition:

Classifies the wall (2 dimensional member) by its primary form and possible use in other types of structural member.

EXPRESS specification:

```
*)
TYPE member_wall_type
= ENUMERATION OF
    (load_bearing_wall,
    retaining_wall,
    shear_wall);
END_TYPE;
```

(*

Notes:

New for CIS/2 2nd Edition.

3.2.19 shop_or_site

Type definition:

Classifies an assembly or joint system by its place of assembly; i.e. in the fabrication shop or on the construction site. The place of assembly may be declared as ‘undefined’.

EXPRESS specification:

*)

TYPE shop_or_site

= ENUMERATION OF

(shop_process, site_process, undefined);

END_TYPE;

(*

Notes:

Known as place_of_asbly in CIS/1 - enumeration items changed. Unchanged in 2nd Edition.

3.3 Decomposition entity definitions

The following entities have been modified for the 2nd Edition of CIS/2:

- assembly
- assembly_design_structural_member_linear
- assembly_design_structural_member_planar

The following entities have been added for the 2nd Edition of CIS/2:

- assembly_design_structural_member_linear_beam
- assembly_design_structural_member_linear_brace
- assembly_design_structural_member_linear_cable
- assembly_design_structural_member_linear_campered
- assembly_design_structural_member_linear_campered_absolute
- assembly_design_structural_member_linear_campered_relative
- assembly_design_structural_member_linear_column
- assembly_design_structural_member_planar_plate
- assembly_design_structural_member_planar_slab
- assembly_design_structural_member_planar_wall
- assembly_with_bounding_box

3.3.1 assembly

Entity definition:

A representation of the structure for design or manufacturing purposes. An abstract SUPERTYPE of either an `assembly_design` or an `assembly_manufacturing`. The assembly may be related to other assemblies using the `assembly_relationship`.

As a SUBTYPE of `structural_frame_product`, an assembly inherits the three attributes `item_number`, `item_name`, and `item_description` (from `structural_frame_item`). It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that an assembly may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`) – this allows the appropriate national standard or code of practice used in the definition of the assembly to be described in detail
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

An assembly may also be related to another assembly (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`. However, the more constrained entity `assembly_relationship` should be used in preference.

As a SUBTYPE of `structural_frame_product`, an assembly may also be assigned a material (via the entity `structural_frame_product_with_material`).

EXPRESS specification:

```

*)
ENTITY assembly
ABSTRACT SUPERTYPE OF (ONEOF
    (assembly_design,
     assembly_manufacturing) ANDOR
     assembly_with_shape)
SUBTYPE OF (structural_frame_product);
    assembly_sequence_number : OPTIONAL INTEGER;
    complexity : OPTIONAL complexity_level;
DERIVE
    uses : SET [0:?] OF located_assembly := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.LOCATED_ASSEMBLY.
        DESCRIPTIVE_ASSEMBLY'));
END_ENTITY;
(*

```

Attribute definitions:

assembly_sequence_number

Specifies the assembly's position (as an integer value) in the sequence in which the assembly is assembled. The lower the sequence number, the higher the priority that assembly has it when it is used to make up a higher level assembly. For example, an

assembly with a sequence number of 10 will be assembled (i.e. added to an assembly) before one with a sequence number of 11.

complexity

Declares the classification of the assembly to be of low, medium, or high complexity based on the predicted difficulty of manufacture. The definitions of complexity is based upon the CIMsteel document 'Design for Manufacturing Guidelines'. This may be used to aid initial cost estimates.

uses

Derives the set of zero, one or more instances of `located_assembly` that reference this instance of `assembly` via its attribute `descriptive_assembly`. The `located_assembly` instance adds a location and orientation to the specification provided by the `assembly` instance.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

Known as ASSEMBLY in CIS/1. This entity has been considerably expanded and its attributes have been modified.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition (Derive clause added).

3.3.2 assembly_design

Entity definition:

An abstract conceptual subset of the structure created for design purposes. Top down design is supported since the (eventual) definition of the `assembly_design` is in terms of the `design_parts` they require, and this can be deferred. An `assembly_design` decomposition may be viewed as an organizational tool, rather than as an actual physical decomposition. An `assembly_design` can also be a structural frame (`assembly_design_structural_frame`), a structural member (`assembly_design_structural_member`), or a structural connection (`assembly_design_structural_connection`), recognizing that all of these types of `assembly_design` are made up from a number of parts and joints or other assemblies. An `assembly_design` can also be an `assembly_design_child` (due to the ANDOR SUPERTYPE declaration). This facilitates the decomposition.

EXPRESS specification:

*)

ENTITY `assembly_design`

SUPERTYPE OF (ONEOF

(`assembly_design_structural_frame`,
`assembly_design_structural_member`,


```

        assembly_design_structural_connection) ANDOR
        assembly_design_child)
SUBTYPE OF (assembly);
    designed : BOOLEAN;
    checked : BOOLEAN;
    roles : SET [0:?] OF functional_role;
    governing_criteria : SET [0:?] OF design_criterion;
END_ENTITY;
(*)

```

Attribute definitions:

designed

Declares whether this instance of *assembly_design* has been designed (TRUE) or not (FALSE) - to give the status of the *assembly_design*.

checked

Declares whether this instance of *assembly_design* has been checked (TRUE) or not (FALSE) - to give the status of the *assembly_design*.

roles

Declares the set of instances of *functional_role* that may be associated with this instance of *assembly_design*. Collectively, these instances of *functional_role* describe what the assembly is designed to contribute to the structure.

governing_criteria

Declares the set of instances of *design_criterion* that may be associated with this instance of *assembly_design*. Collectively, these instances of *design_criterion* describe the regulations and specifications that govern the design of the assembly. Such design criteria may be documented in national standards or codes of practice.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

Known as *design_assembly* in CIS/1 (attributes modified). The decomposition mechanism has been placed in a separate entity - *assembly_design_child*.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

The attributes *roles* and *governing_criteria* are defined as SETs whose lower bounds (minimum number of items in the aggregation) are zero. In other words, empty sets are allowed. This allows the application to leave the aggregation empty. Nevertheless, since the attributes are not optional, null values are not allowed. As a minimum, the attributes should be populated as empty sets.

3.3.3 assembly_design_child

Entity definition:

An assembly_design_child is an assembly_design at any 'node' of the decomposition network (a product of the design process). This decomposition hierarchy defines how the initial assembly_design at the top of the hierarchy (which corresponds to the complete structure) is progressively broken down by the designer into other instances of assembly_design. If necessary an assembly_design may be decomposed into two or more sub assembly_design.

EXPRESS specification:

```
*)
ENTITY assembly_design_child
SUBTYPE OF (assembly_design);
    parent_assemblies : SET [1:?] OF assembly_design;
WHERE
    WRA27 : SIZEOF(QUERY(assembly <* parent_assemblies | assembly := (SELF))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

parent_assemblies

Declares the set of assemblies which it forms part of (and is therefore a child of). Because the assembly_design is an abstract concept, the child may have many parents. In the simplest case - a hierarchical design decomposition - the child will only have one parent. Because its SUPERTYPE assembly_design has been declared as an ANDOR, the assembly_design_child can also be a structural member, a structural frame or a structural connection.

Formal propositions:

WRA27

The size of the set of instances of assembly_design referenced by the attribute parent_assemblies that are instance equal to this assembly_design_child shall equal zero. In other words, an assembly_design_child cannot be its own parent. It can be a parent of another instance of assembly_design_child. That is, instance recursion is prohibited while entity recursion is permitted.

Informal propositions:

The SET data type of the attribute parent_assemblies means that all of the instances referenced shall be distinct and separate.

Because the SET data type of the attribute parent_assemblies has a lower bound of 1, there must be at least one instance of assembly_design associated with each instance of assembly_design_child. Thus, an instance of assembly_design_child cannot exist without an associated instance of assembly_design or one of its SUBTYPES; i.e. assembly_design_child is said to be existence dependent on assembly_design.

Because the INVERSE has not been declared in the entity assembly_design, the default bounds spec [0:?] applies. Thus, an instance of assembly_design can be associated with any number of instances of assembly_design.

Notes:

New for CIS/2. The decomposition mechanism has now been removed from the DESIGN_ASSEMBLY (in CIS/1) and placed here.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.4 assembly_design_structural_connection**Entity definition:**

A type of assembly_design. Allows for the explicit description (or categorization) of an assembly_design as a structural connection, for analysis, or connection design purposes. There will be times when the structural connections must be designed as collections of parts and joints. In other words, the connection must be viewed as a whole for design purposes. The connection may be categorized through its SUBTYPEs as either internal or external.

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_connection
SUPERTYPE OF (ONEOF
    (assembly_design_structural_connection_internal,
    assembly_design_structural_connection_external))
SUBTYPE OF (assembly_design);
    struc_connection_type : OPTIONAL connection_type;
END_ENTITY;
(*

```

Attribute definitions:**struc_connection_type**

Declares the classification of the rigidity of the connection assumed for connection design purposes, in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used; i.e. pinned, rigid, full or partial strength, etc. This attribute states a basic assumption used during analysis and design, and may be used as an ‘initializer’ by some applications. As the attribute is OPTIONAL, it need not be populated for every instance.

Notes:

Known as STRUC_CONNECTION in CIS/1.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.5 assembly_design_structural_connection_external**Entity definition:**

A type of assembly_design_structural_connection. Declares the structural connection to be made between a single structural member and an external component. This entity could be used to define a column base connection.

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_connection_external
SUBTYPE OF (assembly_design_structural_connection);
    connected_member : assembly_design_structural_member;
END_ENTITY;
(*)

```

Attribute definitions:*connected_member*

Declares the association between the connection and the member (i.e. declares the instance of `assembly_design_structural_member` which is associated with this instance of `assembly_design_structural_connection_external`).

Notes:

New for CIS/2.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.6 assembly_design_structural_connection_internal**Entity definition:**

A type of `assembly_design_structural_connection`. Declares the structural connection to be made between several structural members. This entity could be used to define a column to beam connection, or a beam to beam connection, or the more complicated connections shown in Figure 3.2.

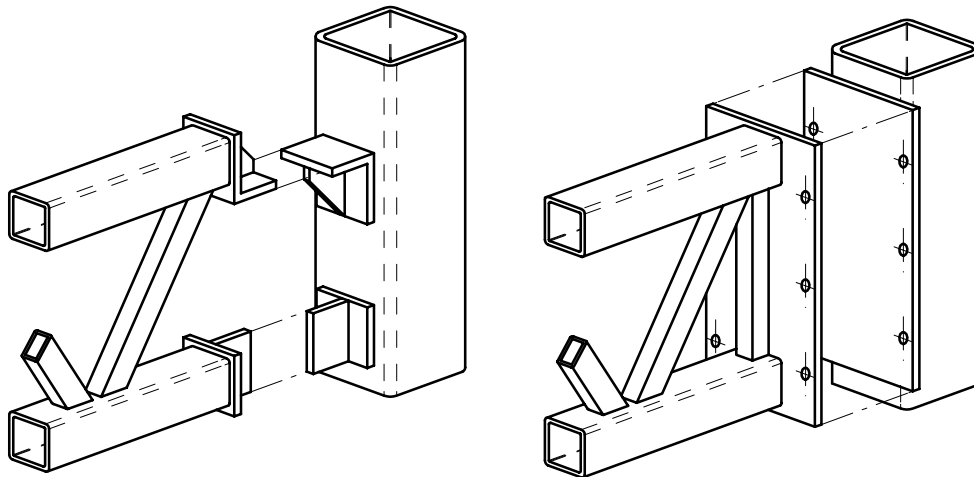


Figure 3.2 Examples of internal structural connections

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_connection_internal
SUBTYPE OF (assembly_design_structural_connection);
    connected_members : SET [2:?] OF assembly_design_structural_member;
END_ENTITY;
(*)

```

Attribute definitions:***connected_members***

Declares the association between the connection and the set of members (i.e. declares the instances of `assembly_design_structural_member` which are associated with this instance of `assembly_design_structural_connection_internal`.)

Formal propositions:

There must be at least two instances of `assembly_design_structural_member` associated with each and every instance of `assembly_design_structural_connection_internal`. There could be more than two.

Notes:

New for CIS/2.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.7 assembly_design_structural_frame**Entity definition:**

A type of `assembly_design`. Allows for the explicit description (or categorization) of an `assembly_design` as a frame, for analysis, or member design purposes. There will be times when the structural members must be designed as part of a frame, and thus, the frame must be considered as a whole for structural design purposes.

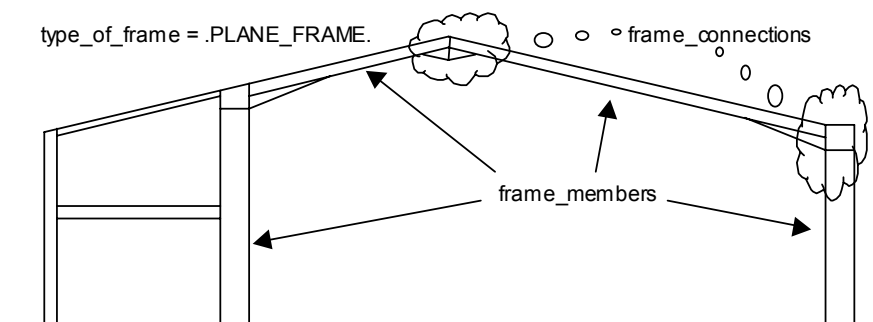


Figure 3.3 *Example of a structural frame*

EXPRESS specification:

*)

```
ENTITY assembly_design_structural_frame
SUBTYPE OF (assembly_design);
  type_of_frame : frame_type;
  continuity : OPTIONAL frame_continuity;
  sway_frame : OPTIONAL BOOLEAN;
  braced_frame : OPTIONAL BOOLEAN;
  bracing_frame : OPTIONAL BOOLEAN;
  frame_members : SET [0:?] OF assembly_design_structural_member;
  frame_connections : SET [0:?] OF assembly_design_structural_connection;
END_ENTITY
(*
```

Attribute definitions:***type_of_frame***

Declares the classification of the structural frame based on its dimensionality (i.e. 2D or 3D) and the end fixity of the frame's elements (i.e. pinned or rigid), in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used. This attribute states a basic assumption used during analysis and design and may be used by some applications as an initializer.

continuity

Declares the classification of the continuity of the structural frame assumed for frame design purposes (i.e. simple, continuous, semi-continuous), in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used; i.e. simple, continuous, semi-continuous. This attribute states a basic assumption used during analysis and design, and may be used as an initializer by some applications.

sway_frame

Declares the classification of the nature of the structural frame assumed for frame design purposes, in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used; i.e. sway frame (TRUE) or no-sway frame (FALSE). A sway frame must be analysed considering those arrangements of the imposed and wind loads which are critical for failure in sway mode (in addition to its analysis as a non-sway frame); i.e. additional sway moments induced by horizontal displacement need to be designed for. A sway frame (usually) provides its own lateral stability by means of rigid joints within the frame itself, and must be designed with this in mind. This attribute states a basic assumption used during analysis and design.

braced_frame

Declares the classification of the stability of the structural frame assumed for frame design purposes, in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used; i.e. braced frame (TRUE) or unbraced frame (FALSE). A braced frame provides its own lateral stability by means of a bracing system independent of the frame itself (e.g. cross members, shear walls and a floor diaphragm), and must be designed with this in mind. This attribute states a basic assumption used during analysis and design.

bracing_frame

Declares the classification of the stability of the structural frame assumed for frame design purposes, in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used; i.e. bracing frame (TRUE), or non-bracing frame (FALSE). A bracing frame provides lateral stability to other frames, and must be designed with this in mind. This attribute states a basic assumption used during analysis and design.

frame_members

Declares the set of structural members associated with (and assumed to be belonging to) this instance of structural frame. If the set is empty (as the lower bound is zero) there are no structural members declared as being associated with this structural frame.

frame_connections

Declares the set of structural connections associated with (and assumed to be belonging to) this instance of structural frame. If the set is empty (as the lower bound is zero) there are no structural connections declared as being associated with this structural frame.

Informal propositions:

The only attribute that has to be populated for every instance of `assembly_design_structural_frame` is the `type_of_frame`, as all the other attributes are effectively optional. However, it is expected that frames would normally be associated with their members and connections.

Notes:

Known a `STRUC_FRAME` in CIS/1.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.8 assembly_design_structural_member**Entity definition:**

A type of `assembly_design`. Allows for the explicit description (or categorization) of an `assembly_design` as a structural member, for analysis or member design purposes. There will be times when the structural member must be designed as a collection of parts and joints. In other words, the member must be viewed as a whole for design purposes.

EXPRESS specification:

*)

ENTITY `assembly_design_structural_member`

SUPERTYPE OF (ONEOF

(`assembly_design_structural_member_cubic`,
`assembly_design_structural_member_linear`,
`assembly_design_structural_member_planar`))

SUBTYPE OF (`assembly_design`);

`key_member` : OPTIONAL BOOLEAN;

`structural_member_use` : `member_role`;

`structural_member_class` : `member_class`;

DERIVE

`restraints` : SET [0:?] OF `restraint` := `bag_to_set` (USEDIN(SELF,
`'STRUCTURAL_FRAME_SCHEMA.RESTRAINT.RESTRAINED_MBR'`));

`effective_lengths` : SET [0:?] OF `effective_buckling_length` := `bag_to_set` (USEDIN(SELF,
`'STRUCTURAL_FRAME_SCHEMA.EFFECTIVE_BUCKLING_LENGTH.APPLICABLE_MEMBER'`));

END_ENTITY;

(*

Attribute definitions:**key_member**

Declares the classification of the type of the structural member assumed for design purposes, in accordance with the appropriate standard (e.g. EC3 in Europe) and appropriate to the analysis method used; i.e. key member (TRUE), or non-key member (FALSE). For example, in Europe, key elements or members are considered to be single structural members which support a floor area or roof area of more than 70m² or 15% of the area of the storey, or any other structural component providing lateral restraint vital to the stability of such a member. Key elements must be designed to withstand the dead, imposed and accidental loads in an accidental situation. This attribute states a basic assumption used during analysis and design.

structural_member_use

Declares the primary role of the member; i.e. whether its main function is in compression, tension, bending, or a combination of actions.

structural_member_class

Declares the classification of the level in the structural hierarchy of the structural member. This hierarchy is based on load transfer: e.g. secondary members are supported by primary members.

restraints

Derives the set of zero, one or more instances of restraint that reference this instance of assembly_design_structural_member.

effective_lengths

Derives the set of zero, one or more instances of effective_buckling_length that reference this instance of assembly_design_structural_member.

Notes:

Known as STRUC_MEMBER in CIS/1.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition (Derive clause added).

3.3.9 assembly_design_structural_member_cubic**Entity definition:**

A type of assembly_design_structural_member. Allows for the explicit description (or categorization) of a three dimensional structural member, for analysis, or member design purposes. Shells, floors, stairs and lift shafts may be considered as cubic structural members. This may be used as part of the design hierarchy. That is, a cubic member may be made up from planar and linear members. For example, a floor may be made up from slabs and beams.

EXPRESS specification:

*)

ENTITY assembly_design_structural_member_cubic

SUBTYPE OF (assembly_design_structural_member);

cubic_member_type : member_cubic_type;

cubic_member_components : SET [0:?] OF assembly_design_structural_member;

WHERE

WRA28 : SIZEOF(QUERY(member <* cubic_member_components | member :=: (SELF))
)= 0;

END_ENTITY;

(*

Attribute definitions:*cubic_member_type*

Declares the classification of the cubic member (i.e. 3 dimensional) by its primary form and possible decomposition into other types of structural member. For example, floors and stair are both considered to be 3 dimensional structural members. A (cubic) floor is likely to be made up of (planar) slabs and (linear) beams. Similarly, a (cubic) stair would require (planar) stair elements.

cubic_member_components

Declares the set of structural members associated with this instance of cubic member. If the set is empty there are no structural members associated with this instance.

Formal propositions:*WRA28*

An instance of *assembly_design_structural_member_cubic* may not be a component of itself. It can, however, be a component of another instance of *assembly_design_structural_member_cubic*.

Notes:

New for CIS/2.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.10 assembly_design_structural_member_linear**Entity definition:**

A type of *assembly_design_structural_member*. Allows for the explicit description (or categorization) of a one dimensional structural member, for analysis, or member design purposes. Beams, columns, braces and ties may be considered as linear structural members. This may be used as part of the design hierarchy. That is, a linear member may be part of a planar or cubic member. For example, beams may be part of a floor structure. The *assembly_design_structural_member_linear* may be further specialized through its subtypes – as a beam, brace, or column.

EXPRESS specification:

*)

```
ENTITY assembly_design_structural_member_linear
SUPERTYPE OF (ONEOF (
    assembly_design_structural_member_linear_beam,
    assembly_design_structural_member_linear_brace,
    assembly_design_structural_member_linear_cable,
    assembly_design_structural_member_linear_column) ANDOR
    assembly_design_structural_member_linear_campered)
SUBTYPE OF (assembly_design_structural_member);
    linear_member_type : member_linear_type;
END_ENTITY;
(*
```

Attribute definitions:*linear_member_type*

Declares the classification of the linear member (i.e. 1 dimensional) by its primary form and possible use in other types of structural assembly. For example, beams and columns are considered to be 1 dimensional structural members. A frame is likely to be made up of (linear) beams, columns and braces.

Informal propositions:

The linear structural member is at the lowest level in the member design hierarchy; i.e. it cannot decompose into anything of lower dimensionality. It can, however, be a parent of

another `assembly_design` through its involvement in the `assembly_design_child` via the ANDOR SUPERTYPE. Linear structural members will ultimately be made up from design parts and design joint systems. Linear structural members may be further specialised through the subtypes.

Notes:

New for CIS/2.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition. (Subtypes added.)

3.3.11 `assembly_design_structural_member_linear_beam`

Entity definition:

A type of `assembly_design_structural_member_linear`, allowing the explicit description (or categorization) of a one dimensional structural member, for analysis, or member design purposes as a beam. As part of a design hierarchy, beams may be part of a floor structure. Beams are characterised as horizontal members (or members slightly inclined to the horizontal) acting primarily in bending.

EXPRESS specification:

*)

ENTITY `assembly_design_structural_member_linear_beam`

SUBTYPE OF (`assembly_design_structural_member_linear`);

`beam_type` : SET [0:?] OF `member_beam_type`;

`beam_role` : SET [0:?] OF `member_beam_role`;

`unrestrained_beam` : LOGICAL;

`deep_beam` : LOGICAL;

WHERE

WRA35 : SELF\`assembly_design_structural_member_linear.linear_member_type` = BEAM;

WRA42 : SIZEOF(SELF\`assembly_design_structural_member.restraints`) > 1;

WRA43 : SIZEOF(SELF\`assembly_design_structural_member.effective_lengths`) > 0;

END_ENTITY;

(*

Attribute definitions:

`beam_type`

Declares the classification of the beam by its primary form. A beam may have zero, one, or many types; e.g. a `haunched_beam` and a `plate_girder`. (See definition of the TYPE `member_beam_type`.)

`beam_role`

Declares the classification of the beam by its primary role or purpose. A beam may have zero, one, or many roles; e.g. a `ring_beam` and a `waling_beam`. (See definition of the TYPE `member_beam_type`.)

`unrestrained_beam`

Declares whether the beam is unrestrained (TRUE) or restrained (FALSE). Where the restrained condition of the beam is unknown, or has not yet been considered, this attribute should be assigned a value of UNKNOWN. It does not necessary follow that a beam is restrained simply because the `assembly_design_structural_member_linear_beam`

is referenced by several instances of restraint. The definition of when a beam is considered ‘restrained’ is dependent on the type of the restraints provided and the type of bending considered. This is usually defined in the code of practice used for the beam design.

deep_beam

Declares whether the beam is considered to be a ‘deep beam’ (TRUE) for design purposes or not (FALSE). Where this design parameter is unknown, or has not yet been considered, this attribute should be assigned a value of UNKNOWN. When a beam is considered to be ‘deep’, it will not behave as a simple beam and normal bending theory does not apply; other aspects of the beam’s behaviour need to be considered during the design process. The definition of when a beam is considered ‘deep’ is dependent on the code of practice used for the beam design.

Formal propositions:

WRA35

The attribute `linear_member_type` inherited from the supertype `assembly_design_structural_member_linear` shall be assigned the value ‘BEAM’.

WRA42

The size of the set of instances of restraint derived by the attribute restraints (inherited from the supertype `assembly_design_structural_member`) shall be greater than 1. In other words, a beam must have at least two restraints. Even if one end of the beam is unrestrained (or even unsupported), this condition must be declared by an instance of restraint.

WRA43

The size of the set of instances of `effective_buckling_length` derived by the attribute `effective_lengths` (inherited from the supertype `assembly_design_structural_member`) shall be greater than 0. In other words, a beam must have at least one effective length defined.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.12 assembly_design_structural_member_linear_brace

Entity definition:

A type of `assembly_design_structural_member_linear`, allowing the explicit description (or categorization) of a one dimensional structural member, for analysis, or member design purposes as a brace. As part of a design hierarchy, braces may be part of a frame. Braces are characterised as being secondary members (usually inclined or rotated relative to building grid) acting primarily in axial tension or compression.

EXPRESS specification:

*)

```
ENTITY assembly_design_structural_member_linear_brace
SUBTYPE OF (assembly_design_structural_member_linear);
    brace_type : member_brace_type;
```

WHERE

```

    WRA36 : SELF\assembly_design_structural_member_linear.linear_member_type =
        'BRACE';
END_ENTITY;
(*)

```

Attribute definitions:

brace_type

Declares the classification of the brace by its primary form and possible use in other types of structural assembly. (See definition of the TYPE member_brace_type.)

Formal propositions:

WRA36

The attribute linear_member_type inherited from the supertype assembly_design_structural_member_linear shall be assigned the value 'BRACE'.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.13 assembly_design_structural_member_linear_cable

Entity definition:

A type of assembly_design_structural_member_linear, allowing the explicit description (or categorization) of a one dimensional structural member, for analysis, or member design purposes as a cable. Cables are characterised as being flexible tension members.

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_member_linear_cable
SUBTYPE OF (assembly_design_structural_member_linear);
    cable_type : member_cable_type;
WHERE
    WRA37 : SELF\assembly_design_structural_member_linear.linear_member_type =
        'CABLE';
END_ENTITY;
(*)

```

Attribute definitions:

cable_type

Declares the classification of the cable by its primary form and possible use in other types of structural assembly. (See definition of the TYPE member_cable_type.)

Formal propositions:

WRA37

The attribute linear_member_type inherited from the supertype assembly_design_structural_member_linear shall be assigned the value 'CABLE'.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.14 assembly_design_structural_member_linear_cambered

Entity definition:

A type of `assembly_design_structural_member_linear`. Allows for the explicit description (or categorization) of a structural member whose longitudinal axis is intended to be distorted from the nominal axis during the manufacturing process by being displaced in one or two directions.

The magnitude of the distortion from the nominal axis is assumed to vary linearly from zero at the ends of the assembly to the maximum value given by the attributes of the SUBTYPE.

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_member_linear_cambered
  SUPERTYPE OF (assembly_design_structural_member_linear_cambered_absolute
    ANDOR assembly_design_structural_member_linear_cambered_relative)
  SUBTYPE OF (assembly_design_structural_member_linear);
    camber_description : OPTIONAL text;
END_ENTITY;
(*

```

Attribute definitions:

camber_description

An optional text description of the type of camber applied to the assembly.

Notes:

New for CIS/2 2nd Edition.

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.15 assembly_design_structural_member_linear_cambered_absolute

Entity definition:

A type of `assembly_design_structural_member_linear_cambered` where the offset is provided by two instances of `length_measure_with_unit`.

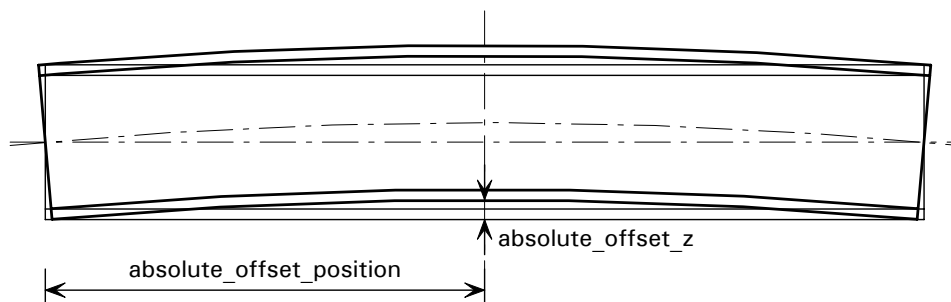


Figure 3.4 *Defining an absolute camber*

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_member_linear_cambered_absolute
  SUBTYPE OF (assembly_design_structural_member_linear_cambered);
    absolute_offset_position : positive_length_measure_with_unit;

```

```

    absolute_offset_y : length_measure_with_unit;
    absolute_offset_z : length_measure_with_unit;
END_ENTITY;
(*)

```

Attribute definitions:

absolute_offset_position

Declares the instance of `positive_length_measure_with_unit` that is associated with the `assembly_design_structural_member_linear_campered` to provide the numerical value with an appropriate unit of the linear dimension to the point where the camber offset is measured. The position is measured along the locating axis of the assembly from its origin in the positive x-direction.

absolute_offset_y

Declares the first instance of `length_measure_with_unit` that is associated with the `assembly_design_structural_member_linear_campered` to provide the numerical value with an appropriate unit of the linear dimension of offset in the positive y-direction.

absolute_offset_z

Declares the second instance of `length_measure_with_unit` that may be associated with the `assembly_design_structural_member_linear_campered` to provide the numerical value with an appropriate unit of the linear dimension of offset in the second direction. The offset is measured perpendicular to the longitudinal axis in the positive z-direction.

Informal propositions:

Where the camber is specified in both relative and absolute terms, this entity is instanced with its sibling SUBTYPE `assembly_design_structural_member_linear_campered_relative`. Such a complex instance is encoded in a STEP Part 21 file using external mapping. This does not mean that there is an offset at two positions. The absolute and relative cambers provide different definitions of the same camber. Both the size of the offset and the position of the offset provided by the `assembly_design_structural_member_linear_campered_absolute` should correspond to those provided `assembly_design_structural_member_linear_campered_relative`. (If they do not, the values provided by the `assembly_design_structural_member_linear_campered_relative` take precedence.)

Notes

New for CIS/2 2nd Edition.

See Diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.16 assembly_design_structural_member_linear_campered_relative

Entity definition:

A type of `assembly_design_structural_member_linear_campered` where the offset is provided by two instances of `ratio_measure_with_unit`.

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_member_linear_campered_relative
SUBTYPE OF (assembly_design_structural_member_linear_campered);
    relative_offset_position : ratio_measure_with_unit;
    relative_offset_y : ratio_measure_with_unit;

```

```

    relative_offset_z : ratio_measure_with_unit;
END_ENTITY;
(*)

```

Attribute definitions:

relative_offset_position

Declares the first instance of ratio_measure_with_unit that may be associated with the assembly_design_structural_member_linear_cambered to provide the numerical value with an appropriate unit of the linear dimension to the point where the camber offset is measured. The position is measured along the longitudinal axis of the assembly from its origin as a proportion of its length.

relative_offset_y

Declares the second instance of ratio_measure_with_unit that may be associated with the assembly_design_structural_member_linear_cambered to provide the numerical value with an appropriate unit of the linear dimension of offset in the first direction. The offset is measured perpendicular to the longitudinal axis as a proportion of the length of the assembly, in the positive y-direction.

relative_offset_z

Declares the third instance of ratio_measure_with_unit that may be associated with the assembly_design_structural_member_linear_cambered to provide the numerical value with an appropriate unit of the linear dimension of offset in the second direction. The offset is measured perpendicular to the longitudinal axis as a proportion of the length of the assembly, in the positive z-direction.

Informal propositions:

Where the camber is specified in both relative and absolute terms, this entity is instanced with the SUBTYPE assembly_design_structural_member_linear_cambered_absolute. Such a complex instance is encoded in a STEP Part 21 file using external mapping. This does not mean that there is an offset at two positions. The absolute and relative cambers provide different definitions of the same camber. Both the size of the offset and the position of the offset provided by the assembly_design_structural_member_linear_cambered_absolute should correspond to those provided assembly_design_structural_member_linear_cambered_relative. (If they do not, the values provided by the assembly_design_structural_member_linear_cambered_relative take precedence.)

Notes

New for CIS/2 2nd Edition.

See Diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.17 assembly_design_structural_member_linear_column

Entity definition:

A type of assembly_design_structural_member_linear, allowing the explicit description (or categorization) of a one dimensional structural member, for analysis, or member design purposes as a column. Columns are characterised as being vertical members (or members slightly inclined to the vertical) acting primarily in axial compression.

EXPRESS specification:

*)

```

ENTITY assembly_design_structural_member_linear_column
SUBTYPE OF (assembly_design_structural_member_linear);
    column_type : SET [1:?] OF member_column_type;
    slender_column : LOGICAL;
WHERE
    WRA38 : SELF\assembly_design_structural_member_linear.linear_member_type =
        COLUMN;
    WRA44 : SIZEOF(SELF\assembly_design_structural_member.restraints) > 1;
    WRA45 : SIZEOF(SELF\assembly_design_structural_member.effective_lengths) > 0;
END_ENTITY;
(*)

```

Attribute definitions:*column_type*

Declares the classification of the column by its primary form and possible use in other types of structural assembly. (See definition of the TYPE member_column_type.)

slender_column

Declares whether the column is slender (TRUE) or stocky (FALSE). Where the slenderness of the column is unknown, or has not yet been considered, this attribute should be assigned a value of UNKNOWN. The definition of when a column is considered 'slender' is dependent on the code of practice used for the column design, which will define when the restraints provided prevent buckling for the type of bending considered. It does not necessary follow that a column is restrained simply because the assembly_design_structural_member_linear_column is referenced by several instances of restraint.

Formal propositions:*WRA38*

The attribute linear_member_type inherited from the supertype assembly_design_structural_member_linear shall be assigned the value 'COLUMN'.

WRA44

The size of the set of instances of restraint derived by the attribute restraints (inherited from the supertype assembly_design_structural_member) shall be greater than 1. In other words, a column must have at least two restraints. Even if one end of the column is unrestrained (or even unsupported), this condition must be declared by an instance of restraint.

WRA45

The size of the set of instances of effective_buckling_length derived by the attribute effective_lengths (inherited from the supertype assembly_design_structural_member) shall be greater than 0. In other words, a column must have at least one effective_length defined.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.18 assembly_design_structural_member_planar

Entity definition:

A type of assembly_design_structural_member. Allows for the explicit description (or categorization) of a two dimensional structural member, for analysis, or member design purposes. Slabs and walls may be considered as planar structural members. This may be used as part of the design hierarchy. That is, a planar member may be part of a cubic member. For example, a slab may be part of a floor. An assembly_design_structural_member_planar may be further specialized through its subtypes as a plate, slab, or wall.

EXPRESS specification:

```
*)
ENTITY assembly_design_structural_member_planar
SUPERTYPE OF (ONEOF (
    assembly_design_structural_member_planar_plate,
    assembly_design_structural_member_planar_slab,
    assembly_design_structural_member_planar_wall))
SUBTYPE OF (assembly_design_structural_member);
    planar_member_type : member_planar_type;
    planar_member_components : SET [0:?] OF
        assembly_design_structural_member_linear;
END_ENTITY;
(*
```

Attribute definitions:

planar_member_type

Declares the classification of the planar member (i.e. 2 dimensional) by its primary form and possible use in other types of structural assembly. For example, slabs and walls are both considered to be 2 dimensional structural members. A (planar) slab is likely to be used as part of a (cubic) floor. Similarly, a stair element is likely to be used as part of a (cubic) stair.

planar_member_components

Declares the set of linear structural members associated with this instance of planar member. If the set is empty there are no linear structural members associated with this instance.

Notes:

New for CIS/2.

See diagram 60 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition (subtypes added)

3.3.19 assembly_design_structural_member_planar_plate

Entity definition:

A type of assembly_design_structural_member_planar, allowing the explicit description (or categorization) of a two dimensional structural member, for analysis, or member design purposes as a plate. Plates are typically idealized by a flat plane with properties based on a uniform thickness distributed about the plane. Plates are often components of other members.

EXPRESS specification:

*)

ENTITY assembly_design_structural_member_planar_plate

SUBTYPE OF (assembly_design_structural_member_planar);

plate_type : member_plate_type;

stiffened_plate : LOGICAL;

thick_plate : LOGICAL;

WHERE

WRA39 : SELF\assembly_design_structural_member_planar.planar_member_type =
PLATE;

END_ENTITY;

(*)

Attribute definitions:*plate_type*

Declares the classification of the plate by its primary form and possible use in other types of structural assembly. (See definition of the TYPE member_plate_type.)

stiffened_plate

Declares whether the plate is stiffened by additional plates (TRUE) or unstiffened (FALSE). Where the stiffened condition of the plate is unknown, or has not yet been considered, this attribute should be assigned a value of UNKNOWN.

thick_plate

Declares whether the plate is considered to be a 'thick plate' (TRUE) for design purposes or a 'thin plate' (FALSE). Where this design parameter is unknown, or has not yet been considered, this attribute should be assigned a value of UNKNOWN. When a plate is considered to be 'thick', it will not behave as a simple thin plate and normal thin plate theory does not apply; other aspects of the plate's behaviour need to be considered during the design process; e.g. through-thickness. The definition of when a plate is considered 'thick' is dependent on the code of practice used for the plate design.

Formal propositions:*WRA39*

The attribute planar_member_type inherited from the supertype assembly_design_structural_member_planar shall be assigned the value 'PLATE'.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.20 assembly_design_structural_member_planar_slab**Entity definition:**

A type of assembly_design_structural_member_planar, allowing the explicit description (or categorization) of a two dimensional structural member, for analysis, or member design purposes as a slab. Slabs are characterised as horizontal members (or members with a slight incline to the horizontal) primarily subject to bending.

EXPRESS specification:

*)

```

ENTITY assembly_design_structural_member_planar_slab
SUBTYPE OF (assembly_design_structural_member_planar);
    slab_type : member_slab_type;
WHERE
    WRA40 : SELF\assembly_design_structural_member_planar.planar_member_type =
        'SLAB';
END_ENTITY;
(*)

```

Attribute definitions:*slab_type*

Declares the classification of the brace by its primary form and possible use in other types of structural assembly. (See definition of the TYPE member_brace_type.)

Formal propositions:*WRA40*

The attribute planar_member_type inherited from the supertype assembly_design_structural_member_planar shall be assigned the value 'SLAB'.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.21 assembly_design_structural_member_planar_wall**Entity definition:**

A type of assembly_design_structural_member_planar, allowing the explicit description (or categorization) of a two dimensional structural member, for analysis, or member design purposes as a wall. Walls are characterised as vertical members.

EXPRESS specification:

```

*)
ENTITY assembly_design_structural_member_planar_wall
SUBTYPE OF (assembly_design_structural_member_planar);
    wall_type : member_wall_type;
WHERE
    WRA41 : SELF\assembly_design_structural_member_planar.planar_member_type =
        'WALL';
END_ENTITY;
(*)

```

Attribute definitions:*wall_type*

Declares the classification of the wall by its primary form and possible use in other types of structural assembly. (See definition of the TYPE member_wall_type.)

Formal propositions:*WRA41*

The attribute planar_member_type inherited from the supertype assembly_design_structural_member_planar shall be assigned the value 'WALL'.

Notes:

New for CIS/2 2nd Edition

See diagram 82 of the EXPRESS-G diagrams in Appendix B.

3.3.22 assembly_manufacturing

Entity definition:

A type of assembly, representing the physical frames and sub-frames. An assembly_manufacturing is any branch of the unique hierarchical composition performed during manufacture and erection. The composition mechanism is provided by the assembly_manufacturing_child.

EXPRESS specification:

```

*)
ENTITY assembly_manufacturing
SUPERTYPE OF (assembly_manufacturing_child)
SUBTYPE OF (assembly);
    surface_treatment : OPTIONAL text;
    assembly_sequence : OPTIONAL text;
    assembly_use : OPTIONAL text;
    place_of_assembly : OPTIONAL shop_or_site;
END_ENTITY;
(*

```

Attribute definitions:

surface_treatment

A free text description of the surface treatment(s) applied to the assembly_manufacturing as a whole (e.g. paint).

Note that 'local' treatments (i.e. only partially affecting a located_part of the assembly_manufacturing) are described as features (feature_surface_treatment) on the corresponding located_parts.

assembly_sequence

A free text description of the order in which the manufacturing assemblies are put together.

assembly_use

A free text description of the use or purpose of the assembly_manufacturing (e.g. temporary or permanent).

place_of_assembly

Declares the classification of the assembly_manufacturing by its place of assembly; i.e. in the fabrication shop or on the construction site.

Notes:

Known as MANUFACT_ASBLY CIS/1.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition

3.3.23 assembly_manufacturing_child

Entity definition:

An assembly_manufacturing_child is an assembly_manufacturing at any 'node' of the composition hierarchy (a product of the manufacturing process). This 'composition hierarchy' defines how located_parts, and those located_joint_systems which connect them, are brought together to form low level manufacturing assemblies. These (sub) manufacturing assemblies are then brought together with those located_joint_systems which connect them, plus any located_parts needed to connect the assemblies, to form higher level manufacturing assemblies. The final assembly_manufacturing at the top of the composition hierarchy corresponds to the complete structure. The composition hierarchy defined by the assembly_manufacturing_child may (but need not) have a strong similarity to the decomposition hierarchy defined by the assembly_design_child.

EXPRESS specification:

```
*)
ENTITY assembly_manufacturing_child
SUBTYPE OF (assembly_manufacturing);
    parent_assembly : assembly_manufacturing;
WHERE
    WRA19 : parent_assembly :<>: (SELF);
END_ENTITY;
(*
```

Attribute definitions:

parent_assembly

Declares the instance of assembly_manufacturing that acts as a parent to this instance of assembly_manufacturing_child. (An instance of assembly_manufacturing_child cannot exist without a corresponding parent instance of assembly_manufacturing.)

Formal propositions:

WRA19

An instance of a child manufacturing assembly cannot be its own parent. (This does not stop the child manufacturing assembly being a parent to another manufacturing assembly.)

Notes:

New for CIS/2. The composition mechanism is now contained in this entity rather than directly in the MANUFACT_ASBLY (in CIS/1).

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition

3.3.24 assembly_map

Entity definition:

An association between an assembly and a set of analytical elements.

EXPRESS specification:

```
*)
ENTITY assembly_map;
    represented_assembly : assembly;
```

representing_elements : SET [1:?] OF element;
 END_ENTITY;
 (*)

Attribute definitions:

represented_assembly

Declares the instance of assembly that is associated with the set of elements.

representing_elements

Declares the set of instance of element that are associated with the assembly.

Formal propositions:

There must be at least one instance of element associated with an instance of assembly_map, but there can be more.

Informal propositions:

The assembly under consideration may be any of the SUBTYPEs of assembly (design or manufacturing), and the elements under consideration may be any of the SUBTYPEs of element. Thus, a linear structural member may be associated with a set of (one or more) curved elements. Because the INVERSE has not been formally declared, its bounds are the default [0:?]. That is, an instance of assembly may be associated with zero, one or many instances of assembly_map. Similarly, an instance of element may be associated with zero, one or many instances of assembly_map.

Notes:

New for CIS/2.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition

3.3.25 assembly_relationship

Entity definition:

An association between two instances of assembly. The relationship must be given a name and may also be given a text description. For example, the relationship name could be 'is connected to'.

EXPRESS specification:

*)
 ENTITY assembly_relationship;
 relationship_name : label;
 relationship_description : OPTIONAL text;
 related_assembly : assembly;
 relating_assembly : assembly;
 WHERE
 WRA20 : related_assembly :<>: relating_assembly;
 END_ENTITY;
 (*)

Attribute definitions:

relationship_name

A short alphanumeric string by which the relationship is referred to.

relationship_description

An informative human-readable piece of text used to provide a general description of the relationship.

related_assembly

Declares the first instance of assembly involved in the relationship.

relating_assembly

Declares the second instance of assembly involved in the relationship.

Formal propositions:

WRA20

An instance of assembly cannot be related to itself. (This does not stop the assembly being related to several other assemblies.)

Informal propositions:

The relationship cannot exist without 2 associated instances of assembly (or the SUBTYPEs of assembly).

Notes:

New for CIS/2. Replaces the more restrictive (and more complex) ASBLY_CONNECTION in CIS/1.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition

3.3.26 assembly_with_bounding_box

Entity definition:

A type of assembly_with_shape that provides the association between the assembly and a block. Here the shape is constrained to be a geometric representation with one item defined using the block construct from STEP Part 42. This entity allows a ‘bounding box’ to be defined for a structural frame in its early design stages.

EXPRESS specification:

```

*)
ENTITY assembly_with_bounding_box
SUBTYPE OF (assembly_with_shape);
DERIVE
    bounding_box : SET [1:?] OF representation_item :=
        (SELF\assembly_with_shape.shape\representation.items);
WHERE
    WRA32 : SIZEOF(bounding_box) = 1;
    WRA33 : SIZEOF(QUERY(tmp <* bounding_box |
        ('STRUCTURAL_FRAME_SCHEMA.BLOCK') IN TYPE OF(tmp))) = 1;
END_ENTITY;
(*

```

Attribute definitions:*bounding_box*

Derives the set of instances of `representation_item` associated with this instance of assembly.

Formal propositions:*WRA32*

The size of the set of instances of `representation_item` derived by the attribute `bounding_box` shall equal one. In other words, the assembly will have only 1 representation item.

WRA33

The type of `representation_item` derived by the attribute `bounding_box` shall be of the type block.

Notes:

New for CIS/2 2nd Edition.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

3.3.27 assembly_with_shape**Entity definition:**

A type of assembly that provides the association between the assembly and the `shape_representation_with_units`. Here the shape could be a geometric representation or a topological representation defined using constructs from STEP Part 42. Because of the ANDOR declaration of the SUPERTYPE, any type of assembly could be given a shape. The subtype of this entity allows a 'bounding box' to be defined for a structural frame in its early design stages. It also allows a more precise and more complex representation of the manufactured sub-frames.

EXPRESS specification:

```
*)
ENTITY assembly_with_shape
SUBTYPE OF (assembly);
    shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:*shape*

Declares the instance of `shape_representation_with_units` associated with this instance of assembly. An `assembly_with_shape` cannot exist without a corresponding instance of `shape_representation_with_units`.

Informal propositions:

The assembly must have one (and only one) `shape_representation_with_units`, but the `shape_representation_with_units` may be associated with any number of assemblies.

Notes:

New for CIS/2.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Entity definition modified and subtype added for CIS/2 2nd Edition.

3.3.28 design_criterion

Entity definition:

Allows a set of governing criteria relevant to analysis and design to be assigned to a particular set of instances of assembly_design. It has a name and a description.

EXPRESS specification:

*)

ENTITY design_criterion

SUPERTYPE OF (design_criterion_documented);

criterion_name : label;

criterion_description : text;

design_assumptions : OPTIONAL text;

INVERSE

governed_assemblies : SET [1:?] OF assembly_design FOR governing_criteria;

END_ENTITY;

(*

Attribute definitions:

criterion_name

A short alphanumeric reference for the instance of design_criterion.

criterion_description

A text description of the design_criterion.

design_assumptions

A free text description of the design assumptions made for the design of instances of assembly_design that refer to this instance of design_criterion.

Formal propositions:

governed_assemblies

The design criterion may be used to govern one, or many instances of assembly_design; i.e. the set has no upper bound.

Notes

Known as DESIGN_CRITERIA in CIS/1.

See diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.29 design_criterion_documented

Entity definition:

A type of design_criterion that has a reference to a part of a document. For example, a particular design_criterion may be documented by a clause of a national standard or code of practice.

EXPRESS specification:

```

*)
ENTITY design_criterion_documented
SUBTYPE OF (design_criterion);
    documented_reference : document_usage_constraint;
END_ENTITY;
(*

```

Attribute definitions:*documented_reference*

Declares the instance of document_usage_constraint associated with this instance of design_criterion_documented. This could be, for example, a clause of a national standard or code of practice.

Notes

New for CIS/2.

See Diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.30 design_joint_system**Entity definition:**

An occurrence level instance of the corresponding specific level joint_system. Here, the specification for a joint_system may be given a number of locations and orientations. It may also be associated with a number of parent instances of assembly_design. It may also be associated with a number of instances of design_part. It is assumed that the instances of design_part associated with the design_joint_system are connected by the design_joint_system.

The design_joint_system can be given a name and number for the purposes of identification.

This is a conceptual device used for connection design purposes. It bridges the gap between the analytical node (merely a point in space) and the located_joint_system. A design_joint_system may be involved in many instances of assembly_design. A design_joint_system may, or may not, correspond directly to an analytical node. A design_joint_system is used to design a number of located_joint_systems. In other words, it is a representation of the physical weld or bolt system for connection design purposes. Just as physical located_parts are joined by located_joint_systems (either bolts or welds), design_parts are connected by design_joint_systems (but it may not always be explicit how that requirement is fulfilled).

The only link between this entity and structural connection is through the assembly_design entity. The structural connection is simply a type of assembly_design.

EXPRESS specification:

```

*)
ENTITY design_joint_system;
    design_joint_system_name : label;
    design_joint_system_spec : joint_system;
    parent_assemblies : LIST [1:?] OF assembly_design;
    locations : OPTIONAL LIST [1:?] OF coord_system;

```

```

    connected_parts : SET [0:?] OF design_part;
WHERE
    WRD8 : NOT(EXISTS(locations) AND (SIZEOF(locations) <>
        SIZEOF(parent_assemblies)));
END_ENTITY;
(*)

```

Attribute definitions:

design_joint_system_name

A short alphanumeric reference for the design_joint_system.

design_joint_system_spec

Declares the instance of joint_sytem associated with this design_joint_system, and which provides the specification for the design_joint_system.

parent_assemblies

Declares the list of instances of assembly_design associated with this design_joint_system, and to which it belongs. The design_joint_system must belong to at least one parent instance of assembly_design. Since the INVERSE is not declared in the entity assembly_design, an instance of assembly_design can exist without a design_joint_system.

If a design_joint_system appears twice in one parent assembly, then the reference to the instance of assembly_design appears twice in the list of parent_assemblies.

locations

Declares the list of instances of coord_system that may be associated with this design_joint_system to provide the locations and orientations of the design_joint_system within the parent assembly. The design_joint_system may be located by zero, one or many instances of coord_system. Further, since the INVERSE is not declared in the entity coord_system, an instance of coord_system can exist without a design_joint_system.

The order of the list of instances of coord_system is important. It is implied that the list of locations is in the same order as the list of parent_assemblies. Each design_joint_system coordinate system is expected to be located within the coordinate system of the parent assembly.

If the design_joint_system appears twice in the parent assembly, then a second instance of coord_system is required in the list of locations. Both instances of coord_system will have the same parent coord_system.

connected_parts

Declares the set of instances of design_part associated with this design_joint_system, and which it is assumed to connect. As the set is unbounded, the design_joint_system may connect zero, one or many instances of design_part. Further, since the INVERSE is not declared, the design_part can exist without a design_joint_system.

Formal propositions:

WRD8

If a list of instances of coord_system exists, then the size of that list shall equal the size of the list of instances of assembly_design associated with this instance of

design_joint_system. That is, if the attribute locations is populated, then the size of the list of coordinate systems shall be the same size as the list of the parent assemblies.

Informal propositions:

In the simplest case - a hierarchical design decomposition - the design_joint_system would have one location and belong to one parent assembly_design.

Notes:

Known as CONNECTOR in CIS/1.

See diagram 38 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.31 design_part

Entity definition:

An occurrence level instance of the corresponding specific level part. Here, the specification for a part may be given a number of locations and orientations. It may also be associated with a number of parent instances of assembly_design. The design_part can be given a name for the purposes of identification.

This is a conceptual device used for member design purposes. It bridges the gap between the analytical elements (merely lines in space) and the physical located_parts. A design_part may be involved in many design_assemblies. A design_part may, or may not, correspond directly to an analytical element. A design_part is used to design a number of located_parts. In other words, it is a representation of the physical part for member design purposes. Just as physical located_parts and joined by located_joint_systems (either bolts or welds), design_parts are connected by design_joint_systems (but it may not always be explicit how that requirement is fulfilled). The specification of a design_part is through its attribute design_part_spec, which provides the relationship to part.

EXPRESS specification:

```
*)
ENTITY design_part;
    design_part_name : label;
    design_part_spec : part;
    parent_assemblies : LIST [1:?] OF assembly_design;
    locations : OPTIONAL LIST [1:?] OF coord_system;
WHERE
    WRD9 : NOT(EXISTS(locations) AND (SIZEOF(locations) <>
        SIZEOF(parent_assemblies)));
END_ENTITY;
(*
```

Attribute definitions:

design_part_name

A short alphanumeric reference for the design_part.

design_part_spec

Declares the instance of part associated with this design_part, and which provides the specification for the part.

parent_assemblies

Declares the list of instances of `assembly_design` associated with this `design_part`, and to which it belongs. The `design_part` must belong to at least one parent instance of `assembly_design`. Further, since the INVERSE is not declared in the entity `assembly_design`, an instance of `assembly_design` can exist without a `design_part`.

If a `design_part` appears twice in one parent assembly, then the reference to the instance of `assembly_design` appears twice in the list of `parent_assemblies`.

locations

Declares the list of instances of `coord_system` associated with this `design_part`, and which provide the locations and orientations of the `design_part` within the parent assembly. The `design_part` must be located by at least one instance of `coord_system`. Further, since the INVERSE is not declared in the entity `coord_system`, instances of `coord_system` can exist without a `design_part`.

The order of the list of instances of `coord_system` is important. It is implied that the list of locations is in the same order as the list of `parent_assemblies`. Each `design_part` coordinate system is expected to be located within the coordinate system of the parent assembly.

If the `design_part` appears twice in the parent assembly, then a second instance of `coord_system` is required in the list of locations. Both instances of `coord_system` will have the same parent `coord_system`.

Formal propositions:*WRD9*

If a list of instances of `coord_system` exists, then the size of that list shall equal the size of the list of instances of `assembly_design` associated with this instance of `design_part`. That is, if the attribute `locations` is populated, then the size of the list of coordinate systems shall be the same size as the list of the parent assemblies.

Informal propositions:

In the simplest case - a hierarchical design decomposition - the `design_part` would have one location and belong to one parent `assembly_design`.

Notes:

Known as `DESIGN_PART` in CIS/1.

See diagram 38 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.32 effective_buckling_length**Entity definition:**

A structural member may have a number of effective lengths depending upon the axis in question. This entity allows for the definition of an effective length for a structural member subject to buckling.

EXPRESS specification:

*)

```
ENTITY effective_buckling_length;
    effective_length_name : label;
    effective_length_factor : OPTIONAL REAL;
```

```

    effective_length_use : OPTIONAL text;
    effective_length_direction : OPTIONAL buckling_direction;
    applicable_member : assembly_design_structural_member;
END_ENTITY;
(*)

```

Attribute definitions:

effective_length_name

A short text reference for the effective_buckling_length.

effective_length_factor

Numerical value of the factor which, when multiplied by the actual length, gives the effective buckling length. This is specified in accordance with the appropriate standard.

effective_length_use

A free text description of how the effective buckling length is to be used.

effective_length_direction

Declares the classification of the direction about which the effective buckling length acts; i.e.

- x_dir = effective length when considering buckling in the x-direction,
- y_dir = buckling in the y-direction (for an I section - buckling about the minor axis),
- z_dir = buckling in the z-direction (for an I section - buckling about the major axis).

applicable_member

Declares the structural member to which this instance of effective buckling length applies.

Informal propositions:

Because the INVERSE has not be explicitly declared, an assembly_design_structural_member may have zero, one or many effective buckling lengths.

Notes:

Known as EFFECTIVE_BUCKLING_LENGTH in CIS/1.

See diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.33 functional_role

Entity definition:

Defines and describes the function of the assembly_design. It recognizes that an assembly_design may have many different functional roles. In many cases, the same structural member or frame may be acting in more than one role. For example, a column may be part of the vertical load-carrying structure, and at the same time, it may be acting as part of the wind bracing.

EXPRESS specification:

```

*)
ENTITY functional_role
SUPERTYPE OF (functional_role_documented);
    functional_role_name : label;
    functional_role_description : text;
INVERSE
    role_for_assemblies : SET [1:?] OF assembly_design FOR roles;
END_ENTITY;
(*

```

Attribute definitions:

functional_role_name

A short alphanumeric reference for the functional role.

functional_role_description

A free text description of the functional role (e.g. wind bracing).

Formal propositions:

role_for_assemblies

The functional role must be used with at least one design assembly. An instance of functional_role cannot exist without being referenced by an instance of assembly_design.

Informal propositions:

The assembly_design may have many functional roles, but may exist without one.

Notes:

Known as FUNCTIONAL_ROLE in CIS/1.

See diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.34 functional_role_documented

Entity definition:

A type of functional_role that makes reference to part of a document. For example, the function that an assembly is designed to perform may be documented by a clause of a national standard or code of practice.

EXPRESS specification:

```

*)
ENTITY functional_role_documented
SUBTYPE OF (functional_role);
    document_reference : document_usage_constraint;
END_ENTITY;
(*

```

Attribute definitions:*document_reference*

Declares the instance of `document_usage_constraint` associated with this instance of `functional_role_documented`. The instance of `document_usage_constraint` references a part (chapter, subsection or clause) of a document.

The entities `document` and `document_usage_constraint` are defined in STEP Part 41.

Notes

New for CIS/2.

See Diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.35 restraint**Entity definition:**

Used to define the properties and location of a restraint. The restraint may then be used to derive the effective length of a structural member.

EXPRESS specification:

*)

ENTITY restraint

ABSTRACT SUPERTYPE OF (ONEOF(restraint_logical, restraint_spring));

 restraint_name : label;

 restraint_description : OPTIONAL text;

 restraint_location : point;

 restrained_mbr : assembly_design_structural_member;

END_ENTITY;

(*

Attribute definitions:*restraint_name*

A short alphanumeric reference for the restraint.

restraint_description

A free text description of the restraint.

restraint_location

The numerical value with a specified unit of the distance to the restraint measured from the start of the structural member, in the local member coordinate system.

restrained_mbr

Declares the structural member (i.e. the instance of `assembly_design_structural_member`), to which the restraint applies.

The restraint cannot exist without a corresponding instance of `assembly_design_structural_member`. The instance of restraint applies for only one instance of `assembly_design_structural_member`. Because the INVERSE clause has not been explicitly declared, the structural member can exist without a restraint, or may be associated with one or more restraints.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

Known as RESTRAINT in CIS/1.

The conventions used for member restraints differ from those for nodal boundary conditions and element end releases. Thus, the definitions for the entity restraint and its SUBTYPES differ from those of boundary_condition and release and their SUBTYPES.

See diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.36 restraint_logical**Entity definition:**

A type of restraint where the six degrees of freedom are given logical values (true, false or unknown). This is a simple representation of a member restraint. Structural members whose design is considered in three dimensions should be referenced by a restraint with all of its attributes assigned values as either TRUE or FALSE. Structural members whose design is considered in one plane only (i.e. out of plane conditions ignored) will be referenced by a restraint with some of its attributes assigned a value of UNKNOWN.

Each of the six attributes may be assigned a value of either TRUE, FALSE, or UNKNOWN. This corresponds to a member restraint defined in the local coordinate system of the member that is either 'fixed', 'free', or UNKNOWN. The value will be UNKNOWN if the degree of freedom is not being considered – as may be the case for a member design that ignores out of plane conditions.

EXPRESS specification:

```
*)
ENTITY restraint_logical
SUBTYPE OF (restraint);
    restraint_x_displacement : LOGICAL;
    restraint_y_displacement : LOGICAL;
    restraint_z_displacement : LOGICAL;
    restraint_x_rotation : LOGICAL;
    restraint_y_rotation : LOGICAL;
    restraint_z_rotation : LOGICAL;
END_ENTITY;
(*
```

Attribute definitions:**restraint_x_displacement**

Declares the value of the degree of freedom for the restraint for linear displacement in the x direction, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the x direction is restrained this attribute is assigned the value TRUE. If it is unrestrained, this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

restraint_y_displacement

Declares the value of the degree of freedom for the restraint for linear displacement in the y direction, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the y direction is restrained this attribute is assigned the value TRUE. If it is unrestrained, this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

restraint_z_displacement

Declares the value of the degree of freedom for the restraint for linear displacement in the z direction, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the z direction is restrained this attribute is assigned the value TRUE. If it is unrestrained, this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

restraint_x_rotation

Declares the value of the degree of freedom for the restraint for rotation about the x axis of the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the x axis is restrained this attribute is assigned the value TRUE. If it is unrestrained, this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

restraint_y_rotation

Declares the value of the degree of freedom for the restraint for rotation about the y axis of the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the y axis is restrained this attribute is assigned the value TRUE. If it is unrestrained, this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

restraint_z_rotation

Declares the value of the degree of freedom for the restraint for rotation about the z axis of the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the z axis is restrained this attribute is assigned the value TRUE. If it is unrestrained, this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

Notes

New for CIS/2.

See Diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

3.3.37 restraint_spring**Entity definition:**

A type of restraint where the six degrees of freedom are given by references to spring stiffness values. This is a more complex representation of a member restraint than the *restraint_logical*. Structural members whose design is considered in three dimensions should be referenced by a restraint with all of its attributes assigned values that reference

instances of `linear_stiffness_measure_with_unit` or `rotational_stiffness_measure_with_unit`. Structural members whose design is considered in one plane only (i.e. out of plane conditions ignored) will be referenced by a restraint with some of its attributes unpopulated.

EXPRESS specification:

*)

ENTITY `restraint_spring`

SUPERTYPE OF (`restraint_warping`)

SUBTYPE OF (`restraint`);

`restraint_x_displacement` : OPTIONAL `linear_stiffness_measure_with_unit`;

`restraint_y_displacement` : OPTIONAL `linear_stiffness_measure_with_unit`;

`restraint_z_displacement` : OPTIONAL `linear_stiffness_measure_with_unit`;

`restraint_x_rotation` : OPTIONAL `rotational_stiffness_measure_with_unit`;

`restraint_y_rotation` : OPTIONAL `rotational_stiffness_measure_with_unit`;

`restraint_z_rotation` : OPTIONAL `rotational_stiffness_measure_with_unit`;

WHERE

WRR31 : EXISTS (`restraint_x_displacement`) OR

EXISTS (`restraint_y_displacement`) OR

EXISTS (`restraint_z_displacement`) OR

EXISTS (`restraint_x_rotation`) OR

EXISTS (`restraint_y_rotation`) OR

EXISTS (`restraint_z_rotation`);

END_ENTITY;

(*

Attribute definitions:

restraint_x_displacement

Declares the first instance of `linear_stiffness_measure_with_unit` that may be associated with this instance of `restraint_spring`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for linear displacement in the x direction, in the local coordinate system. If the linear displacement in the x direction is unknown this attribute is unpopulated and would appear in a STEP Part 21 file as the null value (\$).

restraint_y_displacement

Declares the second instance of `linear_stiffness_measure_with_unit` that may be associated with this instance of `restraint_spring`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for linear displacement in the y direction, in the local coordinate system. If the linear displacement in the y direction is unknown this attribute is unpopulated and would appear in a STEP Part 21 file as the null value (\$).

restraint_z_displacement

Declares the third instance of `linear_stiffness_measure_with_unit` that may be associated with this instance of `restraint_spring`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for linear displacement in the z direction, in the local coordinate system. If the linear displacement in the z direction is unknown this attribute is unpopulated and would appear in a STEP Part 21 file as the null value (\$).

restraint_x_rotation

Declares the first instance of `rotational_stiffness_measure_with_unit` that may be associated with this instance of `restraint_spring`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for rotation about the x axis of the local coordinate system. If the rotation about the x axis is unknown this attribute is unpopulated and would appear in a STEP Part 21 file as the null value (\$).

restraint_y_rotation

Declares the second instance of `rotational_stiffness_measure_with_unit` that may be associated with this instance of `restraint_spring`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for rotation about the y axis of the local coordinate system. If the rotation about the y axis is unknown this attribute is unpopulated and would appear in a STEP Part 21 file as the null value (\$).

restraint_z_rotation

Declares the third instance of `rotational_stiffness_measure_with_unit` that may be associated with this instance of `restraint_spring`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for rotation about the z axis of the local coordinate system. If the rotation about the z axis is unknown this attribute is unpopulated and would appear in a STEP Part 21 file as the null value (\$).

Formal propositions:***WRR31***

One of the attributes `restraint_x_displacement`, `restraint_y_displacement`, `restraint_z_displacement`, `restraint_x_rotation`, or `restraint_y_rotation` `restraint_z_rotation` must be assigned a value. That is, a spring restraint must be given at least one value for its spring stiffness for either linear displacement or rotation

Notes

New for CIS/2.

See Diagram 61 of the EXPRESS-G diagrams in Appendix B.

Attribute definitions clarified for 2nd Edition.

3.3.38 restraint_warping***Entity definition:***

A type of spring restraint where a seventh degree of freedom is added by references to an additional spring stiffness value. This is a more complex representation of a member restraint than the `restraint_logical` or the `restraint_spring`.

EXPRESS specification:

*)

ENTITY `restraint_warping`

SUBTYPE OF (`restraint_spring`);

`restraint_w_rotation` : `rotational_stiffness_measure_with_unit`;

END_ENTITY;

(*)

Attribute definitions:

restraint_w_rotation

Declares the first instance of `rotational_stiffness_measure_with_unit` that may be associated with this instance of `restraint_warping`. This attribute provides the value with a specified unit of the spring stiffness of the degree of freedom for the restraint for rotation about the warping axis of the local coordinate system.

Notes

New for CIS/2.

See Diagram 61 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4 LPM/6 PROJECT DEFINITION

4.1 Project Definition concepts and assumptions

4.1.1 Conceptual overview

A structure may be placed in the context of a project, a site, a building or a building complex. Each of these may be divided into zones. Buildings, sites and structures may all have interrelated grids, with setting out points, which may be resolved to geographical locations. LPM/6 supports orthogonal, radial and skewed grids.

4.2 Project Definition type definitions

No types have been added or modified for the 2nd Edition in this subject area.

4.2.1 degrees_rotation

Type definition:

The integer value of degrees of angular rotational measure. Used when specifying a global location.

EXPRESS specification:

```
*)
TYPE degrees_rotation
= INTEGER;
WHERE
    WRTD2 : {-180 < SELF <= 180};
END_TYPE;
(*
```

Formal propositions:

WRTD2

The integer value shall lie between -180 and 180.

Notes:

New for CIS/2. Unchanged in 2nd Edition.

4.2.2 minutes_rotation

Type definition:

The integer value of the minutes of angular rotational measure. Used when specifying a global location.

EXPRESS specification:

```
*)
TYPE minutes_rotation
= INTEGER;
WHERE
    WRTM3 : {0 <= SELF < 60};
END_TYPE;
(*
```

Formal propositions:*WRTM3*

The integer value shall lie between 0 and 60.

Notes:

New for CIS/2. Unchanged in 2nd Edition.

4.2.3 seconds_rotation**Type definition:**

The real number value of seconds of angular rotational measure. Used when specifying a global location.

EXPRESS specification:

```
*)
TYPE seconds_rotation
  = REAL;
WHERE
  WRTS2 : {0.0 <= SELF < 60.0};
END_TYPE;
(*
```

Formal propositions:*WRTS2*

The real value shall lie between 0 and 60.

Notes:

New for CIS/2. Unchanged in 2nd Edition.

4.3 Project Definition entity definitions

The following entities have been modified for the 2nd Edition of CIS/2:

- building
- zone_of_structure

The following entities have been added for the 2nd Edition of CIS/2:

- zone_of_structure_sequence
- zone_of_structure_sequence_lot

4.3.1 building**Entity definition:**

A type of *structural_frame_item* that associates a list of structures with a building owner. The concept of a building implies an enclosure and a supporting structure. The building must be given a number and a name, and may be given a description and a class. A building may be given a shape through the SUBTYPE *building_with_shape*. It may belong to a building complex - which in turn is associated with a site.

As a SUBTYPE of *structural_frame_item*, a building inherits the three attributes *item_number*, *item_name*, and *item_description*. It also inherits the many implicit

relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a building may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A building may also be related to another building (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

```
*)
ENTITY building
SUPERTYPE OF (building_with_shape)
SUBTYPE OF (structural_frame_item);
    building_class : OPTIONAL label;
    owner : OPTIONAL person_and_organization;
    building_structures : OPTIONAL LIST [1:?] OF structure;
UNIQUE
    URB1 : SELF\structural_frame_item.item_number,
        SELF\structural_frame_item.item_name;
END_ENTITY;
(*
```

Attribute definitions:

SELF\structural_frame_item.item_number

An integer reference that provides one half of the unique identification of the building. (This attribute is inherited from the SUPERTYPE `structural_frame_item`.)

SELF\structural_frame_item.item_name

A short alphanumeric reference that provides the other half of the unique identification of the building. (This attribute is inherited from the SUPERTYPE `structural_frame_item`.)

SELF\structural_frame_item.item_description

A free text description of the building. (This attribute is inherited from the SUPERTYPE `structural_frame_item`.)

building_class

A short alphanumeric reference that declares the classification of the building for design purposes. Values of applied physical actions may be dependent on the building class - the building class may be used to define the derivation factors in such cases.

owner

Declares the instance of `person_and_organization` that may be associated with this building, and assumed to act as the owner of the building.

building_structures

Declares the list of structures associated with this building. The LIST data type allows repetition such that a building may have two building structures, but each of them may be specified using the same instance of structure.

Formal propositions:*URB1*

The combination of the item_name and the item_number (both inherited from the SUPERTYPE structural_frame_item) must be unique to this instance of building. That is, no other building may have the same combination of name and number.

Informal propositions:

Because the INVERSE clause has not been declared explicitly, the structure may be associated with zero, one or more buildings.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Modified for CIS/2 2nd Edition – attribute building_structures made OPTIONAL. There may be times when a building needs to be defined without reference to a structure.

4.3.2 building_complex**Entity definition:**

A type of structural_frame_item that associates a site with a number of buildings. The building_complex must be given a number and a name and may be given a description.

As a SUBTYPE of structural_frame_item, a building_complex inherits the three attributes item_number, item_name, and item_description. It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a building_complex may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented)
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A building_complex may also be related to another building_complex (or any other structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

*)

ENTITY building_complex

SUBTYPE OF (structural_frame_item);

building_site : site;

buildings : LIST [1:?] OF building;

UNIQUE

```

        URB2 : SELF\structural_frame_item.item_number,
              SELF\structural_frame_item.item_name;
    END_ENTITY;
    (*)

```

Attribute definitions:

SELF\structural_frame_item.item_number

An integer reference that provides one half of the unique identification of the building_complex. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_name

A short alphanumeric reference that provides the other half of the unique identification of the building_complex. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_description

A free text description of the building_complex. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

building_site

Declares the instance of site associated with this building_complex, and which it is assumed to be located (and built) on.

buildings

Declares the set of instances of building associated with this building_complex. It is implied that the building_complex is made up of these buildings. There must be at least one building associated with the building_complex; i.e. the building_complex cannot exist without a building. The LIST data type allows repetition such that a building_complex may have two buildings, but each of them may be specified using the same instance of building.

Formal propositions:

URB2

The combination of the item_name and the item_number (both inherited from the SUPERTYPE structural_frame_item) must be unique to this instance of building_complex. That is, no other building_complex may have the same combination of name and number.

Informal propositions:

Because the INVERSE clause has not been declared explicitly, the building may be associated with zero, one or more building complexes.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.3 building_with_shape

Entity definition:

A type of building, providing the association between the building and the shape_representation_with_units. Here the shape could be a geometric representation or a topological representation defined using constructs from STEP Part 42. Providing the shape in the SUBTYPE allows the building to be defined with or without a shape.

EXPRESS specification:

```
*)
ENTITY building_with_shape
SUBTYPE OF (building);
    shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:

shape

Declares the instance of shape_representation_with_units associated with the building_with_shape, providing the specification for the shape of the building. A building_with_shape cannot exist without a corresponding instance of shape_representation_with_units.

Informal propositions:

The building must have one (and only one) shape_representation_with_units, but the shape_representation_with_units may be associated with any number of buildings. That is, the building can have only one shape, but several buildings may have the same shape.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.4 geographical_location

Entity definition:

Provides the location of a point relative to a global system. A geographical_location must be specified as either a global_location (specified in terms of longitude and latitude), or a map_location (specified in terms of Northings and Eastings). In both cases, the heights of the positions are located with respect to a datum.

EXPRESS specification:

```
*)
ENTITY geographical_location
ABSTRACT SUPERTYPE OF (ONEOF(global_location, map_location));
    height_above_datum : length_measure_with_unit;
    datum_name : label;
END_ENTITY;
(*
```

Attribute definitions:*height_above_datum*

The numerical value with a specified unit of the height of the position above a specified datum reference.

datum_name

The short alphanumeric reference of the datum for this geographical_location (e.g. in the UK this would probably be related to the Ordinance Survey reference datum).

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2. The SITE entity in CIS/1 has had the attributes st_altitude and grid_ref removed.

See diagram 11 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.5 global_location**Entity definition:**

A type of geographical_location which adds the position (in terms of longitude and latitude) to the height above datum it inherits from geographical_location.

EXPRESS specification:

*)

ENTITY global_location

SUBTYPE OF (geographical_location);

latitude_degrees : degrees_rotation;

latitude_minutes : minutes_rotation;

latitude_seconds : OPTIONAL seconds_rotation;

longitude_degrees : degrees_rotation;

longitude_minutes : minutes_rotation;

longitude_seconds : OPTIONAL seconds_rotation;

END_ENTITY;

(*

Attribute definitions:*latitude_degrees*

Specifies the value of the position's latitude in terms of degrees of angular rotation. The value must lie between ± 180 . A positive value represents a line of latitude in the Northern hemisphere. A zero value represents the equator, while negative values represent lines of latitude lying in the Southern hemisphere.

latitude_minutes

Specifies the value of the position's latitude in terms of minutes of angular rotation. The value must lie between 0 and 60.

latitude_seconds

Specifies the value of the position's latitude in terms of seconds of angular rotation. A value need not be given. If a value is provided, it must lie between 0 and 60.

longitude_degrees

Specifies the value of the position's longitude in terms of degrees of angular rotation. The value must lie between ± 180 . A positive value represents a line of longitude in the Eastern hemisphere. A zero value represents the Greenwich meridian, while negative values represent lines of longitude lying in the Western hemisphere.

longitude_minutes

Specifies the value of the position's longitude in terms of minutes of angular rotation. The value must lie between 0 and 60.

longitude_seconds

Specifies the value of the position's longitude in terms of seconds of angular rotation. A value need not be given. If a value is provided, it must lie between 0 and 60.

Notes:

New for CIS/2.

See diagram 11 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.6 grid**Entity definition:**

An abstract concept that allows a collection of gridlines and grid levels to be associated with a building, a site or a structure. The grid can be orthogonal, skewed or radial. The grid can be given a name, a description and a use. (Only the name is compulsory.)

EXPRESS specification:

*)

ENTITY grid

SUPERTYPE OF (ONEOF

(grid_of_building,
grid_of_site,
grid_of_structure) ANDOR ONEOF
(grid_orthogonal,
grid_skewed,
grid_radial));

grid_name : label;

grid_description : OPTIONAL text;

grid_use : OPTIONAL text;

DERIVE

gridlines : SET [1:?] OF gridline := bag_to_set (USEDIN (SELF,
'STRUCTURAL_FRAME_SCHEMA.GRIDLINE.PARENT_GRID'));

grid_levels : SET [0:?] OF grid_level := bag_to_set (USEDIN (SELF,
'STRUCTURAL_FRAME_SCHEMA.GRID_LEVEL.PARENT_GRID'));

INVERSE

constituent_lines : SET [1:?] OF gridline FOR parent_grid;

END_ENTITY;

(*

Attribute definitions:

grid_name

A short alphanumeric reference for the grid.

grid_description

A free text description of the grid.

grid_use

A free text description of use or purpose of the grid.

gridlines

This attribute derives the set of instances of gridline that are associated with this instance of grid. There must be at least one gridline that uses this instance of grid as its parent_grid. Each instance in the set must be unique, i.e. no repetition of gridlines is allowed because the data type is a SET. The maximum number of gridlines that use this instance of grid as its parent_grid is unlimited.

This derived attribute will not appear in a STEP Part 21 file.

grid_levels

This attribute derives the set of instances of grid_level that are associated with this instance of grid. The maximum number of instances of grid_level using this instance of grid as its parent_grid is unlimited. There may not be any instances of grid level that use this instance of grid as its parent_grid. Each instance in the set must be unique, i.e. no repetition of grid levels is allowed because the data type is a SET.

This derived attribute will not appear in a STEP Part 21 file.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

DERIVE

The derived attributes gridlines and grid_levels do not appear in the STEP Part 21 file.

constituent_lines

The grid must have at least one gridline, but may have more.

This INVERSE clause constrains the relationship from the entity gridline created by the attribute parent_grid. This serves a different function from the derived attribute gridlines.

Notes:

Known as GRIDLINE_SET in CIS/1.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.7 grid_intersection

Entity definition:

An abstract concept that allows the intersection of two gridlines and (optionally) a grid level to be given a name. The intersection may be given a geographical location through its SUBTYPE.

EXPRESS specification:

```
*)
ENTITY grid_intersection
SUPERTYPE OF (grid_intersection_resolved);
  grid_intersection_name : label;
  gridlines : SET [2:2] OF gridline;
  level : OPTIONAL grid_level;
WHERE
  WRG4 : gridlines[1].parent_grid :=: gridlines[2].parent_grid;
  WRG5 : NOT (EXISTS(level) AND (level.parent_grid :<>: gridlines[1].parent_grid));
END_ENTITY;
(*
```

Attribute definitions:

grid_intersection_name

A short alphanumeric reference for the grid_intersection.

gridlines

Declares the two distinct instances of gridline associated with this grid_intersection. The line of intersection lies where the vertical planes through the two gridlines cross.

level

Declares the instance of grid_level that may be associated with this grid_intersection. The point of intersection lies where the horizontal plane of the grid level and the vertical planes of the two gridlines cross.

Formal propositions:

WRG4

The instance of grid associated with the first referenced gridline (and used as its parent_grid) must be the same instance of grid that is associated with the second gridline referenced by the attribute gridlines. That is, both gridlines must belong to the same parent grid.

WRG5

If the attribute level is populated, then the instance of grid associated with the that instance of grid_level (and used as its parent_grid) must be the same instance of grid that is associated with the first gridline referenced by the attribute gridlines. That is, if the grid_intersection has a grid_level, then it must belong to the same parent grid as the gridlines.

Informal propositions:

The two instances of gridline are not parallel.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

4.3.8 grid_intersection_resolved

Entity definition:

A type of grid_intersection that provides the geographical location of the grid_intersection.

EXPRESS specification:

```
*)
ENTITY grid_intersection_resolved
  SUBTYPE OF (grid_intersection);
    resolution_point : geographical_location;
END_ENTITY;
(*
```

Attribute definitions:

resolution_point

Declares the instance of geographical_location associated with the grid_intersection_resolved, and which therefore locates the grid_intersection within a global referencing system.

Informal propositions:

The grid_intersection_resolved cannot exist without a geographical_location. However, the geographical_location may be used to locate zero, one or many grid intersections.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.9 grid_level

Entity definition:

A type of plane that is related to a grid (which is in turn related to a building, a site, or a structure). A grid level is defined such that it lies in a horizontal plane (although nothing in the EXPRESS schema enforces this constraint). The grid_level is defined by a plane, which is defined by three orthogonal axes (inherited from the attribute position of the SUPERTYPE elementary_surface). The z-axis lies normal to the plane.

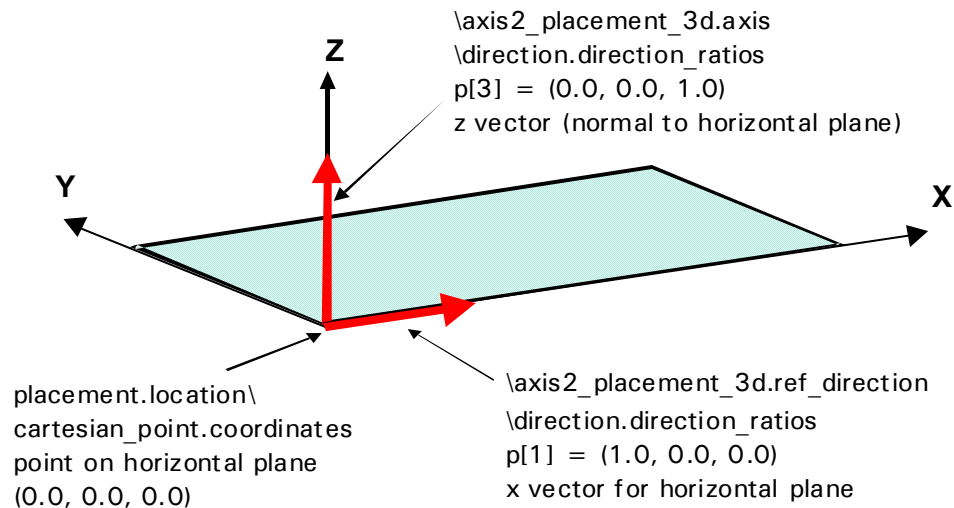


Figure 4.1 *Defining a grid_level at the origin of the grid (zero altitude)*

EXPRESS specification:

```
*)
ENTITY grid_level
SUBTYPE OF (plane);
    parent_grid : grid;
UNIQUE
    URG1 : SELFelementary_surface.position;
END_ENTITY;
(*
```

Attribute definitions:

parent_grid

Declares the instance of grid that is associated with the grid_level. The grid_level can have (or belong to) only one parent grid. The grid_level cannot exist without a parent grid.

Formal propositions:

URG1

The instance of axis2_placement_3d (referenced by the attribute position inherited from the SUPERTYPE elementary_surface) must be unique to this instance of grid_level. In other words, grid levels must not share the same position in 3D Cartesian space.

Informal propositions:

The grid level lies in a horizontal plane.

Notes:

New for CIS/2.

The SUPERTYPE entities plane and elementary_surface are defined in STEP Part 42.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.10 grid_of_building

Entity definition:

A type of grid that is associated with a building.

EXPRESS specification:

```
*)
ENTITY grid_of_building
SUBTYPE OF (grid);
    grid_for_building : building;
END_ENTITY;
(*)
```

Attribute definitions:

grid_for_building

Declares the instance of building associated with the grid_of_building. The grid_of_building can have (or belong to) only one building. The grid_of_building cannot exist without a building.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.11 grid_of_site

Entity definition:

A type of grid that is associated with a site.

EXPRESS specification:

```
*)
ENTITY grid_of_site
SUBTYPE OF (grid);
    grid_for_site : site;
END_ENTITY;
(*)
```

Attribute definitions:

grid_for_site

Declares the instance of site associated with the grid_of_site. The grid_of_site can have (or belong to) only one site. The grid_of_site cannot exist without a site.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.12 grid_of_structure

Entity definition:

A type of grid that is associated with a structure.

EXPRESS specification:

```
*)
ENTITY grid_of_structure
  SUBTYPE OF (grid);
    grid_for_structure : structure;
END_ENTITY;
(*
```

Attribute definitions:

grid_for_structure

Declares the instance of structure associated with the grid_of_structure. The grid_of_structure can have (or belong to) only one structure. The grid_of_structure cannot exist without a structure.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.13 grid_offset

Entity definition:

Allows a position (of an assembly) to be defined relative to a grid_intersection.

EXPRESS specification:

```
*)
ENTITY grid_offset;
  intersection : grid_intersection;
  offset : LIST [2:3] OF length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

intersection

Declares the instance of grid_intersection associated with the grid_offset. A grid_offset cannot exist without a grid_intersection. The grid_intersection may be associated with zero, one or many grid_offsets.

offset

Specifies the numerical values with a unit of the offsets of the position from the grid_intersection. The offset is defined by 2 or 3 values. In a Cartesian coordinate system these would be the x, y, and z offsets respectively.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.14 grid_orthogonal

Entity definition:

A type of grid that allows the specification of an orthogonal grid (i.e. one where the gridlines are at right angles to each other) by a list of values in the two directions.

EXPRESS specification:

```
*)
ENTITY grid_orthogonal
SUBTYPE OF (grid);
    spacing_1 : LIST [0:?] OF positive_length_measure_with_unit;
    spacing_2 : LIST [0:?] OF positive_length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

spacing_1

The list of numerical values with a unit of the linear spacing of the gridlines in the first direction (measured perpendicular to the gridlines). The list of values associated with spacing_1 allow a sequence of gridlines to be established. References may be repeated. Thus, the first instance in the list gives the spacing from the first to the second gridline, while the second instance in the list gives the spacing from the second to the third gridline. If the spacing is the same, the reference may be repeated as many times as required. For example, the second instance in the list may be the same as the first instance in the list.

spacing_2

The list of numerical values with a unit of the linear spacing of the gridlines in the second direction (measured perpendicular to the gridlines). See also the definition for spacing_1.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.15 grid_radial

Entity definition:

A type of grid that allows the specification of a radial grid by lists of angles. A radial grid is one where the set of gridlines form the spokes of a wheel.

EXPRESS specification:

```
*)
ENTITY grid_radial
SUBTYPE OF (grid);
    spacing_1 : LIST [0:?] OF plane_angle_measure_with_unit;
END_ENTITY;
```

(*)

Attribute definitions:*spacing_1*

The list of numerical values with a unit of the angular spacing of the gridlines that form the ‘spokes of the wheel’.

Informal propositions:

The values of *spacing_1* allow a sequence of gridlines to be established. References may be repeated. The list of values associated with *spacing_1* allow a sequence of gridlines to be established. References may be repeated. Thus, the first instance in the list gives the angle between the first and second gridlines, while the second instance in the list gives the angle from the second to the third gridline. If the spacing is the same, the reference may be repeated. For example, the second instance in the list may be the same as the first instance in the list.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.16 grid_skewed**Entity definition:**

A type of grid that allows the specification of a skewed grid (i.e. one where the gridlines are not at right angles to each other) by a list of values in the two directions, and an angle.

EXPRESS specification:

*)

ENTITY grid_skewed

SUBTYPE OF (grid);

 skew_angle : plane_angle_measure_with_unit;

 spacing_1 : LIST [0:?] OF positive_length_measure_with_unit;

 spacing_2 : LIST [0:?] OF positive_length_measure_with_unit;

END_ENTITY;

(*)

Attribute definitions:*skew_angle*

The numerical value with a unit of the angle between the non-parallel gridlines. The angle is measured from the first set of gridlines (whose spacing is given by *spacing_1*) to the second set of gridlines (whose spacing is given by *spacing_2*), between equivalent axes. (It should be noted that the gridlines are defined by planes and that each plane is defined by three orthogonal axes – the z-axis being normal to the plane.)

spacing_1

The list of numerical values with a unit of the linear spacing of the gridlines in the first direction (measured perpendicular to the gridlines). The list of values associated with *spacing_1* allow a sequence of gridlines to be established. References may be repeated. Thus, the first instance in the list gives the spacing from the first to the second gridline,

while the second instance in the list gives the spacing from the second to the third gridline. If the spacing is the same, the reference may be repeated. For example, the second instance in the list may be the same as the first instance in the list.

spacing_2

The list of numerical values with a unit of the linear spacing of the gridlines in the second direction (measured perpendicular to the gridlines). See also the definition for *spacing_1*.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.17 gridline

Entity definition:

A type of plane that is related to a grid (which is in turn related to a building, a site, or a structure). A gridline is defined as lying in a vertical plane (although nothing in the EXPRESS schema enforces this constraint). The gridline is defined by a plane, which is defined by three orthogonal axes (inherited from the attribute position of the SUPERTYPE elementary_surface). The z-axis lies normal to the plane.

EXPRESS specification:

*)

ENTITY gridline

SUBTYPE OF (plane);

parent_grid : grid;

preceding_line : OPTIONAL gridline;

INVERSE

succeeding_line : SET [0:1] OF gridline FOR preceding_line;

UNIQUE

URG2 : SELFelementary_surface.position;

WHERE

WRG7 : NOT(EXISTS(preceding_line) AND (preceding_line := (SELF)));

END_ENTITY;

(*

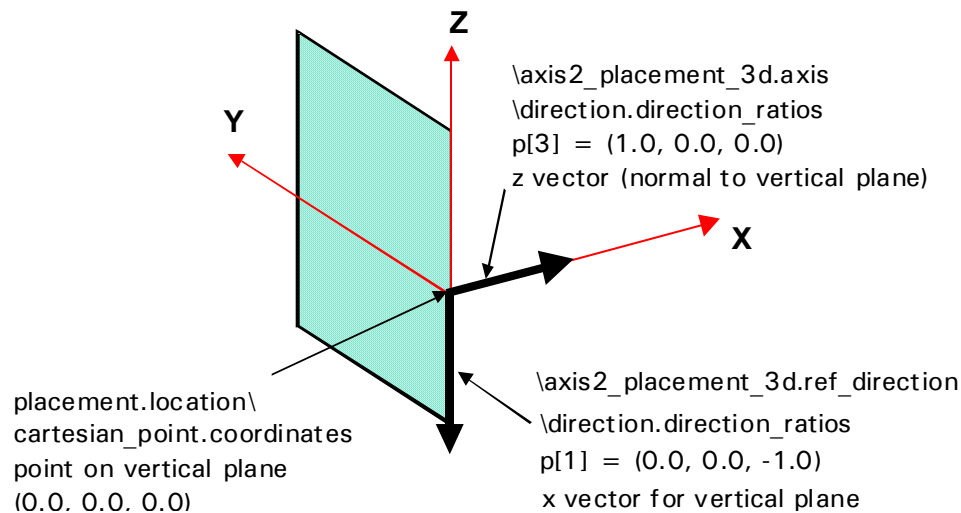


Figure 4.2 Defining a gridline (at the origin of the grid: $x = y = 0$)

Attribute definitions:

parent_grid

Declares the instance of grid that is associated with the gridline. The gridline can have (or belong to) only one parent grid. The gridline cannot exist without a parent grid.

preceding_line

Declares the instance of gridline immediately preceding the gridline (if there is one).

Formal propositions:

succeeding_line

A gridline may only act as a predecessor for (at most) one other gridline.

URG2

The instance of `axis2_placement_3d` (referenced by the attribute position inherited from the SUPERTYPE `elementary_surface`) must be unique to this instance of gridline. In other words, gridlines must not share the same position in 3D Cartesian space.

WRG7

If the attribute `preceding_line` is populated, it shall not be self referencing. In other words, a gridline may not precede itself.

Informal propositions:

A gridline may not be preceded by or succeeded by itself.

The gridline is defined in the vertical plane.

Notes:

Known as GRIDLINE in CIS/1.

The SUPERTYPE entities `plane` and `elementary_surface` are defined in STEP Part 42.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.18 map_location

Entity definition:

A type of geographical_location that adds the position (in terms of Eastings and Northings) to the height above datum it inherits from geographical_location. For example, the School of Civil Engineering at the University of Leeds is located on the Ordnance Survey National Grid at “SE 292 349”. The position is located with respect to a the origin of the grid.

EXPRESS specification:

*)

ENTITY map_location

SUBTYPE OF (geographical_location);

map_name : label;

map_code : identifier;

eastings : length_measure_with_unit;

northings : length_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:

map_name

A short alphanumeric reference for the map (or mapping system) used to define the map_location. For example in the UK the map name could be “OS”.

map_code

A short alphanumeric reference for the grid square on the map sheet used to define the map_location. For example, the University of Leeds School of Civil Engineering lies on the 100km grid square labelled “SE”.

eastings

Declares the first instance of length_measure_with_unit associated with this instance of map_location, which provides the numerical value of the Eastings (in accordance with the unit system and measurement conventions used by mapping system.) The Eastings value is derived from the distance of the position referenced from origin of the grid, and is related to the nearest gridline running North to South. The value given in the map_code allows the position to be located within a local area (e.g. a within a 100km square). For example, for the School of Civil Engineering at the University of Leeds the Eastings value is as quoted “292” (which relates to a dimension of 29.2km within the 100km grid square labelled “SE”).

northings

Declares the second instance of length_measure_with_unit associated with this instance of map_location, which provides the numerical value of the Northings (in accordance with the unit system and measurement conventions used by mapping system.) The Northings value is derived from the distance of the position referenced from origin of the grid, and is related to the nearest gridline running East to West. For example, for the School of Civil Engineering at the University of Leeds the Northings value is quoted as “349” (which relates to a dimension of 34.9km within the 100km grid square labelled “SE”).

Notes:

New for CIS/2.

See diagram 11 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.19 organization_relationship_contractual

Entity definition:

A type of organization_relationship (see STEP Part 41) that allows the relationship between two organizations to be assigned to a contract, a start date and (optionally) an end date.

EXPRESS specification:

*)

ENTITY organization_relationship_contractual

SUBTYPE OF (organization_relationship);

assigned_contract : contract;

effective_start_date : date_and_time;

effective_end_date : OPTIONAL date_and_time;

UNIQUE

URO1 : SELF\organization_relationship.name,
SELF\organization_relationship.relate_organization,
SELF\organization_relationship.related_organization;

WHERE

WRO4 : acyclic_organization_relationship (SELF,
[SELF\organization_relationship.related_organization],
'STRUCTURAL_FRAME_SCHEMA.
ORGANIZATION_RELATIONSHIP_RELATED_ORGANIZATION');

END_ENTITY;

(*

Attribute definitions:

assigned_contract

Declares the instance of contract assigned to the organization_relationship.

effective_start_date

Declares the first instance of date_and_time assigned to the organization_relationship. This specifies the date and time when the organizational relationship came into effect, or is to become valid.

effective_end_date

Declares the second instance of date_and_time assigned to the organizational relationship. This specifies the date and time when the organizational relationship came to an end, or will no longer be effective.

Formal propositions:

URO1

The combination of the name of this organization_relationship and the two instances of organization associated with this organization_relationship must be unique. In other words, the same relationship cannot be declared more than once.

WRO4

The `organization_relationship` must not be cyclic. The STEP Part 41 function `acyclic_organization_relationship` determines whether or not the given organizations have been self-defined by the associations made in the specified `organization_relationship`. The function returns a value of TRUE if none of the elements of the `relatives` argument (populated here as `[SELF\organization_relationship.related_organization]`) occur in the `relation` argument (populated here as `SELF`) of the type which is given in the `specific_relation` argument (populated here as `'STRUCTURAL_FRAME_SCHEMA.ORGANIZATION_RELATIONSHIP_RELATED_ORGANIZATION'`). Otherwise it returns a value of FALSE.

Informal propositions:

An `organization_relationship_contractual` cannot exist without a contract or a start date.

Notes:

New for CIS/2.

The entity `organization_relationship` and the function `acyclic_organization_relationship` are defined in STEP Part 41.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.20 project**Entity definition:**

A type of `structural_frame_item` that allows data about structures, assemblies, parts, etc to be assigned to a group of people. The project must be given a number and a name, by which it is uniquely referenced. It may also be given a text description. It is implied that the type of projects dealt with here are construction projects that involve a building type structural steel frame built on a site. However, LPM/6 does not enforce such restrictions and so this entity could include different types of project.

As a SUBTYPE of `structural_frame_item`, a project inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a project may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A project may also be related to another project (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

```

*)
ENTITY project
SUBTYPE OF (structural_frame_item);
UNIQUE
    URP2 : SELF\structural_frame_item.item_number,
          SELF\structural_frame_item.item_name;
END_ENTITY;
(*)

```

Attribute definitions:*SELF\structural_frame_item.item_number*

An integer reference that provides one half of the unique identification of the project. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_name

A short alphanumeric reference that provides the other half of the unique identification of the project. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_description

A free text description of the project. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

Formal propositions:*URP2*

The combination of the item_name and the item_number (both inherited from the SUPERTYPE structural_frame_item) must be unique to this instance of project. That is, no other project may have the same combination of name and number.

Notes:

Known as PROJECT in CIS/1.

See diagram 58 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.21 project_organization**Entity definition:**

An association between an instance of project and an instance of person_and_organization. The person_and_organization must be assigned a role. The instance of person_and_organization represents a participant in the project, while the instance of person_and_organization_role represents the role that that participant plays; e.g. client, architect, consulting engineer, fabricator, etc. The whole project team would be assigned using a number of instances of project_organization all referencing the same instance of project.

EXPRESS specification:

```

*)
ENTITY project_organization;
    project_participant : person_and_organization;
    related_project : project;

```

```

    role : person_and_organization_role;
END_ENTITY;
(*)

```

Attribute definitions:

project_participant

Declares the instance of person_and_organization associated with this project_organization.

related_project

Declares the instance of project associated with this project_organization.

role

Declares the instance of person_and_organization_role associated with this project_organization. (The entity person_and_organization_role is taken from STEP Part 41.)

Informal propositions:

An instance of project_organization cannot exist without corresponding instances of person_and_organization, project, and person_and_organization_role.

Notes:

New for CIS/2.

The entities person_and_organization and person_and_organization_role are defined in STEP Part 41.

See diagram 58 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.22 project_plan

Entity definition:

A type of structural_frame_item representing a schedule of (work) items that are considered to be required for the project. The project plan could itemize the processes required during the design and/or construction phases of the construction project. The project_plan must be given a number and a name. It may also be given a text description.

As a SUBTYPE of structural_frame_item, a project_plan inherits the three attributes item_number, item_name, and item_description. It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a project_plan may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented)
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A `project_plan` may also be related to another `project_plan` (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

*)

ENTITY `project_plan`

SUBTYPE OF (`structural_frame_item`);

`project_plan_author` : `project_organization`;

`project_plan_date` : `date_and_time`;

`related_project` : `project`;

INVERSE

`items` : SET [1:?] OF `project_plan_item` FOR `item_for_plan`;

UNIQUE

URP3 : SELF\`structural_frame_item.item_number`, `related_project`;

WHERE

WRP25 : `project_plan_author.related_project` := `related_project`;

END_ENTITY;

(*

Attribute definitions:

SELF\structural_frame_item.item_number

An integer reference that uniquely identifies the `project_plan` within a particular project. (This attribute is inherited from the SUPERTYPE `structural_frame_item`.)

SELF\structural_frame_item.item_name

A short alphanumeric reference for the `project_plan`. (This attribute is inherited from the SUPERTYPE `structural_frame_item`.)

SELF\structural_frame_item.item_description

A free text description of the `project_plan`. (This attribute is inherited from the SUPERTYPE `structural_frame_item`.)

project_plan_author

Declares the instance of `project_organization` associated with this `project_plan`. It is implied that the person and organization referenced have acted as the author of the project plan.

project_plan_date

Declares the instance of `date_and_time` associated with this `project_plan`. It is implied that the date and time referenced were when the project plan was created.

related_project

Declares the instance of `project` associated with this `project_plan`. It is implied that the project referenced is the one that the plan has been created for.

Formal propositions:

items

A least one `project_plan_item` must be associated with the `project_plan`. The `project_plan` cannot exist without at least one corresponding instance of `project_plan_item`.

URP3

The combination of the `item_number` (inherited from the SUPERTYPE `structural_frame_item`) and the `related_project` must be unique to this instance of `project_plan`. In other words, within each project, the project plan must have a unique number.

WRP25

The instance of project referenced by the instance of `project_organization` shall be the same instance referenced by the attribute `related_project`. In other words, the author of the project plan must be a team member of the project for which the plan is written.

Notes:

New for CIS/2.

The entity `date_and_time` is defined in STEP Part 41.

See diagram 58 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.23 project_plan_item**Entity definition:**

A type of `structural_frame_item` that provides the definition of a single item of work. The item must be part of a `project_plan`, which in turn is associated with a project. The item must be given a name and number for referencing purposes. The item must also be given a start date, end date and a duration. The item may be given a sequence number - an integer that positions the item in the sequence of items in a project plan.

As a SUBTYPE of `structural_frame_item`, a `project_plan_item` inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a `project_plan_item` may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A `project_plan_item` may also be related to another `project_plan_item` (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`. However, the more constrained entity `project_plan_item_relationship` should be used to represent a one-to-one relationship between two instances of `project_plan_item`.

EXPRESS specification:

*)

ENTITY `project_plan_item`

SUPERTYPE OF (ONEOF(`project_process_item`))

SUBTYPE OF (`structural_frame_item`);

`item_for_plan` : `project_plan`;

```

    start_date : date_and_time;
    end_date : date_and_time;
    item_duration : time_measure_with_unit;
    actors : OPTIONAL SET [1:?] OF project_organization;
    sequence_number : OPTIONAL INTEGER;
    item_status : OPTIONAL label;
UNIQUE
    URP4 : SELF\structural_frame_item.item_number, item_for_plan;
WHERE
    WRP26 : NOT (EXISTS(actors) AND
        (SIZEOF(QUERY(the_project <* actors | the_project.related_project :<>:
            (item_for_plan.related_project)) ) <> 0));
END_ENTITY;
(*)

```

Attribute definitions:

SELF\structural_frame_item.item_number

An integer reference that uniquely identifies the project_plan_item within a particular project_plan. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_name

A short alphanumeric reference for the project_plan_item. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_description

A free text description of the project_plan_item. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

item_for_plan

Declares the instance of project_plan associated with the project_plan_item. It is implied that the item forms part of (and belongs to) this project plan.

start_date

Declares an instance of date_and_time associated with the project_plan_item. It is implied that the date and time referenced are when the item began (or will begin).

end_date

Declares a second instance of date_and_time associated with the project_plan_item. It is implied that the date and time referenced are when the item finished (or will finish).

item_duration

Declares an instance of time_measure_with_unit associated with the project_plan_item. It is implied that the time measure reference is the time taken to complete the item. This may not be simply the difference between the start and end dates, since items may not be continuous. Thus, the item_duration states the total amount of time taken (or anticipated) to complete the item.

actors

Declares the set of one or more instances of project_organization that may be associated with this project_plan_item. This attribute identifies a number of members of the project

team that are collectively responsible for the item of work scheduled. Each person_and_organization referenced here must play a role on the associated project.

sequence_number

An integer reference that positions the project_plan_item in the sequence of items in a project_plan. For example, the first, second and third items in a sequence of a plan would be given the sequence numbers 1, 2, and 3, respectively. (The sequence number may differ from the item_number.)

item_status

A short text reference that provides the status of the item; e.g. 'completed', 'planned', 'postponed', 'cancelled'.

Formal propositions:

URP4

The combination of the item_number (inherited from the SUPERTYPE structural_frame_item) and the item_for_plan must be unique to this instance of project_plan_item. In other words, within each project plan, each item must have a unique number.

WRP26

If the attribute actors is populated, then the instance of project referenced by each member of the set of instances of project_organization shall be the same instance referenced by the related project_plan. In other words, the actors on the project plan item must all be team members of the project for which the plan is written.

Notes:

New for CIS/2.

The entities date_and_time and time_measure_with_unit are defined in STEP Part 41.

See diagram 58 of the EXPRESS-G diagrams in Appendix B.

4.3.24 project_plan_item_relationship

Entity definition:

An association between two distinct project_plan_items. The relationship must be given a name, and may also be given a description.

EXPRESS specification:

*)

```
ENTITY project_plan_item_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    related_plan_item : project_plan_item;
    relating_plan_item : project_plan_item;
WHERE
    WRP24 : related_plan_item :<>: relating_plan_item;
END_ENTITY;
(*
```


Attribute definitions:

relationship_name

A short alphanumeric reference for the project_plan_item_relationship.

relationship_description

A free text description for the project_plan_item_relationship.

related_plan_item

Declares the first instance of project_plan_item associated with this project_plan_item_relationship, and therefore related to the second project_plan_item.

relating_plan_item

Declares the second instance of project_plan_item associated with this project_plan_item_relationship, and therefore related to the first project_plan_item.

Formal propositions:

WRP24

The related_plan_item must reference a difference instance of project_plan_item from the instance referenced by relating_plan_item. That is, a project_plan_item cannot be related to itself.

Informal propositions:

A project_plan_item can be associated with (i.e. involved in) zero, one or many relationships (through the entity project_plan_item_relationship) and therefore can be related to any number of other project_plan_items.

Notes:

New for CIS/2.

See diagram 10 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.25 setting_out_point

Entity definition:

An association between a site and a geographical location. A site may have any number of setting out points (i.e. positions on the site used for locating and controlling the co-ordination of structural frames, parts, etc), but the instance of setting_out_point is defined for only one (instance of) site.

EXPRESS specification:

```
*)
ENTITY setting_out_point;
    set_out_site : site;
    location : geographical_location;
END_ENTITY;
(*
```

Attribute definitions:

set_out_site

Declares the instance of site associated with the setting_out_point.

location

Declares the instance of `geographical_location` associated with the `setting_out_point`. It is implied that `geographical_location` referenced provides the location in a global referencing system for the point on the site for the purposes of setting out.

Informal Propositions

The `setting_out_point` cannot exist without corresponding instances of `site` and `geographical_location`.

Notes:

New for CIS/2.

See diagram 11 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.26 site**Entity definition:**

A type of `structural_frame_item` that represents a geographical area upon which one or more buildings will be (or have been) permanently or temporarily erected. A site may be given a shape via its SUBTYPE, `site_with_shape`. The site must be given a name and a number, and may also be given a description and an address. It is implied that the site is a construction site, although it could be a place of storage such as a marshalling yard.

As a SUBTYPE of `structural_frame_item`, a site inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a site may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A site may also be related to another site (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

*)

ENTITY site

SUPERTYPE OF (`site_with_shape`)

SUBTYPE OF (`structural_frame_item`);

site_address : OPTIONAL address;

UNIQUE

URS5 : SELF\structural_frame_item.item_number,
SELF\structural_frame_item.item_name;

END_ENTITY;

(*

Attribute definitions:*SELF\structural_frame_item.item_number*

An integer reference that provides one half of the unique identification of the site. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_name

A short alphanumeric reference that provides the other half of the unique identification of the site. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_description

A free text description of the site. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

site_address

Declares the instance of address associated with the site. It is implied that the instance of address (see STEP Part 41) referenced provides the postal location of the construction site.

Formal propositions:*URS5*

The combination of the item_name and the item_number (both inherited from the SUPERTYPE structural_frame_item) must be unique to this instance of site. That is, no other site may have the same combination of name and number.

Notes:

Known as SITE in CIS/1.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.27 site_with_shape**Entity definition:**

A type of site that provides the site with a shape, through its reference to a shape_representation_with_units (a SUBTYPE of the STEP Part 41 entity shape_representation). The shape could be made up of one or more geometrical_representation_items or topological_representation_items (constructs taken from STEP Part 42). The shape_representation_with_units may be used to represent the boundary (in which case the items would be curves), or the topography of the site (in which case the items would be surfaces).

EXPRESS specification:

*)

ENTITY site_with_shape

SUBTYPE OF (site);

 shape : shape_representation_with_units;

END_ENTITY;

(*

Attribute definitions:*shape*

Declares the instance of `shape_representation_with_units` associated with this `site_with_shape`. It is implied that the instance of `shape_representation_with_units` referenced defines the shape of the site.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

4.3.28 structure**Entity definition:**

A type of `structural_frame_item` representing an independent structural framework. Situated on a site, and forming part of a project, the structure is the key entity to which many other significant CIS entities relate.

From the perspective of either the decomposition or composition view, a structure can be considered as a hierarchical arrangement of assemblies. The structure corresponds to the 'root' of both the decomposition and composition hierarchies. The 'leaves' of the decomposition and composition hierarchies are (respectively) the `design_parts` and `located_parts`. A physical structure is ultimately composed of connected `located_parts`, with their respective `located_features`, and `located_joint_systems`. The structure must be given a name and may be given a description. The structure is located through the entity `located_structure`.

It is implied that the structure is a steel frame of the type used in building construction, although the LPM does not enforce such a restrictions on the use of the entity structure.

As a SUBTYPE of `structural_frame_item`, a structure inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a structure may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A structure may also be related to another building (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

*)

ENTITY structure

SUBTYPE OF (`structural_frame_item`);

UNIQUE

URS6 : SELF\structural_frame_item.item_number,
SELF\structural_frame_item.item_name;

END_ENTITY;

(*

Attribute definitions:

SELF\structural_frame_item.item_number

An integer reference that provides one half of the unique identification of the structure. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_name

A short alphanumeric reference that provides the other half of the unique identification of the structure. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

SELF\structural_frame_item.item_description

A free text description of the structure. (This attribute is inherited from the SUPERTYPE structural_frame_item.)

Formal propositions:

URS6

The combination of the item_name and the item_number (both inherited from the SUPERTYPE structural_frame_item) must be unique to this instance of structure. That is, no other structure may have the same combination of name and number.

Notes:

Known as STRUCTURE in CIS/1.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.29 zone

Entity definition:

An abstract concept that allows a building, a site or a structure to be divided into organizational sub sections. The association to a building, a site or a structure is made through the SUBTYPES of zone. The zones are not geometrical. The zones may overlap. The zone may be bounded (through the SUBTYPE zone_bounded). A building, a site or a structure cannot use the same zone as the zones are defined individually. The zone must have a name and may also have a description.

EXPRESS specification:

*)

ENTITY zone

ABSTRACT SUPERTYPE OF (ONEOF

 (zone_of_building,

 zone_of_project,

 zone_of_site,

 zone_of_structure) ANDOR

 zone_bounded);

 zone_name : label;

 zone_description : OPTIONAL text;

END_ENTITY;

(*

Attribute definitions:*zone_name*

A short human-readable alphanumeric reference for the zone; e.g. 'Phase 1a'

zone_description

A free text description of the zone.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.30 zone_bounded**Entity definition:**

A type of zone that is associated with upto four gridlines and upto two grid levels. The gridlines and the grid levels define the boundaries of the zone.

EXPRESS specification:

*)

ENTITY zone_bounded

SUBTYPE OF (zone);

 bounding_gridlines : SET [2:4] OF gridline;

 bounding_levels : OPTIONAL SET [1:2] OF grid_level;

DERIVE

 bounding_grid : grid := bounding_gridlines[1].parent_grid;

WHERE

 WRZ1 : SIZEOF(QUERY(line <* bounding_gridlines | line.parent_grid :<>:
 (bounding_grid))) = 0;

 WRZ2 : NOT (EXISTS(bounding_levels) AND (SIZEOF(QUERY(level <* bounding_levels
 | level.parent_grid :<>: (bounding_grid))) <> 0));

END_ENTITY;

(*

Attribute definitions:*bounding_gridlines*

Declares the set of instances of gridline associated with the instance of zone_bounded, which define the boundaries of the zone on plan. There must be at least two instances of gridline associated with the zone_bounded, but there may be upto four. The SET data

type ensures that each of the four instances are unique (repetition is not allowed). If provided, the third and fourth instances of gridline must not be parallel to the first or second instances of gridline referenced in the set. All of the gridlines belong to the same parent grid (see definition of the WHERE rules).

If only two instances are populated in the set then the zone is unbounded in one direction.

bounding_levels

Declares the set of instances of grid_level that may be associated with the instance of zone_bounded to define the lower and upper horizontal planes bounding the zone. The SET_data type ensures that each of the two instances are unique (repetition is not allowed). If provided, the two instances of grid_level are parallel to each other and are defined in the horizontal plane (see definition of grid_level). Both grid_levels belong to the same parent grid (see definition of the WHERE rules).

If this attribute is not populated, then the zone is unbounded vertically, that is, the only boundaries are the vertical planes defined by the gridlines.

bounding_grid

The attribute derives the instance of grid that acts as the parent to the gridlines. The instance is derived from the grid that is referenced by the first instance of gridline in the set of gridlines captured by the attribute bounding_gridlines. The zone is defined within this grid.

This derived attribute does not appear in the STEP Part 21 file.

Formal propositions:

DERIVE

The derived attribute bounding_grid does not appear in the STEP Part 21 file.

WRZ1

The size of the set of instances of grid that act as the parent_grid to each instance of gridline in the set captured by the attribute bounding_gridlines that are different from the instance of grid derived by the attribute bounding_grid shall equal zero. In other words, all of the gridlines belong to the same parent grid.

WRZ2

If the attribute bounding_levels is populated, then the size of the set of instances of grid that act as the parent_grid to each instance of grid_level in the set captured by the attribute bounding_levels that are different from the instance of grid derived by the attribute bounding_grid shall equal zero. In other words, any grid_level provided must belong to the same parent grid as the gridlines.

Notes

New for CIS/2.

See Diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.31 zone_of_building

Entity definition:

A type of zone that allows a zone to be associated with a building. This entity is intended to represent the zone of a building (and not a building in a zone).

EXPRESS specification:

```

*)
ENTITY zone_of_building
SUPERTYPE OF (zone_of_building_storey)
SUBTYPE OF (zone);
    zone_for_building : building;
END_ENTITY;
(*

```

Attribute definitions:*zone_for_building*

Declares the instance of building associated with the zone_of_building. It is implied that the zone_of_building provides the name and description of a zone of the building referenced.

Informal propositions:

If the building zone is bounded, then the zone_of_building is instanced with the sibling SUBTYPE entity zone_bounded. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.32 zone_of_building_storey**Entity definition:**

A type of zone_of_building that allows a zone to be associated with a building and given a name and purpose. The storey of a building is an organizational, rather than geometric, notion. It is not the same as a grid_level. This entity is intended to represent a storey of a building (e.g. “the third floor”) and not a zone of a storey of a building. It is particular type of zone within a building. The zone_name (inherited from the SUPERTYPE zone) should reflect the name of the storey (e.g. “Third Floor”). The zone_description (also inherited from the SUPERTYPE zone) should provide the more detailed description of the storey (e.g. “Third Floor – extending from finished floor level of the third floor to underside of structural slab of the fourth floor”).

EXPRESS specification:

```

*)
ENTITY zone_of_building_storey
SUBTYPE OF (zone_of_building);
    storey_height : positive_length_measure_with_unit;
    storey_level : OPTIONAL length_measure_with_unit;
    datum_name : OPTIONAL text;
END_ENTITY;
(*

```


Attribute definitions:*storey_height*

Declares the instance of `positive_length_measure_with_unit` associated with the `zone_of_building_storey`. It is implied that `length_measure_with_unit` referenced provides the nominal height of a storey of a building; e.g. the floor to floor height.

storey_level

Declares the instance of `length_measure_with_unit` associated with the `zone_of_building_storey`. It is implied that `length_measure_with_unit` referenced provides the nominal height to the bottom of the building storey above a given datum. The height may be measured to (for example) the structural slab level, or the finished floor level.

datum_name

A free text reference for the datum from which the storey level has been given (e.g. “Ground Floor finished floor level”, or “505m AOD”)

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.33 zone_of_project**Entity definition:**

A type of zone that allows a zone to be associated with a project. This is an abstract concept that allows a project to be divided into zones (e.g. “Phase 1”, “Phase 2”). A project can involve several sites, each could be considered to be a separate zone of the project.

EXPRESS specification:

```
*)
ENTITY zone_of_project
  SUBTYPE OF (zone);
    zone_for_project : project;
END_ENTITY;
(*
```

Attribute definitions:*zone_for_project*

Declares the instance of `project` associated with the `zone_of_project`. It is implied that the `zone_of_project` provides the name and description of a zone of the project referenced.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.34 zone_of_site

Entity definition:

A type of zone that allows a zone to be associated with a site. This allows the site to be divided into various sub-sections. The zone_name (inherited from the SUPERTYPE zone) should reflect the name of the zone of the site. A structure may be located within a particular zone of a site (see located_structure).

EXPRESS specification:

```
*)
ENTITY zone_of_site
  SUBTYPE OF (zone);
    zone_for_site : site;
END_ENTITY;
(*
```

Attribute definitions:

zone_for_site

Declares the instance of site associated with the zone_of_site. It is implied that the zone_of_site provides the name and description of a zone of the site referenced.

Informal propositions:

If the site zone is bounded, then the zone_of_site is instanced with the sibling SUBTYPE entity zone_bounded. The use of the AND/OR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

4.3.35 zone_of_structure

Entity definition:

A type of zone that allows a zone to be associated with a structure. This allows a structure to be divided into sub-sections. The zone_name (inherited from the SUPERTYPE zone) should reflect the name of the zone of the structure. An assembly may be located within a particular zone of a structure (see located_assembly).

EXPRESS specification:

```
*)
ENTITY zone_of_structure
  SUPERTYPE OF (ONEOF(zone_of_structure_sequence))
  SUBTYPE OF (zone);
    zone_for_structure : structure;
END_ENTITY;
(*
```

Attribute definitions:**zone_for_structure**

Declares the instance of structure associated with the zone_of_structure. It is implied that the zone_of_structure provides the name and description of a zone of the structure referenced.

Informal propositions:

If the structure zone is bounded, then the zone_of_structure is instanced with the sibling SUBTYPE entity zone_bounded. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2.

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition (subtype added).

4.3.36 zone_of_structure_sequence**Entity definition:**

A type of zone_of_structure that is associated with a higher level zone_of_structure. This allows a structure to be divided into sub-sections, known as sequences. The zone_name (inherited from the SUPERTYPE zone) should reflect the name of the sequence. An assembly may be located within a particular zone of a structure, and hence within a sequence (see located_assembly).

Sequence is another high-level aggregation that decomposes a manufacturing zone into multiple subsections. The sequence is related to the erection of steel assemblies. The sequence is valuable to erectors for scheduling crane works. A sequence may be delivered in multiple lots to the site, and a sequence should indicate information to reference its parent zone.

EXPRESS specification:

*)

ENTITY zone_of_structure_sequence

SUPERTYPE OF (ONEOF(zone_of_structure_sequence_lot))

SUBTYPE OF (zone_of_structure);

parent_zone : zone_of_structure;

DERIVE

lots : SET[0:?] OF zone_of_structure_sequence_lot := bag_to_set
(USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
'ZONE_OF_STRUCTURE_SEQUENCE_LOT.' +
'PARENT_SEQUENCE'));

assemblies : SET[1:?] OF located_assembly := bag_to_set
(USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.
LOCATED_ASSEMBLY.PARENT_STRUCTURE'));

WHERE

WRZ3 : parent_zone :<>: (SELF);

WRZ4 : NOT ('STRUCTURAL_FRAME_SCHEMA.
ZONE_OF_STRUCTURE_SEQUENCE_LOT' IN
TYPE OF(parent_zone));

END_ENTITY;

(*

Attribute definitions:

parent_zone

Declares the instance of zone_of_structure associated with the zone_of_structure_sequence. It is implied that the zone_of_structure_sequence is a subdivision of (and belongs to) the zone referenced here.

lots

Derives the set of instances of zone_of_structure_sequence_lot that use this sequence as their parent zone.

assemblies

Derives the set of instances of located_assembly that use this sequence as their parent structure to define their location.

Formal propositions:

WRZ3

A sequence cannot be its own parent.

WRZ4

A sequence cannot have a lot as its parent

Notes:

New for CIS/2 2nd Edition

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

4.3.37 zone_of_structure_sequence_lot

Entity definition:

A type of zone_of_structure_sequence that is associated with a higher level zone_of_structure_sequence. This allows a structure to be divided into sub-sections, known as lots. The zone_name (inherited from the SUPERTYPE zone) should reflect the name of the lot. An assembly may be located within a particular zone of a structure, and hence within a lot (see located_assembly).

Lot is another aggregation that decomposes a sequence to several lots for delivering purpose. A sequence is subdivided into lots that are based on weights for truck loads. A lot may be composed by multiple located assemblies. Thus, the highest-level located_assembly instance should indicate its parent lot for delivery.

EXPRESS specification:

*)

ENTITY zone_of_structure_sequence_lot

SUBTYPE OF (zone_of_structure_sequence);

DERIVE

parent_sequence : zone_of_structure :=

SELFzone_of_structure_sequence.parent_zone;

WHERE

WRZ5 : 'STRUCTURAL_FRAME_SCHEMA.ZONE_OF_STRUCTURE_SEQUENCE' IN
TYPE OF(parent_sequence);

```
WRZ6 : SIZEOF (SELF\zone_of_structure_sequence.lots) = 0;  
WRZ7 : (SELF\zone_of_structure_sequence.parent_zone) :<>: (SELF);  
END_ENTITY;  
(*
```

Attribute definitions:

parent_sequence

Derives the reference to an instance of zone_of_structure from the attribute parent_zone inherited from the zone_of_structure_sequence associated with the zone_of_structure_sequence_lot. It is implied that the zone_of_structure_sequence_lot is a subdivision of (and belongs to) the zone referenced here.

Formal propositions:

WRZ5

The parent sequence referenced by the attribute parent_sequence shall be of the type zone_of_structure_sequence. In other words, a lot must belong to a sequence, rather than being directly related to a zone.

WRZ6

A lot cannot be sub-divided into other lots.

WRZ7

A lot cannot be its own parent.

Notes:

New for CIS/2 2nd Edition

See diagram 9 of the EXPRESS-G diagrams in Appendix B.

5 LPM/6 STRUCTURAL MODELLING

5.1 Structural Modelling concepts and assumptions

5.1.1 Conceptual overview

For analytical purposes, the structure is represented using analysis models. These are made up of nodes and elements. The elements are categorized by their dimensionality as either point elements (zero-dimensional), curve elements (one-dimensional), surface elements (two-dimensional) or volume elements (three-dimensional). Elements may be represented using implicit or explicit geometry. The structure - in the form of the analysis model - may be restrained at the nodes. These boundary conditions may represent a pinned or fixed support, or a (linear or non-linear) spring. The elements may be eccentric from their connecting nodes. The ends of the elements may be released (as a pinned end, a linear spring or non-linear spring) or remain rigid.

A number of different static and dynamic analysis methods are supported. An additional complication is that an element may be defined with or without a material. A complex ANDOR SUPERTYPE accommodates this in LPM/6.

A simple analysis model is shown in Figure 5.1, while the sign convention for the structural analysis model is shown in Figure 5.2.

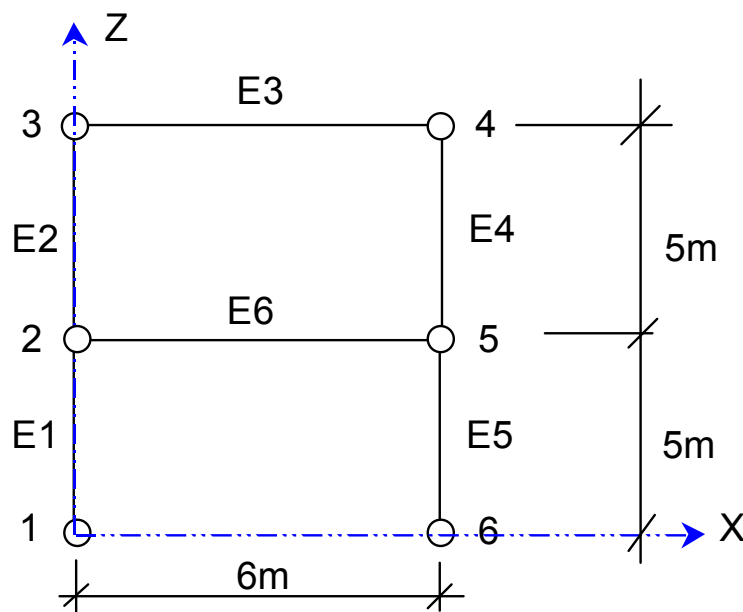


Figure 5.1 *An example of a simple 2D analysis model*

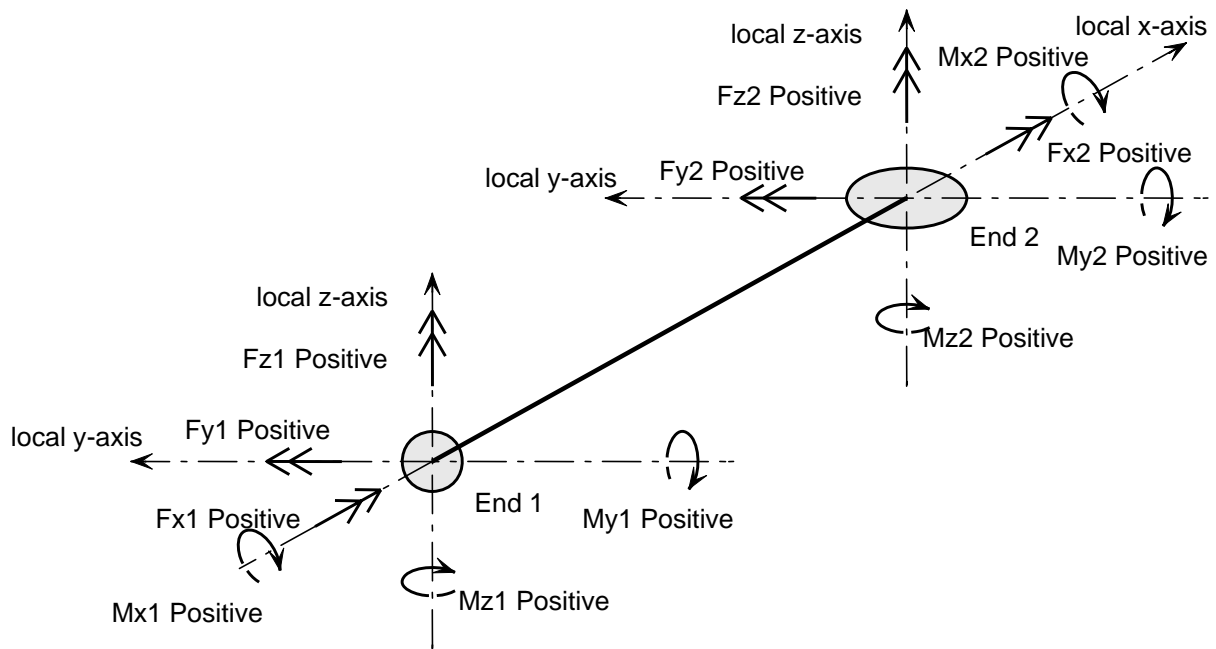


Figure 5.2 The sign conventions used in LPM/6

5.2 Structural Modelling type definitions

The following types have been modified for the 2nd Edition of CIS/2:

- element_volume_shape

5.2.1 dynamic_analysis_type

Type definition:

Classifies the type of dynamic analysis by the assumptions made for the method of analysis. CIS/2 defines five categories of dynamic analysis as follows:

- 1) **free_vibration** - the structure is analysed to determine the natural frequencies and corresponding mode shapes of vibration. This information is needed, for example, in seismic analysis. One or more of the following assumptions may be made:
 - a) Stiffness and inertial effects (or mass properties) of the structure are time independent and hence the free vibration motion is simple harmonic.
 - b) Damping effects are ignored.
 - c) No forces, displacements, pressures or temperature effects are applied to the structure.
- 2) **stressed_free_vibration** - the structure is analysed to determine the natural frequencies and corresponding mode shapes of vibration while taking into consideration membrane effects due to time-independent loads. One or more of the following assumptions may be made:
 - a) Stiffness and inertial effects of the structure are constant.
 - b) Damping effects are ignored.
 - c) No time-dependent forces, displacements, pressures or temperature effects are applied to the structure.

- d) Time-independent forces, displacements, pressures or temperature effects may be applied to the structure.
- 3) `damped_vibration` - the structure is analysed to determine the natural frequencies and corresponding mode shapes of vibration while taking into consideration damping effects. One or more of the following assumptions may be made:
 - a) Stiffness and inertial effects of the structure are constant.
 - b) Damping effects are included.
 - c) No time-dependent forces, displacements, pressures or temperature effects are applied to the structure.
 - d) Time-independent forces, displacements, pressures or temperature effects may be applied to the structure.
- 4) `linear_dynamic` - the structure is analysed to determine its response to arbitrarily time-varying loads. One or more of the following assumptions may be made:
 - a) Stiffness and inertial effects of the structure do not depend on time.
 - b) Initial conditions are known.
 - c) No gyroscopic or Coriolis effects are included.
- 5) `response_spectrum` - the structure is analysed to determine the stresses due to a response spectrum. One or more of the following assumptions may be made:
 - a) The structure is linear-elastic.
 - b) The structure is excited at all support nodes by the same random spectrum.
 - c) The spectrum's direction and frequency content are known.

EXPRESS specification:

```

*)
TYPE dynamic_analysis_type
= ENUMERATION OF
  (free_vibration,
   stressed_free_vibration,
   damped_vibration,
   linear_dynamic,
   response_spectrum,
   undefined);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

5.2.2 element_surface_shape**Type definition:**

Classifies the type of simple surface element as either a quadrilateral or a triangle. The shape and extent of the simple surface element is then determined by the number of connectivities that are associated with the element; that is, there must be 3 connectivities for a triangular element and 4 for a quadrilateral element. The shape is not necessarily a regular polygon.

EXPRESS specification:

```

*)
TYPE element_surface_shape
= ENUMERATION OF
    (quadrilateral, triangle);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

5.2.3 element_volume_shape**Type definition:**

Classifies the type of simple volume element as either a hexahedron, a wedge, a tetrahedron, or a pyramid. A cube is a special case of a hexahedron. The shape and extent of the simple volume element is then determined by the number of connectivities that are associated with the element as follows:

- 1) hexahedron - 6 faces, with at least 5 vertices and 5 connectivities.
(This includes a cube – 6 faces, 8 vertices, 8 connectivities.
- 3) wedge - 5 faces, 6 vertices, 6 connectivities.
- 4) pyramid - 5 faces, 5 vertices, 5 connectivities
- 5) tetrahedron - 4 faces, 4 vertices, 4 connectivities.

These shapes are illustrated in Figure 5.3.

The connectivities define the vertices of the element. The edges of the element connect the vertices and the faces are then defined by the edges. The shape is not necessarily a regular polyhedron. The faces are not necessarily plane: the definition of the positions of the vertices may give rise to warped faces.

In the simplest cases, the polyhedron will be defined by the minimum number of vertices and the faces will be plane.

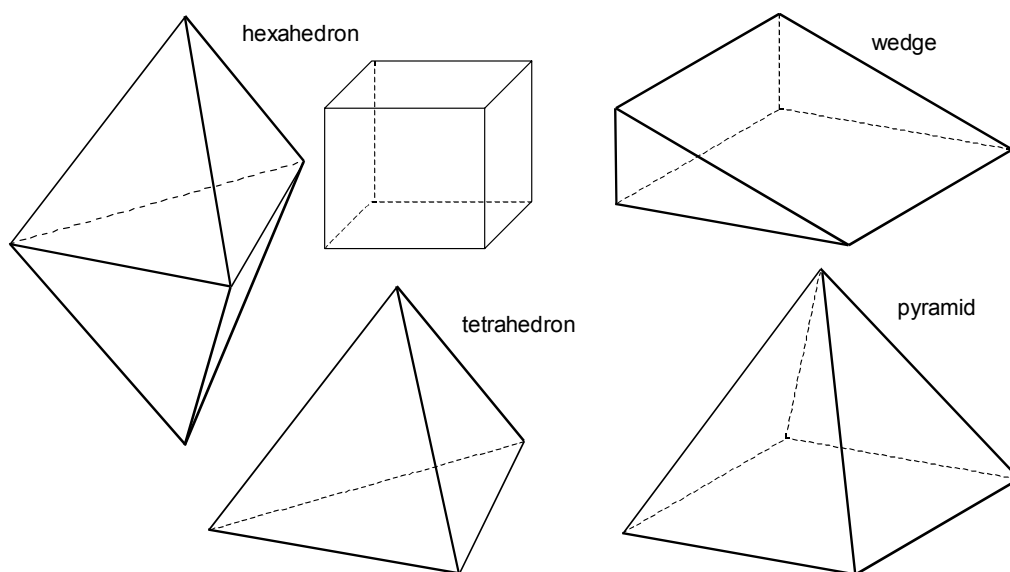


Figure 5.3 *Types of simple volume elements*

EXPRESS specification:

```

*)
TYPE element_volume_shape
= ENUMERATION OF
    (hexahedron_element,
     wedge_element,
     tetrahedron_element,
     pyramid_element);
END_TYPE;
(*

```

Notes:

New for CIS/2. Modified for 2nd Edition (enumeration names clashed with names of STEP Part 42 entites).

5.2.4 orientation_select**Type definition:**

Classifies the type of orientation as being defined by either an instance of plane_angle_measure_with_unit or an instance of direction. This is used with the entities element_curve_simple and element_curve_complex to declare which type of orientation is being used to define the finite_element. That is, the orientations of an element_curve_complex and an element_curve_simple may be defined as either a 'simple beta angle' or an 'orientation vector' made up of three real numbers.

EXPRESS specification:

```

*)
TYPE orientation_select
= SELECT
    (plane_angle_measure_with_unit, direction);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

5.2.5 part_select**Type definition:**

Classifies the type of part as either a (generic) part, a design_part, or located_part. This is used with the element_mapping entity to declare which type of part the finite_element is mapped onto.

EXPRESS specification:

```

*)
TYPE part_select
= SELECT
    (part, design_part, located_part);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

5.2.6 plane_stress_or_strain**Type definition:**

Classifies the type of assumption made in the analysis of simple surface elements as either plane stress or plane strain. Here, plane stress means a linear variation of stress across the plane of an element; i.e. plastic deformation is not occurring. Plane strain means a linear variation of strain across the plane of an element; i.e. the stress variation may not be linear in plastic deformation is occurring.

EXPRESS specification:

```
*)
TYPE plane_stress_or_strain
= ENUMERATION OF
  (plane_stress, plane_strain, undefined);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

5.2.7 static_analysis_type**Type definition:**

Classifies the method of static analysis used by the assumptions made. This enumeration may be used by some applications as an initializer. (Used in entity analysis_model).

EXPRESS specification:

```
*)
TYPE static_analysis_type
= ENUMERATION OF
  (elastic_1st_order,
   elastic_2nd_order,
   rigid_plastic,
   elasto_plastic,
   elastic_perfectly_plastic,
   undefined);
END_TYPE;
(*
```

Notes:

Known as ansys_type in CIS/1. Unchanged in 2nd Edition.

5.3 Structural Modelling entity definitions

No entities have been added or modified for the 2nd Edition in this subject area.

5.3.1 analysis_method

Entity definition:

Provides a description of the important information on the method of structural analysis used. The analysis_method must be given a name (by which it is referenced) and may be given a description. More detailed information will be given in the SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY analysis_method
SUPERTYPE OF (ONEOF
    (analysis_method_dynamic,
    analysis_method_pseudo_dynamic,
    analysis_method_static) ANDOR
    analysis_method_documented);
analysis_name : label;
analysis_assumptions : OPTIONAL text;
END_ENTITY;
(*
```

Attribute definitions:

analysis_name

A short alphanumeric reference for the analysis_method.

analysis_assumptions

A free text description of the assumptions made for the analysis_method.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPEs. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

Known as ANALYSIS in CIS/1. SUBTYPEs added.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.2 analysis_method_documented

Entity definition:

A type of analysis_method that is associated with a part of a document through a reference to an instance of document_usage_constraint. This allows the analysis method to be given by external document references, such as a clause of a national standard.

EXPRESS specification:

*)
 ENTITY analysis_method_documented
 SUBTYPE OF (analysis_method);
 documented_constraints : SET [1:?] OF document_usage_constraint;
 END_ENTITY;
 (*

Attribute definitions:*documented_constraints*

Declares the set of instances of document_usage_constraint (a STEP Part 41 entity) associated with this instance of analysis_method_documented. There must be at least one instance of document_usage_constraint referenced by each instance of analysis_method_documented. Each instance in the set must be different. A particular instance of document_usage_constraint may be referenced by several instances of analysis_method_documented. The instance of document_usage_constraint may describe, for example, a sub-section of a code of practice or a clause within a national standard. The attribute subject_element_value of the entity document_usage_constraint provides the text that formally specifies one aspect of the analysis method. This expands on the attribute analysis_assumptions (inherited from the SUPERTYPE analysis_method)

Notes

New for CIS/2.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.3 analysis_method_dynamic**Entity definition:**

A type of analysis_method that allows additional information to be given about the method of dynamic analysis used. (See also the definition of the type dynamic_analysis_type.)

EXPRESS specification:

*)
 ENTITY analysis_method_dynamic
 SUBTYPE OF (analysis_method);
 analysis_type : dynamic_analysis_type;
 END_ENTITY;
 (*

Attribute definitions:*analysis_type*

Specifies the type of assumptions made for the analysis_method_dynamic. (See also the type definition dynamic_analysis_type.)

Informal propositions:

If the constraints, limitations or assumptions made for the method of the dynamic analysis are documented in national standards, codes or practice or design guides, then the analysis_method_dynamic is instanced with the sibling SUBTYPE entity

analysis_method_documented. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.4 analysis_method_pseudo_dynamic

Entity definition:

A type of analysis_method that allows additional information to be given about the method of pseudo dynamic analysis used. A pseudo dynamic method of analysis uses static analysis methods to solve dynamic analysis problems (e.g. additional loads used to represent seismic actions).

EXPRESS specification:

```
*)
ENTITY analysis_method_pseudo_dynamic
  SUBTYPE OF (analysis_method);
    analysis_type : label;
END_ENTITY;
(*
```

Attribute definitions:

analysis_type

A free text description of the method of pseudo dynamic analysis used.

Informal propositions:

If the constraints, limitations or assumptions made for the method of the pseudo dynamic analysis are documented in national standards, codes or practice or design guides, then the analysis_method_pseudo_dynamic is instanced with the sibling SUBTYPE entity analysis_method_documented. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.5 analysis_method_static

Entity definition:

A type of analysis_method, allowing additional information to be given about the method of static analysis used.

EXPRESS specification:

```
*)
ENTITY analysis_method_static
  SUBTYPE OF (analysis_method);
    analysis_type : static_analysis_type;
```

END_ENTITY;
(*

Attribute definitions:

analysis_type

Specifies the type of static analysis used by the assumptions made in accordance with the appropriate standard (e.g. EC3 in Europe); i.e. Elastic, Plastic, 1st Order, 2nd Order, etc. This states a basic assumption used during analysis (and possibly the subsequent design).

Informal propositions:

If the constraints, limitations or assumptions made for the method of the dynamic analysis are documented in national standards, codes or practice or design guides, then the *analysis_method_static* is instanced with the sibling SUBTYPE entity *analysis_method_documented*. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2, but covered by ANALYSIS in CIS/1.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.6 analysis_model

Entity definition:

A representation of part of the structure (or assembly) for structural analysis. An analysis model is made up of nodes (points in space) and elements. For example, one may wish to analyse a frame with pinned bases, then re-analyse a frame with fixed bases. It is possible to hold both analytical models using the CIS.

An *analysis_model* may be two or three-dimensional, as defined by the SUBTYPEs *analysis_model_2D* and *analysis_model_3D* (respectively).

An analysis model may be associated with a coordinate system (through the SUBTYPE *analysis_model_located*), which in turn may be related to a parent or child coordinate system. Where a coordinate system is not explicitly provided, it is implied that the analysis model has a single global coordinate system with an origin (0.0, 0.0, 0.0) that is not located in any other coordinate system; i.e. the analysis model is isolated.

An analysis model may be child of another analysis model through the SUBTYPE *analysis_model_child*. A child analysis model can also be a parent of another child analysis model. These constructs allow the analysis model to be decomposed into other analysis models. The decomposition is hierarchical; i.e. an *analysis_model_child* can only have one parent.

An analysis model may be associated with any other analysis model through the *analysis_model_relationship*.

An analysis model must be given a name (by which it is referenced) and may be given a description.

EXPRESS specification:

*)
ENTITY *analysis_model*
SUPERTYPE OF (ONEOF

```

        (analysis_model_2D,
        analysis_model_3D) ANDOR
        analysis_model_located ANDOR
        analysis_model_child);
model_name : label;
model_description : OPTIONAL text;
model_type : frame_type;
method_of_analysis : OPTIONAL analysis_method;
coordinate_space_dimension : dimension_count;
INVERSE
    component_elements : SET [1:?] OF element FOR parent_model;
    component_nodes : SET [2:?] OF node FOR parent_model;
END_ENTITY;
(*)

```

Attribute definitions:*model_name*

A short human-readable alphanumeric reference for the analysis_model.

model_description

A free text description of the analysis_model.

model_type

Specifies the type of framing assumed for analysis for the analysis_model. (See type definition for frame_type.)

method_of_analysis

Declares the instance of analysis_method associated with the analysis_model. Each analysis_model may have only one analysis_method. When the same analysis_model is analysed several times using different methods of analysis this is done by recreating the analysis_model as a new instance and making the association between them through the analysis_model_relationship.

coordinate_space_dimension

A integer value of the dimensionality of the analysis_model; i.e. whether 2-dimensional or 3-dimensional.

Formal propositions:**ANDOR SUPERTYPE**

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPEs. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

component_elements

A least one instance of element must be associated with the analysis_model; i.e. the analysis_model cannot exist without at least one element.

The dimensionality of the elements that use this analysis model as its parent must be suitable for the dimensionality of the analysis model as given by the attribute coordinate_space_dimension. (See the definition for the entity element.)

component_nodes

A least two distinct instances of node must be associated with the analysis_model; i.e. the analysis_model cannot exist without at least two nodes.

The dimensionality of the defining points of the node that use this analysis model as its parent must be suitable for the dimensionality of the analysis model as given by the attribute coordinate_space_dimension. (See the definition for the entity node.)

Notes:

Known as ANALYSIS_MODEL in CIS/1.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.7 analysis_model_2D**Entity definition:**

A type of analysis_model that represents part of the structure or assembly in two-dimensional space. This entity is used to represent a plane frame, a plane truss or a grillage for analysis purposes.

EXPRESS specification:

*)

ENTITY analysis_model_2D

SUBTYPE OF (analysis_model);

WHERE

WRA2 : SELF\analysis_model.coordinate_space_dimension = 2;

WRA3 : (SELF\analysis_model.model_type = PLANE_FRAME) OR
(SELF\analysis_model.model_type = PLANE_TRUSS) OR
(SELF\analysis_model.model_type = GRILLAGE);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own (i.e. attributes that are declared within the entity declaration). However, it does inherit several attributes from its SUPERTYPE analysis_model. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRA2**

The attribute coordinate_space_dimension (inherited from the SUPERTYPE analysis_model) shall be assigned a value of 2. In other words, the analysis model is defined in two-dimensional space.

WRA3

The attribute model_type (inherited from the SUPERTYPE analysis_model) shall be assigned a value of either PLANE_FRAME, PLANE_TRUSS or GRILLAGE. In other words, the analysis model represents either a plane frame, a plane truss or a grillage.

Notes

New for CIS/2.

See Diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.8 analysis_model_3D

Entity definition:

A type of analysis_model that represents part of the structure or assembly in three-dimensional space. This entity is used to represent a space frame, or a space truss for analysis purposes.

EXPRESS specification:

*)

ENTITY analysis_model_3D

SUBTYPE OF (analysis_model);

WHERE

WRA4 : SELF\analysis_model.coordinate_space_dimension = 3;

WRA5 : (SELF\analysis_model.model_type = SPACE_FRAME) OR
(SELF\analysis_model.model_type = SPACE_TRUSS);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own (i.e. attributes that are declared within the entity declaration). However, it does inherit several attributes from its SUPERTYPE analysis_model. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:

WRA4

The attribute coordinate_space_dimension (inherited from the SUPERTYPE analysis_model) shall be assigned a value of 3. In other words, the analysis model is defined in three-dimensional space. This means that the boundary conditions for the nodes and the releases for the element ends must be given values for six degrees of freedom.

WRA5

The attribute model_type (inherited from the SUPERTYPE analysis_model) shall be assigned a value of either SPACE_FRAME or SPACE_TRUSS. In other words, the analysis model represents either a space frame or a space truss.

Notes

New for CIS/2.

See Diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.9 analysis_model_child

Entity definition:

A type of analysis_model that allows the analysis_model to be associated with another parent analysis_model. A child analysis model can also be a parent of another child analysis_model. This allows the analysis_model to be decomposed into other

analysis_models. The decomposition is hierarchical; i.e. an analysis_model_child can only have one parent.

The LPM does not define how analysis models are decomposed; it merely requires that each instance of analysis_model is valid in its own right.

EXPRESS specification:

```
*)
ENTITY analysis_model_child
SUBTYPE OF (analysis_model);
    parent_model : analysis_model;
WHERE
    WRA6 : parent_model :<>: (SELF);
    WRA7 : SELF.analysis_model.coordinate_space_dimension <=
        parent_model.coordinate_space_dimension;
END_ENTITY;
(*
```

Attribute definitions:

parent_model

Declares the instance of analysis_model associated with the analysis_model_child. It is implied that the instance of analysis_model referenced to is the parent of the analysis_model_child; i.e. the child forms part of the parent analysis_model.

Formal propositions:

WRA6

The analysis_model referred to by the parent_model will be a separate instance. That is, an analysis_model cannot be its own parent. However, its parent can be another instance of analysis_model_child.

WRA7

The value of the coordinate_space_dimension for this instance of analysis_model_child (inherited from the SUPERTYPE analysis_model, must be less than or equal to the value of the coordinate_space_dimension of the parent model. In other words, a three-dimensional analysis model can only form part of another three-dimensional analysis model. Conversely, a plane frame will fit inside space frame.

Informal propositions:

If the child analysis model is located within a parent analysis model by coordinate systems, then the analysis_model_child is instanced with the sibling SUBTYPE entity analysis_model_located. Obviously, the child's coordinate system will be located within the parent's coordinate system. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2, but was addressed in ANALYSIS_MODEL in CIS/1.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.10 analysis_model_located

Entity definition:

A type of analysis_model that allows an analysis_model to be associated with (and thus located by) a coordinate system.

EXPRESS specification:

```
*)
ENTITY analysis_model_located
SUBTYPE OF (analysis_model);
    model_coord_sys : coord_system;
WHERE
    WRA8 : SELF\analysis_model.coordinate_space_dimension <=
        model_coord_sys.coord_system_dimensionality;
END_ENTITY;
(*
```

Attribute definitions:

model_coord_sys

Declares the instance of coord_system associated with the analysis_model_located. It is implied that the coord_system referenced provides the location of the analysis model.

Formal propositions:

WRA8

The value of the attribute coordinate_space_dimension (inherited from the SUPERTYPE analysis_model) must be less than or equal to the value assigned to the attribute coord_system_dimensionality of the coord_system referenced by the attribute model_coord_sys. In other words, the coordinate system used to locate the analysis model must be appropriate to the dimensionality of the analysis_model. For example, a space frame must be located by a three-dimensional coordinate system, but a plane frame may be located by either a two or three-dimensional coordinate system.

Informal propositions:

If the located analysis model is also a child analysis model, then the analysis_model_located is instanced with the sibling SUBTYPE entity analysis_model_child. Obviously, the child's coordinate system will be located within the parent's coordinate system. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2, but was addressed in ANALYSIS_MODEL in CIS/1.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.11 analysis_model_mapping

Entity definition:

An association between an analysis_model and a set of at least one assembly. The assembly may be any of its SUBTYPES (i.e. design, manufacturing, etc.). This provides the mapping between the analysis to the design and manufacturing aspects of the CIS. This entity represents the relationship between assemblies and analysis models, and is

intended to capture the stage in the design process when the frames are simplified for analysis purposes.

EXPRESS specification:

```
*)
ENTITY analysis_model_mapping;
    mapped_analysis_model : analysis_model;
    represented_assemblies : SET [1:?] OF assembly;
END_ENTITY;
(*
```

Attribute definitions:

mapped_analysis_model

Declares the instance of analysis_model associated with the analysis_model_mapping, and thereby associated with (and mapped onto) the assemblies.

represented_assemblies

Declares the set of instances of assembly associated with the analysis_model_mapping, and thereby associated with (and mapped onto) the analysis_model. There must be at least one instance of assembly associated with each instance of analysis_model_mapping. Each instance in the set must be distinct; i.e. repetition is not allowed.

Informal propositions:

An analysis_model may be mapped onto any number of assemblies.

An assembly may be mapped onto any number of analysis_models.

Notes:

New for CIS/2, but was partially addressed in ANALYSIS_MODEL in CIS/1.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.12 analysis_model_relationship

Entity definition:

An association between two distinct and separate analysis models. The relationship must be given a name and may be given a description. An analysis_model may be involved with any number of relationships, and therefore may be related to any number of other analysis_models.

EXPRESS specification:

```
*)
ENTITY analysis_model_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    relating_model : analysis_model;
    related_model : analysis_model;
WHERE
    WRA9 : relating_model :<>: related_model;
END_ENTITY;
(*
```

Attribute definitions:*relationship_name*

A short alphanumeric reference for the analysis_model_relationship; e.g. “is connected to”, “supports”, “is loaded by”, etc.

relationship_description

A free text description of the analysis_model_relationship.

relating_model

Declares the first instance of analysis_model associated with the analysis_model_relationship. It is implied that the analysis_model referenced here is related to the second analysis_model referenced.

related_model

Declares the second instance of analysis_model associated with the analysis_model_relationship. It is implied that the analysis_model referenced here is related to the first analysis_model referenced.

Formal propositions:*WRA9*

The instance of analysis_model referred to by the relating_model will be separate from the instance of analysis_model referred to by the related_model. That is, an analysis model cannot be related to itself.

Notes:

New for CIS/2, but was partially addressed in ANALYSIS_MODEL in CIS/1.

See diagram 22 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.13 boundary_condition**Entity definition:**

Defines the support conditions for a node or a set of nodes. The SUBTYPEs of this abstract SUPERTYPE may be used to define up to 7 degrees of freedom: i.e. 3 translations and 3 rotations and warping. For each direction the given value determines if the displacement is free, fixed, or spring. It applies to those nodes that act as supports and describes how a node is related to the external world. The boundary_condition must be given a name and may be given a description. As it is an abstract SUPERTYPE one of the SUBTYPEs must be instantiated. The detailed information is held in the SUBTYPEs. Examples of simple boundary conditions are shown in Figure 5.4.

Boundary conditions are defined in the global coordinate system of the analysis model. Where the degrees of freedom under consideration do not align with the nodal axes as defined by the global coordinate system, the boundary_condition is said to be skewed. The angle of the skew is captured by the entity boundary_condition_skewed.

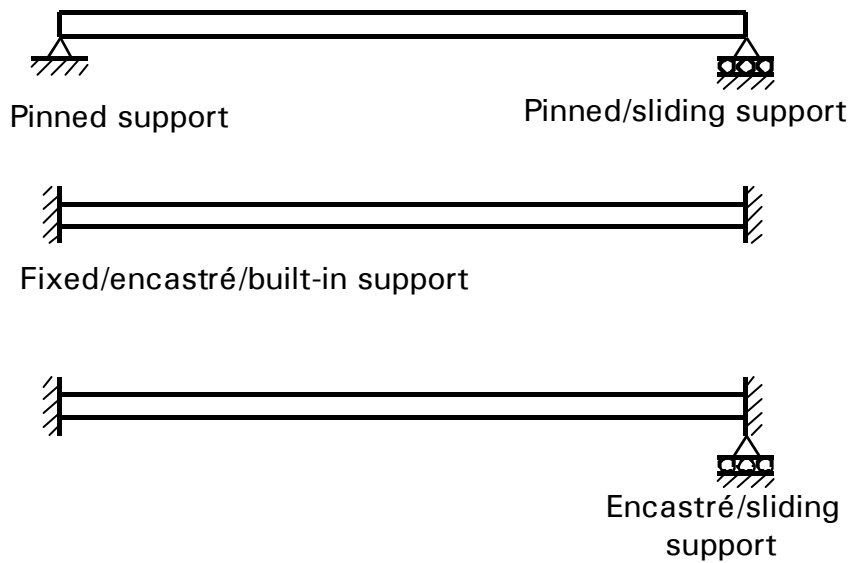


Figure 5.4 *Examples of boundary conditions*

EXPRESS specification:

```

*)
ENTITY boundary_condition
ABSTRACT SUPERTYPE OF (ONEOF
    (boundary_condition_logical,
    boundary_condition_spring_linear,
    boundary_condition_spring_non_linear) ANDOR
    boundary_condition_skewed);
    boundary_condition_name : label;
    boundary_condition_description : OPTIONAL text;
INVERSE
    restrained_nodes : SET [1:?] OF node FOR restraints;
END_ENTITY;
(

```

Attribute definitions:

boundary_condition_name

A short human-readable alphanumeric reference for the boundary_condition; e.g. “fully fixed”, or “pinned about z-axis”.

boundary_condition_description

A free text description of the boundary_condition.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

restrained_nodes

The boundary condition shall apply to a set of one or more nodes; i.e. the boundary_condition cannot exist without it being referenced by at least one node.

Notes:

Known as BOUNDARY_CONDITION in CIS/1, SUBTYPEs added.

The conventions used for nodal boundary conditions differ from those for member restraints. Thus, the definitions for the entity boundary_condition and its SUBTYPEs differ from those of restraint and its SUBTYPEs.

See diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.14 boundary_condition_logical***Entity definition:***

A type of boundary_condition where six degrees of freedom are given logical values (true, false or unknown). This is a simple representation of a nodal support in which warping is ignored.

Analysis models defined in three dimensions should have nodes that reference instances of boundary_condition_logical that have all of their attributes assigned values as either TRUE or FALSE. Those defined in one plane only (i.e. out of plane conditions ignored) should have nodes that reference instances of boundary_condition_logical with some of their attributes assigned the value UNKNOWN.

Each of the six attributes may be assigned a value of either TRUE, FALSE, or UNKNOWN. This corresponds to a nodal boundary condition defined in the coordinate system of the parent analysis model that is either 'free', 'fixed', or UNKNOWN. The value will be UNKNOWN if that particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model.

EXPRESS specification:

*)

ENTITY boundary_condition_logical

SUBTYPE OF (boundary_condition);

bc_x_displacement_free : LOGICAL;

bc_y_displacement_free : LOGICAL;

bc_z_displacement_free : LOGICAL;

bc_x_rotation_free : LOGICAL;

bc_y_rotation_free : LOGICAL;

bc_z_rotation_free : LOGICAL;

END_ENTITY;

(*

Attribute definitions:***bc_x_displacement_free***

Declares the value of the degree of freedom for the boundary condition for linear displacement in the x direction, in the global coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the x direction is unrestrained (i.e. 'free') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

bc_y_displacement_free

Declares the value of the degree of freedom for the boundary condition for linear displacement in the y direction, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the y direction is unrestrained (i.e. 'free') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

bc_z_displacement_free

Declares the value of the degree of freedom for the boundary condition for linear displacement in the z direction, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the z direction is unrestrained (i.e. 'free') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

bc_x_rotation_free

Declares the value of the degree of freedom for the boundary condition for rotation about the x axis, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the x axis is unrestrained (i.e. 'free') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

bc_y_rotation_free

Declares the value of the degree of freedom for the boundary condition for rotation about the y axis, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the y axis is unrestrained (i.e. 'free') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

bc_z_rotation_free

Declares the value of the degree of freedom for the boundary condition for rotation about the z axis, in the local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the z axis is unrestrained (i.e. 'free') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

Informal propositions:

If the boundary condition is skewed (such that the degrees of freedom under consideration do not align with the nodal axes as defined by the global coordinate system

of the analysis model), then the `boundary_condition_logical` is instanced with the sibling SUBTYPE entity `boundary_condition_skewed`. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Although it is possible to assign a value of UNKNOWN to all of the attributes, this is not recommended as the populated instance would have little semantic value.

Notes

New for CIS/2.

See Diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.15 boundary_condition_skewed

Entity definition:

A type of `boundary_condition` where the values for the degrees of freedom are defined at an angle to the nodal axes. This attribute is used when the degrees of freedom under consideration do not align with the nodal axes as defined by the global coordinate system of the analysis model.

EXPRESS specification:

*)

ENTITY `boundary_condition_skewed`

SUBTYPE OF (`boundary_condition`);

`x_skew_angle` : OPTIONAL `plane_angle_measure_with_unit`;

`y_skew_angle` : OPTIONAL `plane_angle_measure_with_unit`;

`z_skew_angle` : OPTIONAL `plane_angle_measure_with_unit`;

WHERE

WRB9 : EXISTS (`x_skew_angle`) OR

EXISTS (`y_skew_angle`) OR

EXISTS (`z_skew_angle`);

END_ENTITY;

(*

Attribute definitions:

`x_skew_angle`

Declares the first instance of `plane_angle_measure_with_unit` that may be associated with the instance of `boundary_condition_skewed`. This attribute provides the numerical value with an appropriate unit of the angle of rotation from the global x-axis of the analysis model to the defining axis for the boundary condition, when the boundary condition applies to a skewed support. The value of the angle should lie between the equivalent of ± 90 degrees. The value should not be zero.

`y_skew_angle`

Declares the second instance of `plane_angle_measure_with_unit` that may be associated with the instance of `boundary_condition_skewed`. This attribute provides the numerical value with an appropriate unit of the angle of rotation from the global y-axis of the analysis model to the defining axis for the boundary condition, when the boundary condition applies to a skewed support. The value of the angle should lie between the equivalent of ± 90 degrees. The value should not be zero.

z_skew_angle

Declares the third instance of `plane_angle_measure_with_unit` that may be associated with the instance of `boundary_condition_skewed`. This attribute provides the numerical value with an appropriate unit of the angle of rotation from the global z-axis of the analysis model to the defining axis for the boundary condition, when the boundary condition applies to a skewed support. The value of the angle should lie between the equivalent of ± 90 degrees. The value should not be zero.

Formal propositions:*WRB9*

One of the attributes `x_skew_angle`, `y_skew_angle`, or `z_skew_angle` must be assigned a value. In other words, a skewed boundary condition must have at least one skew angle.

Informal propositions:

Although it is possible to instance this entity on its own, it is more likely to be instanced with one of its sibling SUBTYPEs `boundary_condition_logical`, `boundary_condition_spring_linear`, or `boundary_condition_spring_non_linear`. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Although it is possible to assign a value of 'zero' to all of the skew angles, this is not recommended as the populated instance would have little semantic value.

Notes

New for CIS/2.

See Diagram 23 of the EXPRESS-G diagrams in Appendix B.

5.3.16 boundary_condition_spring_linear**Entity definition:**

A type of `boundary_condition`, providing the values of the stiffnesses of a linear spring boundary condition. If this entity is instanced on its own, the boundary condition is defined in the global axes of the analysis model. Where the degrees of freedom under consideration do not align with the nodal axes as defined by the global coordinate system, the boundary condition is said to be skewed, and the angle of the skew is captured by the entity `boundary_condition_skewed`.

Analysis models defined in three dimensions should have nodes that reference instances of `boundary_condition_spring_linear` that have all of their attributes populated. Those defined in one plane only (i.e. out of plane conditions ignored) should have nodes that reference instances of `boundary_condition_spring_linear` with some of their attributes assigned a NULL (\$) value.

EXPRESS specification:

*)

```
ENTITY boundary_condition_spring_linear
SUPERTYPE OF (boundary_condition_warping)
SUBTYPE OF (boundary_condition);
```

```
    bc_x_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    bc_y_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    bc_z_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    bc_x_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
```

```

    bc_y_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
    bc_z_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
WHERE
    WRB10 : EXISTS (bc_x_displacement) OR
            EXISTS (bc_y_displacement) OR
            EXISTS (bc_z_displacement) OR
            EXISTS (bc_x_rotation) OR
            EXISTS (bc_y_rotation) OR
            EXISTS (bc_z_rotation);
END_ENTITY;
(*)

```

Attribute definitions:

bc_x_displacement

Declares the first instance of `linear_stiffness_measure_with_unit` that may be used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for linear displacement in the x-direction. If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ boundary condition, this attribute should be assigned a value equivalent to a zero spring stiffness.

bc_y_displacement

Declares the second instance of `linear_stiffness_measure_with_unit` that may be used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for linear displacement in the y-direction. If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ boundary condition, this attribute should be assigned a value equivalent to a zero spring stiffness.

bc_z_displacement

Declares the third instance of `linear_stiffness_measure_with_unit` that may be used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for linear displacement in the z-direction. If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ boundary condition, this attribute should be assigned a value equivalent to a zero spring stiffness.

bc_x_rotation

Declares the first instance of `rotational_stiffness_measure_with_unit` that may be used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for rotation about the x-axis. If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated

(and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ boundary condition, this attribute should be assigned a value equivalent to a zero spring stiffness.

bc_y_rotation

Declares the second instance of `rotational_stiffness_measure_with_unit` that may be used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for rotation about the y-axis. If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ boundary condition, this attribute should be assigned a value equivalent to a zero spring stiffness.

bc_z_rotation

Declares the third instance of `rotational_stiffness_measure_with_unit` that may be used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for rotation about the z-axis. If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ boundary condition, this attribute should be assigned a value equivalent to a zero spring stiffness.

Formal propositions

WRB10

At least one of the attributes `bc_x_displacement`, `bc_y_displacement`, `bc_z_displacement`, `bc_x_rotation`, `bc_y_rotation`, or `bc_z_rotation` must be assigned a value. In other words, the boundary condition must have a spring stiffness in at least one linear direction or about at least one axis.

Informal propositions:

If the boundary condition is skewed (such that the degrees of freedom under consideration do not align with the nodal axes as defined by the global coordinate system of the analysis model), then the `boundary_condition_spring_linear` is instantiated with the sibling SUBTYPE entity `boundary_condition_skewed`. The use of the ANDOR SUPERTYPE construct will require the complex instance to be externally mapped in the STEP Part 21 file.

Notes:

New for CIS/2, was partially covered by `BOUNDARY_CONDITION` in CIS/1.

See diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.17 boundary_condition_spring_non_linear

Entity definition:

Type of `boundary_condition`, providing the values of the spring stiffnesses of a non-linear spring boundary condition. The non-linear spring boundary condition is defined as a

series of linear spring boundary conditions with change values between consecutive conditions. In this way, a lower and upper bound are specified for the range in which the linear spring boundary condition is valid.

The effect of a non-linear spring boundary condition on the moment-rotation characteristics of the nodal support are illustrated in Figure 5.5.

EXPRESS specification:

```

*)
ENTITY boundary_condition_spring_non_linear
SUBTYPE OF (boundary_condition);
    change_values : LIST [2:?] OF measure_with_unit;
    values : LIST [2:?] OF boundary_condition_spring_linear;
DERIVE
    number_of_values : INTEGER := SIZEOF(change_values);
WHERE
    WRB12 : SIZEOF(values) = SIZEOF(change_values);
END_ENTITY;
(*

```

Attribute definitions:

change_values

Declares the list of instances of `measure_with_unit` associated with the instance of `boundary_condition_spring_non_linear`. There must be at least two instances of `measure_with_unit` associated with each instance of `boundary_condition_spring_non_linear`. The order of the list is important as consecutive instances in the list represent the change values for consecutive conditions. Each instance in the list provides the numerical value with an appropriate unit of the upper bound for the range of validity of the previous instance of `boundary_condition_spring_linear` and a lower bound for the range of validity of the next instance of `boundary_condition_spring_linear`.

The lower bound for the first instance of `boundary_condition_spring_linear` (referenced by the attribute `values`) is assumed to be zero. Thus, the first instance of `measure_with_unit` (referenced by the attribute `change_values`) provides the upper bound for the range of validity of the first instance of `boundary_condition_spring_linear`. It also provides the lower bound for the range of validity of the second instance of `boundary_condition_spring_linear`.

values

Declares the list of instances of `boundary_condition_spring_linear` that are associated with (and are used to define) the `boundary_condition_spring_non_linear`. There must be at least two instances of `boundary_condition_spring_linear` associated with each instance of `boundary_condition_spring_non_linear`. The order of the list is important as consecutive instances in the list represent consecutive conditions.

number_of_values

This attribute derives the integer value of the number of instances of `boundary_condition_spring_linear` that are used to define the `boundary_condition_spring_non_linear` from the size of the aggregation associated with the attribute `change_values`.

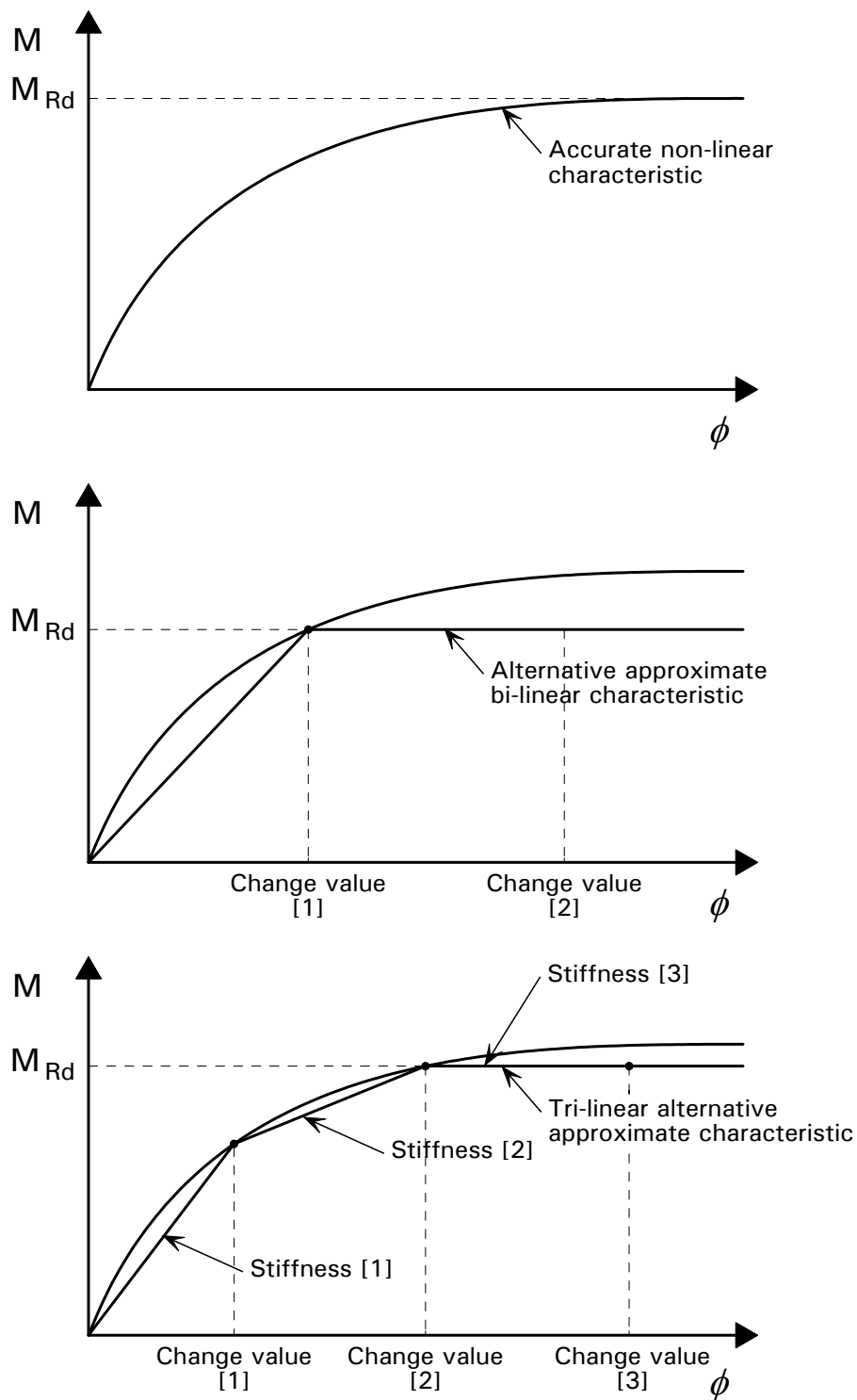


Figure 5.5 *The moment-rotation relationship associated with of non-linear spring boundary condition*

Formal propositions:

DERIVE

The derived attribute number_of_values does not appear in the STEP Part 21 file.

WRB12

The size of the aggregation associated with the attribute `change_values` shall be equal to the size of the aggregation associated with the attribute `values`. In other words, a non-linear spring must be defined as a sequence of at least two linear springs each provided with a change value.

Notes:

New for CIS/2.

See diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.18 boundary_condition_warping**Entity definition:**

A type of `boundary_condition` that is defined as a linear spring boundary condition with the seventh degree of freedom populated. The value of this warping boundary condition (for rotation about a 'warping axis') is defined by reference to an instance of `rotational_stiffness_measure_with_unit`.

EXPRESS specification:

```
*)
ENTITY boundary_condition_warping
  SUBTYPE OF (boundary_condition_spring_linear);
    bc_warping : rotational_stiffness_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:**bc_warping**

Declares the instance of `rotational_stiffness_measure_with_unit` used to define the `boundary_condition_linear`. This provides the numerical value with an appropriate unit of the degree of freedom for the boundary condition for rotation about the warping axis. The spring stiffness value provided must be greater than zero.

Notes

New for CIS/2.

See Diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.19 element**Entity definition:**

An element is a component of (and belongs to) an analysis model. Elements are defined in space by their relationship to a number of nodes. The element must be given a name, and may also be given a description. It must also be associated with an analysis model. The element will usually be instanced as one of its SUBTYPES. An element can have 0, 1, 2 or 3 dimensions.

Elements may have their geometry defined through the SUBTYPES which make references to various geometric entities. If a SUBTYPE is not used then the element has no shape. Elements may have their physical properties defined through the SUBTYPE

element_with_material by the associated material. Because of the ANDOR nature of the SUPERTYPE construct, an element may be defined with or without a material.

The connectivity between nodes and elements is established by their relationships with the entity element_node_connectivity.

EXPRESS specification:

```

*)
ENTITY element
SUPERTYPE OF (ONEOF
    (element_volume,
    element_surface,
    element_curve,
    element_point) ANDOR
    element_with_material);
element_name : label;
element_description : OPTIONAL text;
parent_model : analysis_model;
element_dimensionality : INTEGER;
INVERSE
    connectivity : SET [1:?] OF element_node_connectivity FOR connecting_element;
UNIQUE
    URE1 : element_name, parent_model;
WHERE
    WRE2 : element_dimensionality <= parent_model.coordinate_space_dimension;
    WRE21 : (element_dimensionality >= 0) AND (element_dimensionality <= 3);
END_ENTITY;
(*

```

Attribute definitions:

element_name

A short human-readable alphanumeric reference for the element, used to uniquely identify the element within its parent analysis model.

element_description

A free text description of the element.

parent_model

Declares the instance of analysis_model associated with the element. It is implied that the element belongs to the analysis_model referenced here.

element_dimensionality

Specifies the dimensionality of the element as an integer; i.e.

- 0 - point element,
- 1 - line element,
- 2 - surface / plate element,
- 3 - volume/space element.

Formal propositions:**ANDOR SUPERTYPE**

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instantiated with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

connectivity

The element must be associated with at least one instance of element_node_connectivity; i.e. the element cannot exist without the connectivity being specified.

URE1

The combination of the element_name and the parent_model shall be unique to the element; i.e. no other instance of element belonging to an analysis_model may have the same name.

WRE2

The value assigned to the attribute element_dimensionality must be less than or equal to the value assigned to the attribute coordinate_space_dimension of the analysis_model to which the element belongs. In other words, the dimensionality of the element must be appropriate to the dimensionality of the analysis model. For example, a one-dimensional element may be defined in a 2D or a 3D analysis model. Conversely, a three-dimensional element cannot be defined within a 2D analysis model.

WRE21

The value of the attribute element_dimensionality shall be greater than or equal to 0 and less than or equal to 3. (See SUBTYPES for further constraints).

Notes:

Known as ELEMENT in CIS/1, SUBTYPES added.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.20 element_curve**Entity definition:**

A type of element that is connected to two nodes, and whose dimensionality is set equal to one. The detailed information about the shape of the element is given in the SUBTYPES. The element_curve can be either a simple or complex curved element depending on how the shape is defined. The shape of the element is defined by one or more cross sections and an axis.

An element_curve has an implied coordinate system, which is initially defined by the two connecting nodes. Where non-zero values are assigned to an associated element_eccentricity, the end of the element does not coincide with the connecting node and the element's coordinate system is relocated to suit. The origin of the element's coordinate system is taken to be located at the element's 'start end' (i.e. the first connectivity).

EXPRESS specification:

*)

ENTITY element_curve

ABSTRACT SUPERTYPE OF (ONEOF

```

    (element_curve_simple,
    element_curve_complex))
SUBTYPE OF (element);
    element_subdivision : OPTIONAL INTEGER;
DERIVE
    connectivities : SET [2:2] OF element_node_connectivity := bag_to_set (USEDIN (SELF,
    'STRUCTURAL_FRAME_SCHEMA.
    ELEMENT_NODE_CONNECTIVITY.CONNECTING_ELEMENT'));
WHERE
    WRE3 : SELF\element.element_dimensionality = 1;
    WRE4 : connectivities[1] :<>: connectivities[2];
    WRE5 : connectivities[1].connecting_node :<>: connectivities[2].connecting_node;
END_ENTITY;
(*)

```

Attribute definitions:

element_subdivision

Declares the number of divisions made in the element. If provided, this value can be used to determine, for example, the positions at which the analysis results are calculated or output.

connectivities

This attribute derives the set of two instances of *element_node_connectivity* that reference this instance of *element_curve* (via the attribute *connecting_element*). Both the upper and lower bounds of the set are fixed as 2; thus, there must be two (and only two) distinct and separate instances of *element_node_connectivity* that reference this instance of *element_curve*. In other words, a curved element must have two (and only two) distinct and separate ends.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

DERIVE

The derived attribute *connectivities* does not appear in the STEP Part 21 file.

WRE3

The dimensionality of the element (inherited from the SUPERTYPE element) shall be set = 1.

WRE4

The instance of *element_node_connectivity* referenced by the first member of set derived by the attribute *connectivities*, must not be the same instance referenced by the second member of the set. In other words, a curved element must have two distinct and separate ends.

WRE5

The instance of node referenced by the first member of the set instances of *element_node_connectivity* derived by the attribute *connectivities*, must not be the same

instance referenced by the second member of the set. In other words, a curved element must have two distinct and separate connecting nodes.

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.21 element_curve_complex

Entity definition:

A type of `element_curve` that allows the shape of the curved element to be defined using explicit geometry. The `element_curve_complex` has an axis, which is defined by a series of (at least 2) points on a curve. At each of these points along the axis, a `section_profile` is referenced which defines the shape of the cross section of the element at that point. It is assumed that between each defining point there is a linear transition between each section profile.

EXPRESS specification:

```
*)
ENTITY element_curve_complex
SUBTYPE OF (element_curve);
    cross_sections : LIST [2:?] OF section_profile;
    points_defining_element_axis : LIST [2:?] OF point_on_curve;
    element_orientations : LIST [2:?] OF orientation_select;
DERIVE
    number_of_sections : INTEGER := SIZEOF (cross_sections);
    curve_defining_element : curve :=
        points_defining_element_axis[1]\point_on_curve.basis_curve;
WHERE
    WRE6 : ( (SIZEOF (points_defining_element_axis) = number_of_sections)
        AND (SIZEOF (element_orientations) = number_of_sections) );
    WRE7 : SIZEOF(QUERY(temp <* points_defining_element_axis |
        (temp\point_on_curve.basis_curve) :<>: curve_defining_element)) = 0;
END_ENTITY
(*
```

cross_sections

Declares the list of instances of `section_profile` associated with the `element_curve_complex`. It is implied that each of the section profiles referenced defines the shape of the complex curved element. It is assumed that between each defining point there is a linear transition between each section profile. The LIST data type allows a reference to a `section_profile` to be repeated. If the cross section of the element is constant throughout, then this attribute will reference the same instance of `section_profile` (at least) twice. The section profile is located such that its cardinal point sits on the defining curve of the element. The cardinal point may (but need not) be the geometric centroid of the section profile. The cardinal point may (but need not) be constant for each instance of `section_profile` referenced. In the simplest case, the same cardinal point will be used throughout the element.

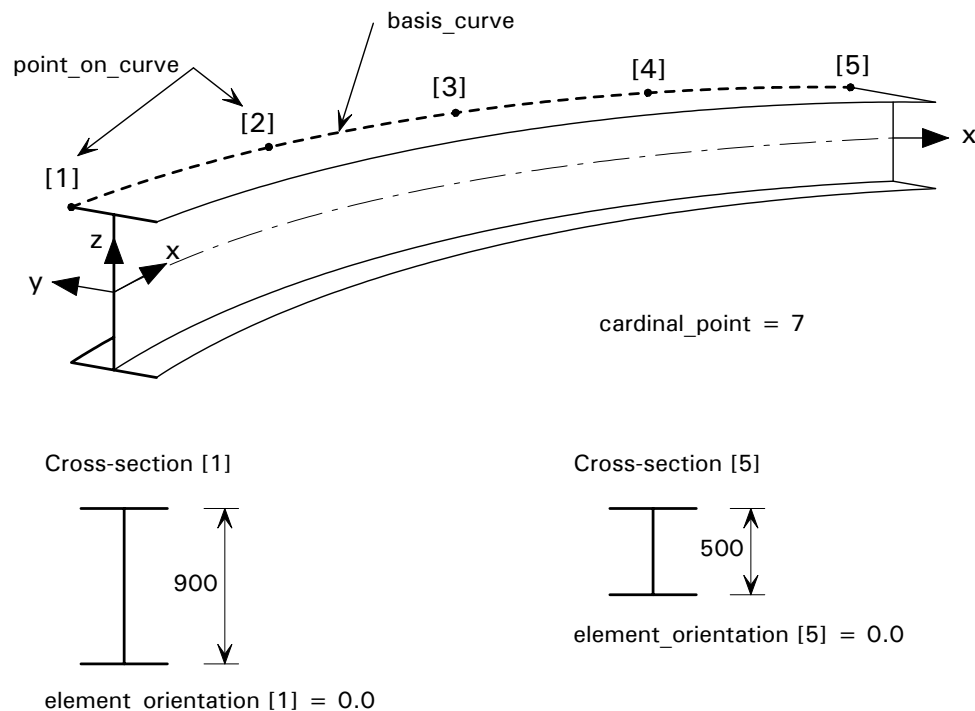


Figure 5.6 Example of an *element_curve_complex*

Attribute definitions:

points_defining_element_axis

Declares the list of unique instances of *point_on_curve* associated with the *element_curve_complex*. It is implied that the longitudinal axis of the element is defined by the curve that is referenced by the *point_on_curve*.

It is assumed that all the instances of *point_on_curve* are all defined on the same curve, as referenced by the attribute *point_on_curve* of the entity *basis_curve*. That is, all the points for which the cross-section is defined all lie on the same curve. It is also assumed that this curve defines the locating longitudinal (x) axis of the element, upon which the cardinal points of the section profiles are placed. This may (but need not) correspond to the analytical longitudinal axis of the element.

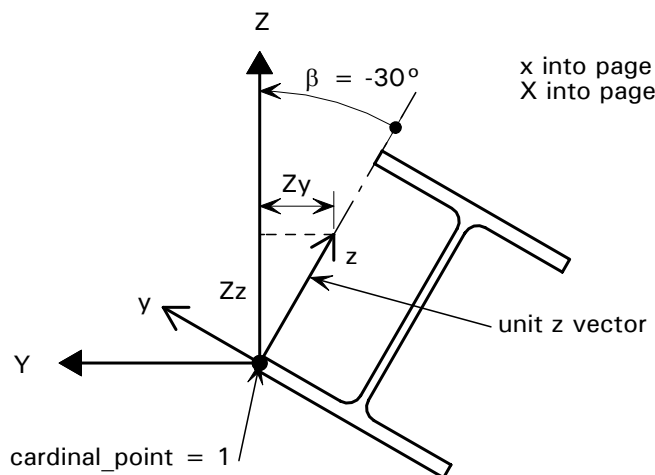
element_orientations

Declares the list of two or more instances of *plane_angle_measure_with_unit* or *direction* associated with this instance of *element_curve_complex*. The choice of which entity is used is made through the SELECT type *orientation_select*. The order of the instances in the list is significant. Each instance gives the numerical value of the orientation of the element at the corresponding point on the element's locating longitudinal axis. As the cardinal point of the section profile defines the locating longitudinal axis of the element, the rotation of the section profile is defined as being about its cardinal point.

If this attribute references an instance of *plane_angle_measure_with_unit* (via the SELECT type) then the orientation of the element is the beta angle measured (with an appropriate unit) from the element's z-axis to the plane generated by the projection of the element's local x-axis in the global Z-axis direction. In the case where the element's x-axis lies parallel to the global Z-axis, the orientation of the element will be the angle measured from the element's z-axis to the global X-axis.

If this attribute references an instance of *direction* (via the SELECT type) then the orientation of the element is defined by the three real number values of the

direction_ratios attribute (of the entity direction). These numbers define the ‘orientation vector’ for the element. That is, the direction of the re-oriented local z-axis relative to the global coordinate system. For example, the normal (unrotated) orientation for a ‘beam’ is given as (0.0, 0.0, 1.0), while the orientation vector for unrotated ‘columns’ is given as (1.0, 0.0, 0.0).



$$\begin{aligned}
 \text{orientation vector} &= [Zx, Zy, Zz] \\
 &= [0, \sin \beta, \cos \beta] \\
 &= [0, -0.5, 0.866]
 \end{aligned}$$

Figure 5.7 *The relationship between a beta angle and an orientation vector for a beam element (Example 1)*

The two different forms of orientation are related such that one may be derived from the other (as shown in Figure 5.7 and Figure 5.8). Thus:

- for beams, the orientation vector = $[0.0, \sin \beta, \cos \beta]$
- for columns, the orientation vector = $[\cos \beta, -\sin \beta, 0.0]$

Thus, the values provided for the orientation vector will also indicate whether the element represents a column or not.

It should be noted that if a direction with three values is used to define the orientation, one of the values assigned to the direction_ratios attribute should be zero, while the other two values should be non-zero. As the orientation is defined in one plane (perpendicular to the element’s x-axis), a direction assigned three non-zero values would be incorrect. The number of values assigned to the direction_ratios attribute is governed by the global rule compatible_dimension. This ensures that the count of direction_ratios matches the coordinate_space_dimension of the geometric_context in which the direction is geometrically_founded. This should be the same geometric_context used for the other geometric_representation_items in the model; e.g the points defining the nodes of the analysis_model.

It should also be noted that this attribute does NOT represent the global orientation of the element.

number_of_sections

This attribute derives the total number of cross sections that are used to define the instance of element_curve_complex. The value is derived by calculating the size of the list of instances of section_profile in the populated attribute cross_sections.

curve_defining_element

This attribute derives the curve upon which all the defining points of the element_curve_complex lie. The curve is derived by examining the first instance in the list of instances of point_on_curve referenced by the attribute points_defining_element_axis.

Formal propositions:**DERIVE**

The derived attributes number_of_sections and curve_defining_element do not appear in the STEP Part 21 file.

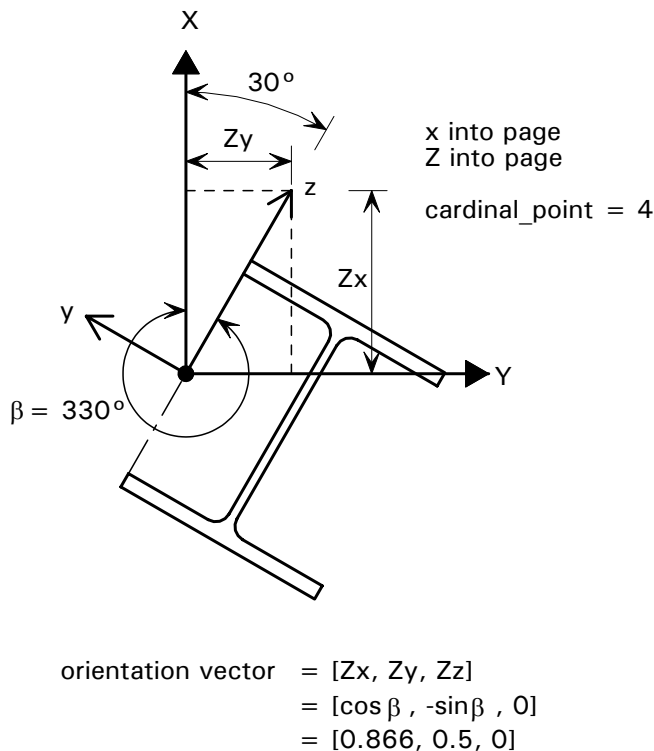


Figure 5.8 The relationship between a beta angle and an orientation vector for a column element (Example 1)

WRE6

The size of the list of instances of point_on_curve referenced by the attribute points_defining_element_axis shall equal the value of the derived attribute number_of_sections. Furthermore, the size of the list of instances of plane_angle_measure_with_unit referenced by the attribute element_orientations shall also equal the value of the derived attribute number_of_sections. In other words, for every point defining the axis of the element_curve_complex there must be a corresponding section profile and angle of orientation.

WRE7

The size of the list of instances of point_on_curve having a different basis_curve from the first instance of point_on_curve shall equal zero. In other words, all the points defining the longitudinal axis of the element must lie on the same curve.

Informal propositions:

The LIST data types used in the attributes `cross_sections` and `points_defining_element_axis` imply that the orders of the instances are important. It is assumed that the each instance of `section_profile` referred to by an instance of `element_curve` lies on the appropriate instance of `point_on_curve` given in sequence. That is, the third `section_profile` lies on the third `point_on_curve`, and so on.

Notes:

New for CIS/2.

The entity `point_on_curve` is defined in STEP Part 42.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.22 element_curve_simple**Entity definition:**

A type of `element_curve` that allows the shape of the curved element to be defined by the shape of its cross section; i.e. a simple straight line element. An example of how a simple curved element is defined is illustrated in Figure 5.9.

The `element_curve_simple` has its longitudinal axis defined initially by the straight line between the two nodes it is connected to (through the entity `element_node_connectivity`). The locating longitudinal axis of the element may be repositioned by the non-zero values given in `element_eccentricity`.

The nodes are defined within the global coordinate system of the analysis model, which may or may not be located within another coordinate system.

When using a Cartesian coordinate system, this longitudinal axis of a curved element is defined as the element's x-axis. An `element_curve_simple` has an implied local coordinate system with its origin (0.0, 0.0, 0.0) initially taken as the first connected node. The element's x-axis initially lies between the first 2 nodes; the y and z-axes are initially defined by the default values of the y and z-axes of the section profile associated with the element. The origin and the x-axis will shift if non-zero values are assigned to the `element_eccentricity`. The y and z-axes may be rotated by the `element_orientation`. If the `element_orientation` is given a zero or null value, the y and z-axes have not been rotated and lie in the original position defined by the `section_profile` and its cardinal point.

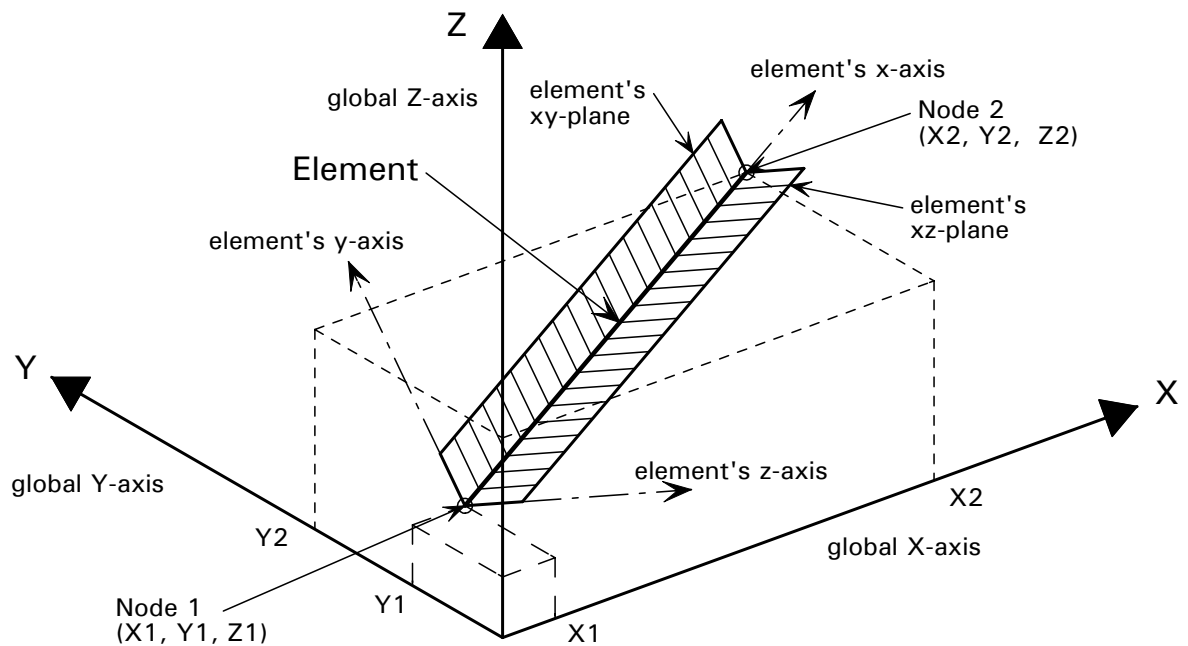


Figure 5.9 *Defining simple curve elements*

EXPRESS specification:

```

*)
ENTITY element_curve_simple
SUBTYPE OF (element_curve);
    cross_section : section_profile;
    element_orientation : orientation_select;
END_ENTITY;
(*

```

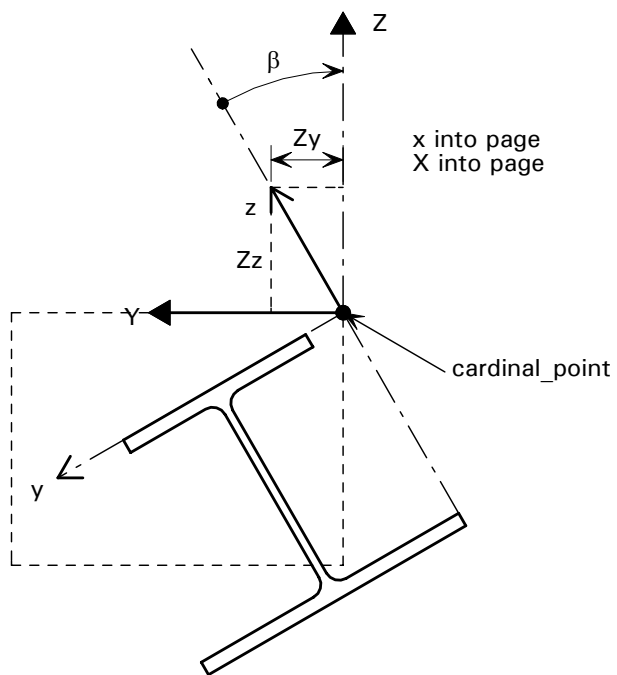
Attribute definitions:

cross_section

Declares the instance of `section_profile` associated with the `element_curve_simple`. It is implied that this instance defines the cross section of the whole element. The section profile is located such that its cardinal point sits on the locating longitudinal axis of the element. The cardinal point may (but need not) be the geometric centroid of the section profile.

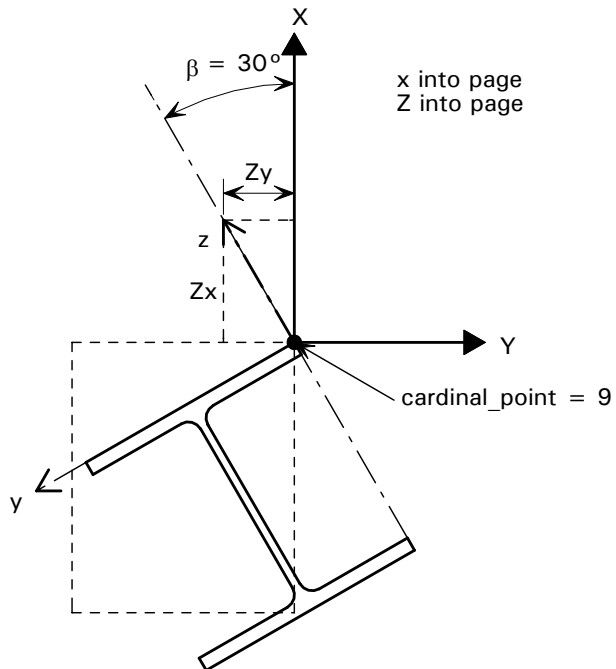
element_orientation

Declares the instance of `plane_angle_measure_with_unit` or `direction` associated with this instance of `element_curve_simple`, which provides the numerical value of the rotation of the section profile about the element's locating longitudinal axis.



orientation vector = $[Zx, Zy, Zz]$
 $[0, \sin \beta, \cos \beta]$

Figure 5.10 *The relationship between a beta angle and an orientation vector for a beam element (Example 2)*



orientation vector = $[Zx, Zy, Zz]$
 $= [\cos \beta, -\sin \beta, 0]$
 $= [0.866, -0.5, 0]$

Figure 5.11 *The relationship between a beta angle and an orientation vector for a column element (Example 2)*

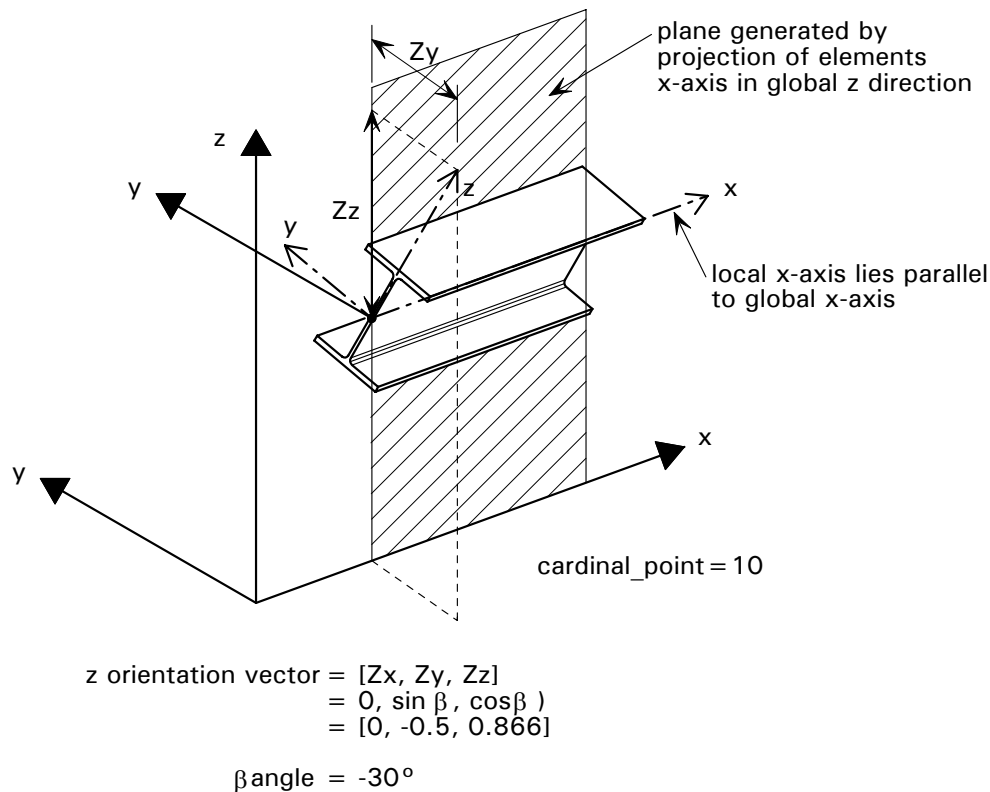


Figure 5.12 *Defining the orientation of a beam element*

This attribute specifies how the element's axis system is oriented with respect to the axis system of the analysis model. As the cardinal point of the section profile defines the locating longitudinal axis of the element, the rotation of the section profile is defined as being about its cardinal point. The choice of which entity is used is made through the SELECT type orientation_select.

If this attribute references an instance of plane_angle_measure_with_unit (via the SELECT type) then the orientation of the element is the beta angle measured (with an appropriate unit) from the element's z-axis to the plane generated by the projection of the element's local x-axis in the global Z-axis direction. In the case where the element's x-axis lies parallel to the global Z-axis (e.g. columns), the orientation of the element will be the angle measured from the element's z-axis to the global X-axis.

If this attribute references an instance of direction (via the SELECT type) then the orientation of the element is defined by the three real number values of the direction_ratios attribute (of the entity direction). These numbers define the 'orientation vector' for the element. That is, the direction of the re-oriented local z-axis relative to the global coordinate system. For example, the normal (unrotated) orientation for a 'beam' is given as (0.0, 0.0, 1.0), while the orientation vector for unrotated 'columns' is given as (1.0, 0.0, 0.0).

The two different forms of orientation are related such that one may be derived from the other (as shown in Figure 5.10 and Figure 5.11). Thus:

- for beams, the orientation vector = $[0.0, \sin \beta, \cos \beta]$
- for columns, the orientation vector = $[\cos \beta, -\sin \beta, 0.0]$

Thus, the values provided for the orientation vector will also indicate whether the element represents a column or not.

It should be noted that if a direction with three values is used to define the orientation, one of the values assigned to the `direction_ratios` attribute should be zero, while the other two values should be non-zero. As the orientation is defined in one plane (perpendicular to the element's x-axis), a direction assigned three non-zero values would be incorrect. The number of values assigned to the `direction_ratios` attribute is governed by the global rule `compatible_dimension`. This ensures that the count of `direction_ratios` matches the `coordinate_space_dimension` of the `geometric_context` in which the direction is geometrically founded. This should be the same `geometric_context` used for the other `geometric_representation_items` in the model.

Figure 5.12 shows a I section that is positioned in the analysis model such that its z-axis lies 30 degrees from the vertical. As it is measured from the element's z-axis and clockwise rotation is taken as positive, the element orientation of this element may be expressed as either 330 or -30 degrees (or the equivalent value in radians).

The definition of the orientation of column elements differs from that of beam elements. Examples are shown in Figure 5.13, Figure 5.14, and Figure 5.15.

It should be noted that this attribute does NOT represent the global orientation of the element.

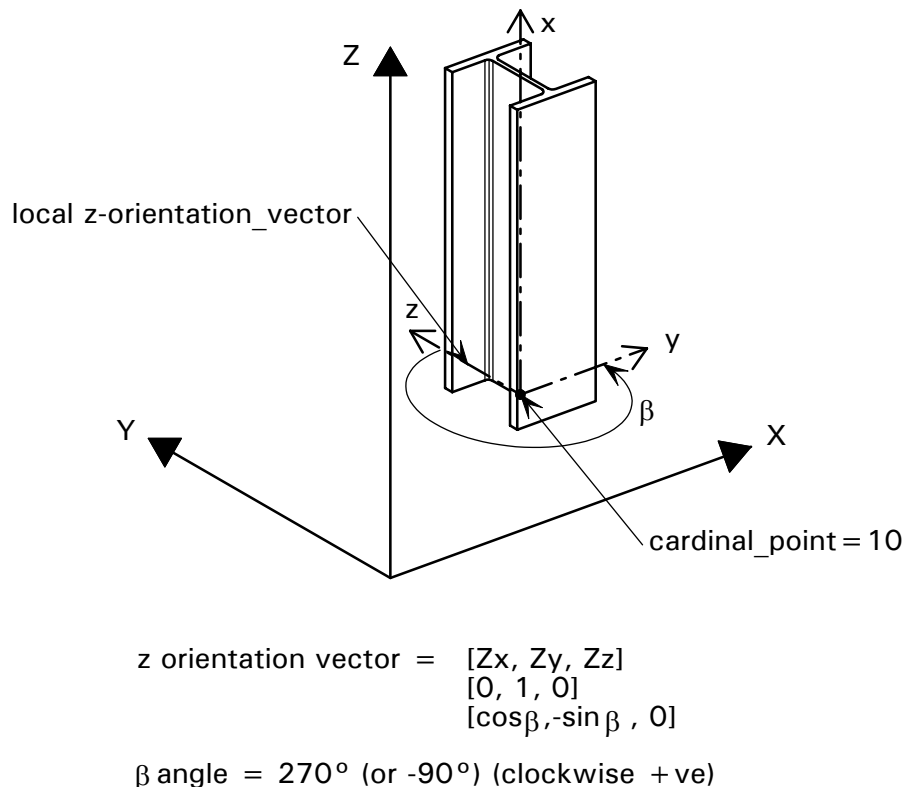
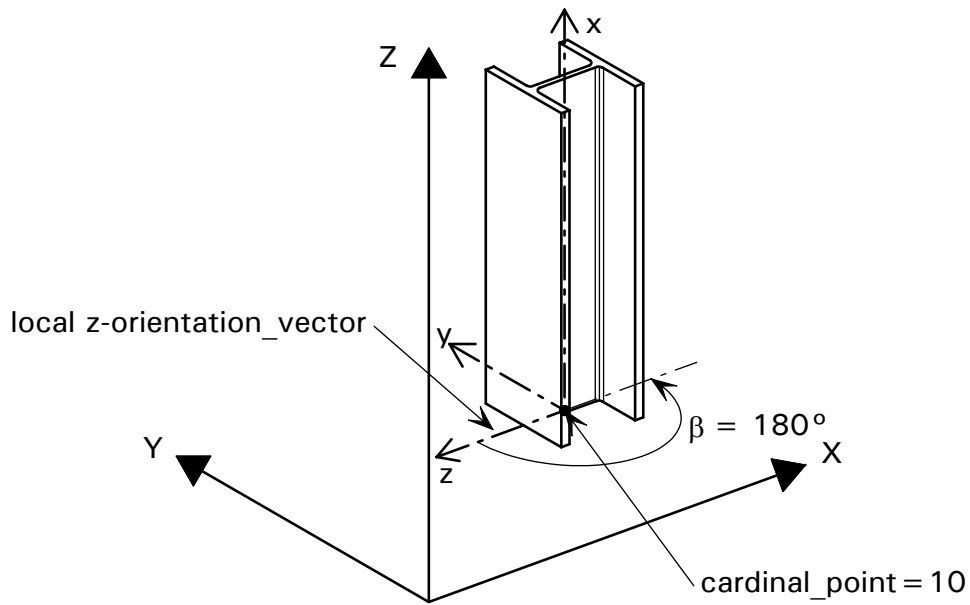
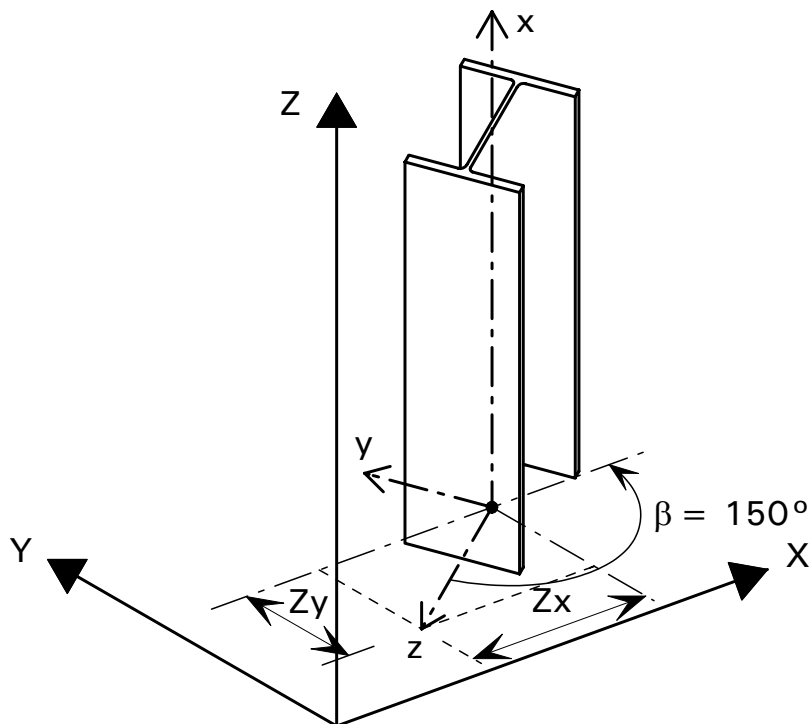


Figure 5.13 *Defining the orientation of a column element (Example 1)*



$$\begin{aligned}\text{orientation vector} &= [Z_x, Z_y, Z_z] \\ &= [-1, 0, 0] \\ &= [\cos \beta, -\sin \beta, 0]\end{aligned}$$

Figure 5.14 *Defining the orientation of a column element (Example 2)*



$$\begin{aligned}\text{orientation vector} &= [Z_x, Z_y, Z_z] \\ &= [-0.866, -0.5, 0]\end{aligned}$$

Figure 5.15 *Defining the orientation of a column element (Example 3)*

Notes:

New for CIS/2, but partly covered by ELEMENT in CIS/1.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.23 element_eccentricity**Entity definition:**

Specification of a 'rigid link'; i.e. the offset of the element from its connecting node. This entity is used to model elements whose ends are eccentric from their connecting nodes by some specified amount. Default values can be defined to specify the situation where nodes connect elements at the centre of the 'element end' (i.e. concentric connections).

The eccentricity is measured from the node to the 'defining point' of the section in the direction of the Global axes in which the nodes are defined. The eccentricity is defined independently of the orientation of the element's local axes. This is illustrated in Figure 5.16.

For an element_curve_simple and an element_curve_complex, the defining point is defined as the specified cardinal point of the section profile. It should be noted that the cardinal point of the section profile may be different from its geometric centroid. Only when the cardinal point of the section_profile is given a value of 10 do the two positions coincide.

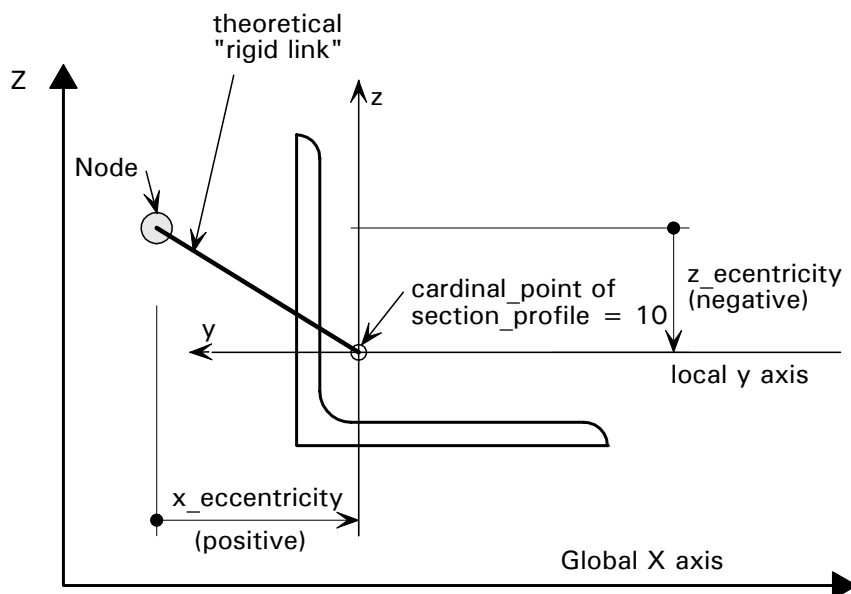


Figure 5.16 *Defining element eccentricity for linear elements*

For an element_surface, the eccentricity is measured to the mid-depth of the through thickness of the surface at the vertex defined by the element_node_connectivity (see Figure 5.19).

For an element_volume, the eccentricity is measured to the vertex of the volume defined by the element_node_connectivity (see Figure 5.20).

There will be one instance of element_eccentricity created for each rigid link defined. Instances may be reused; i.e. several instances of element_node_connectivity may reference the same instance of element_eccentricity.

Where both ends of a one-dimensional element are eccentric, each instance of `element_node_connectivity` associated with that element will refer to an instance of `element_eccentricity` (see Figure 5.18 and Figure 5.19). For example, if a beam is 'cut-back' from its supporting columns, each end of the element that represents that beam will be eccentric. In that case, the two instances of `element_eccentricity` are 'mirrored offsets', and will be in opposite directions.

EXPRESS specification:

*)

ENTITY `element_eccentricity`;

`element_eccentricity_name` : label;

`x_eccentricity` : OPTIONAL length_measure_with_unit;

`y_eccentricity` : OPTIONAL length_measure_with_unit;

`z_eccentricity` : OPTIONAL length_measure_with_unit;

INVERSE

`eccentric_connectivities`: SET [1:?] OF `element_node_connectivity` FOR `eccentricity`;

WHERE

 WRE8 : EXISTS (`x_eccentricity`) OR
 EXISTS (`y_eccentricity`) OR
 EXISTS (`z_eccentricity`);

END_ENTITY;

(*)

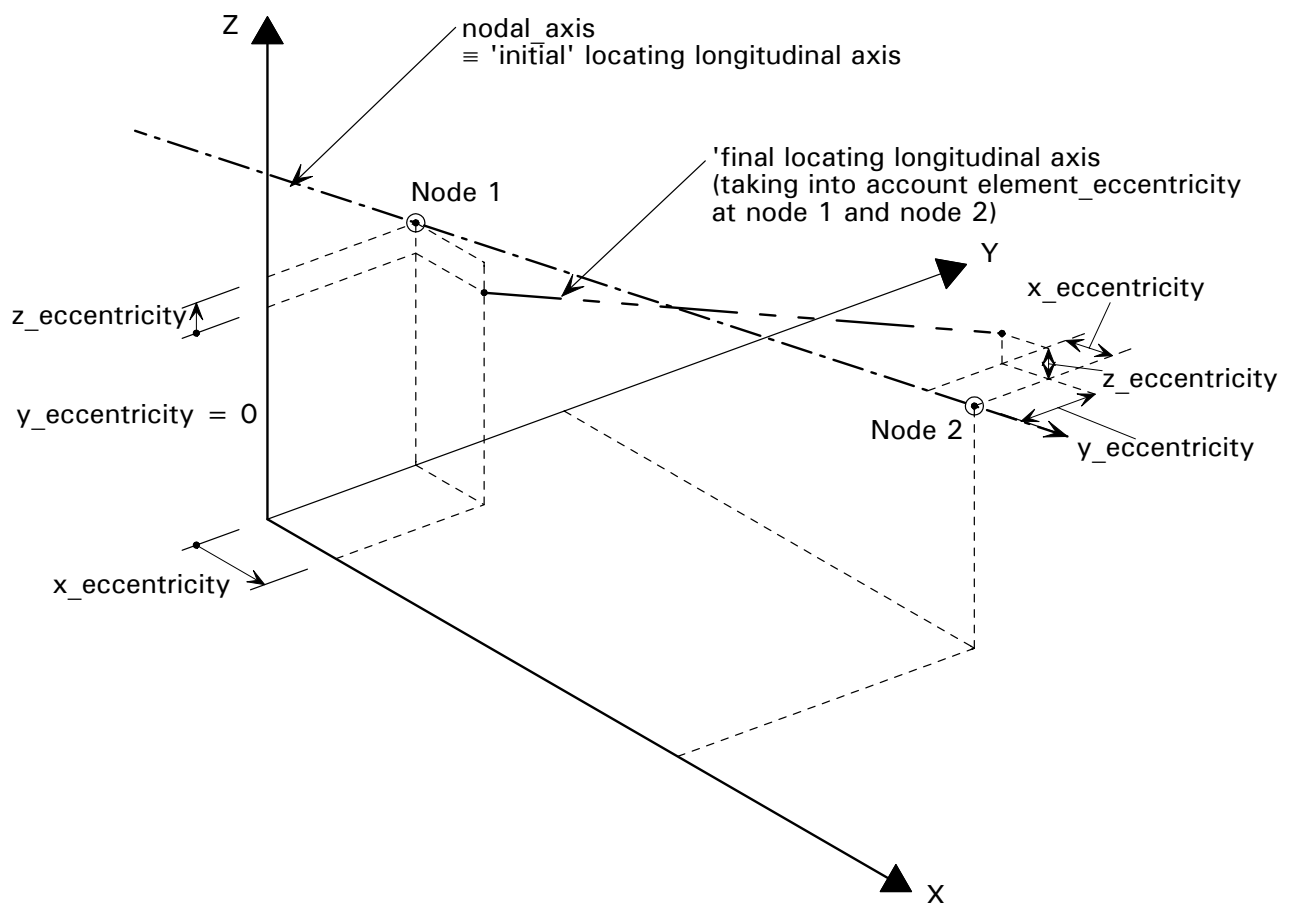


Figure 5.17 *The effect of `element_eccentricity` on the definition of the locating longitudinal axis of a beam element*

Attribute definitions:***element_eccentricity_name***

A short human-readable alphanumerical reference for the element_eccentricity.

x_eccentricity

Declares the first instance of length_measure_with_unit associated with (and defining) the element_eccentricity, which provides the numerical value - with a specified unit - of the eccentricity in the Global X-direction of the element 'end' from the connecting node. This linear dimension is measured in the Global X direction from the connecting node to the defining point of the element. A positive value indicates a displacement of the 'element end' in the direction of the Global X axis.

A zero or NULL value describes the situation where elements meet concentrically; i.e. the connection between element and node occurs at the centre of the element 'end'.

For an element_curve, the specified cardinal point is the defining point of the element, and thus the element's x-axis at the element's end lies on the cardinal point of the section_profile. Then the x_eccentricity is the distance from the connecting node to the cardinal point of the section_profile at the end of the element in the Global X direction.

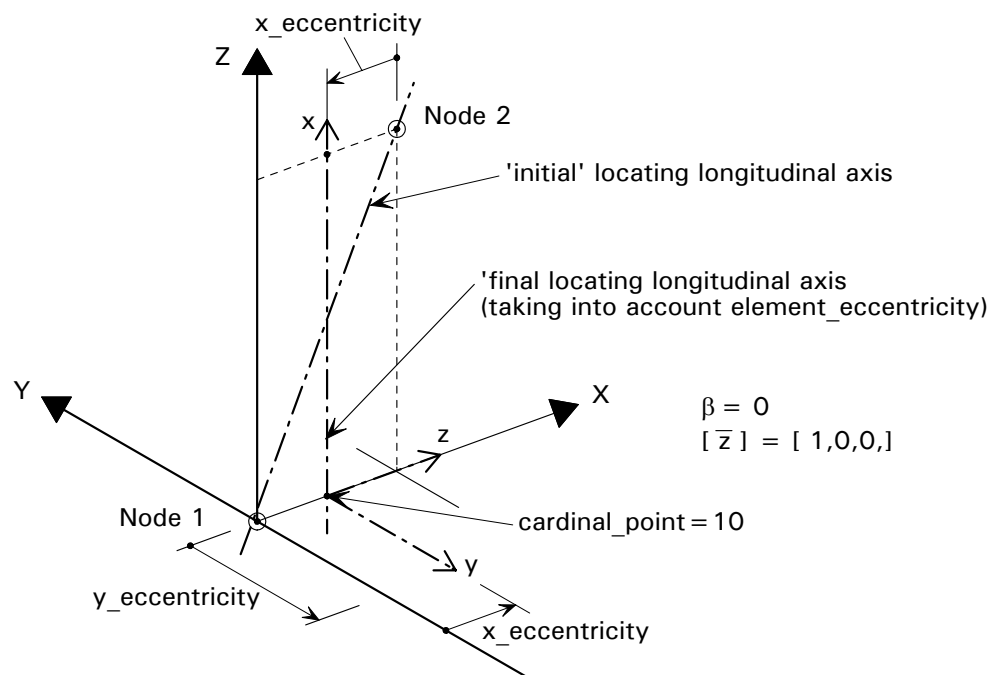


Figure 5.18 *The effect of element_eccentricity on the definition of the locating longitudinal axis of a column element*

y_eccentricity

Declares the second instance of length_measure_with_unit associated with (and defining) the element_eccentricity, which provides the numerical value - with a specified unit - of the eccentricity in the Global Y-direction of the element 'end' from the connecting node. This linear dimension is measured in the Global Y direction from the connecting node to the defining point of the element. A positive value indicates a displacement of the 'element end' in the direction of the Global Y axis. A zero or NULL value describes the situation where elements meet concentrically; i.e. the connection between element and node occurs at the centre of the element 'end'.

For an *element_curve*, the specified cardinal point is the defining point of the element, and thus the element's x-axis at the element's end lies on the cardinal point of the section_profile. Then the *y_eccentricity* is the distance from the connecting node to the cardinal point of the section_profile at the end of the element in the Global Y direction.

z_eccentricity

Declares the third instance of *length_measure_with_unit* associated with (and defining) the *element_eccentricity*, which provides the numerical value - with a specified unit - of the eccentricity in the Global Z-direction of the element 'end' from the connecting node. This linear dimension is measured in the Global Z direction from the connecting node to the defining point of the element. A positive value indicates a displacement of the 'element end' in the direction of the Global Z axis.

A zero or NULL value describes the situation where elements meet concentrically; i.e. the connection between element and node occurs at the centre of the element 'end'.

For an *element_curve*, the specified cardinal point is the defining point of the element, and thus the element's x-axis at the element's end lies on the cardinal point of the section_profile. Then the *z_eccentricity* is the distance from the connecting node to the cardinal point of the section_profile at the end of the element in the Global Z direction.

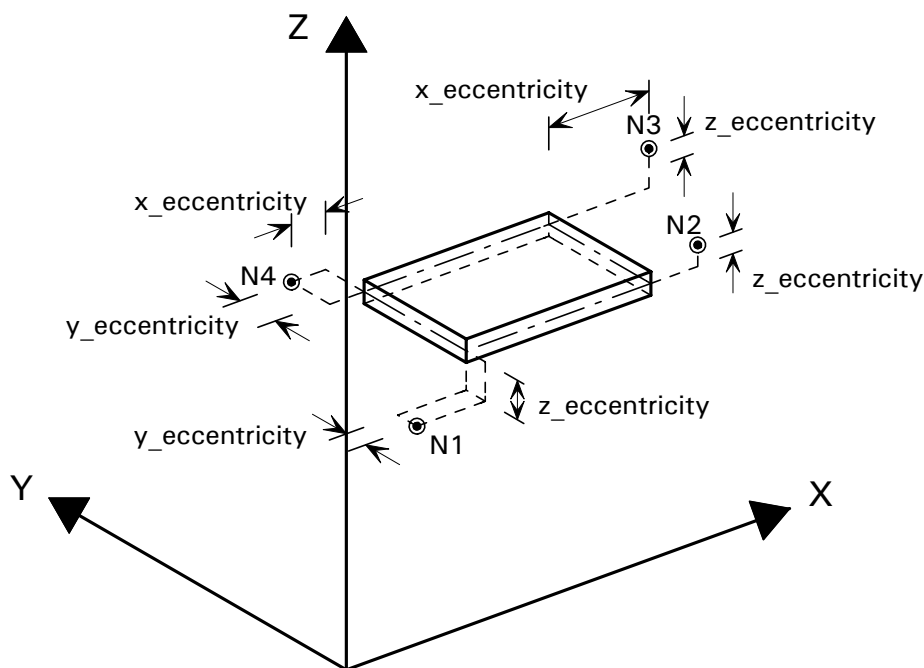


Figure 5.19 *Defining element_eccentricity for planar elements*

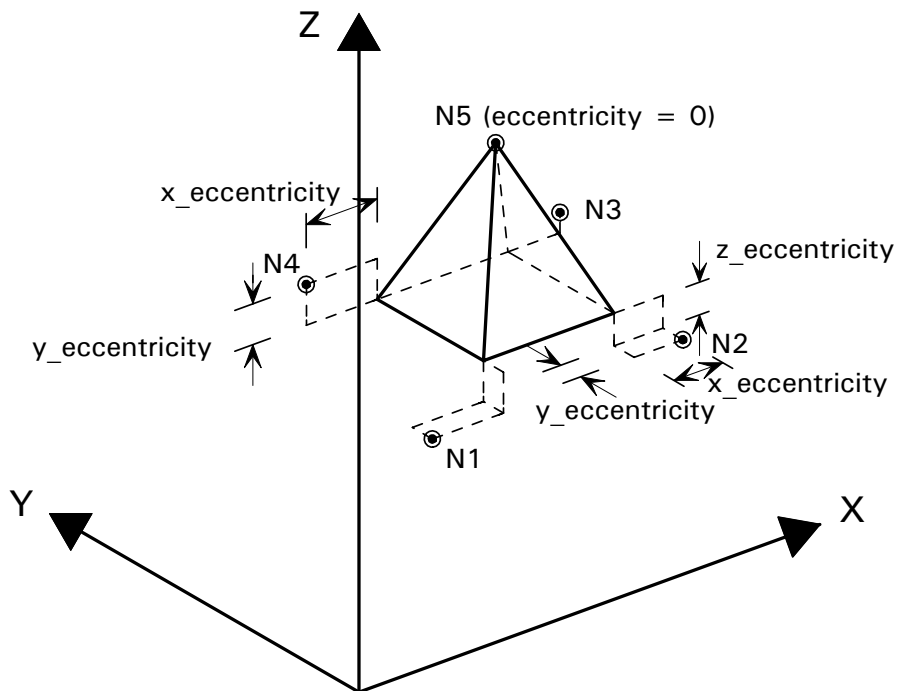


Figure 5.20 *Defining element_eccentricity for cubic elements*

Formal propositions:

eccentric_connectivities

There must be at least one instance of `element_node_connectivity` associated with the `element_eccentricity`. Thus, the `element_eccentricity` depends for its existence on `element_node_connectivity`.

WRE8

At least one of the attributes `x_eccentricity`, `y_eccentricity`, or `z_eccentricity` must be given a value. (They cannot all be assigned as NULL.)

Informal propositions:

Although it is possible to create an instance of `element_eccentricity` with all of its values set to zero, such an instance would have little semantic value.

Notes:

Known as `ELEMENT_ECCENTRICITY` in CIS/1.

In some analytical systems, ‘dummy’ elements with infinite stiffness are used to join the ends of ‘real’ elements to their nodes. These ‘rigid links’ must be converted into instances of ‘`element_eccentricity`’.

See diagram 24 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.24 element_mapping

Entity definition:

An association between an element and either a part, a `design_part`, or a `located_part`. This allows the designer to map the analysis data onto the design or detailing data.

EXPRESS specification:

```

*)
ENTITY element_mapping;
    mapped_element : element;
    represented_part : part_select;
END_ENTITY;
(*

```

Attribute definitions:*mapped_element*

Declares the instance of element associated with the element_mapping

represented_part

Declares the instance of part, design_part, or located part associated with the element_mapping. The choice of part is made through the part_select construct.

Informal propositions:

An element_mapping effects a one-to-one relationship between element and part, design_part, or located part. The element may be involved any number (zero, one, or many) instances of element_mapping. Thus, the element can be mapping onto any number of parts. This is another construct to capture the stage in the design process where analytical elements are mapped onto components of the physical structure (or vice versa).

Notes:

New for CIS/2; partially addressed by ELEMENT and DESIGN_PART in CIS/1.

See diagram 38 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.25 element_node_connectivity**Entity definition:**

An association between an element and a node, which defines the connectivity point of an element in relation to a node. It may be thought of as an 'element end' for 1 dimensional elements.

EXPRESS specification:

```

*)
ENTITY element_node_connectivity;
    connectivity_number : INTEGER;
    connectivity_name : label;
    connecting_node : node;
    connecting_element : element;
    eccentricity : OPTIONAL element_eccentricity;
    fixity : OPTIONAL release;
UNIQUE
    URE2 : connecting_node, connecting_element;
WHERE
    WRE9 : NOT( (connectivity_number > 2) AND
        (connecting_element.element_dimensionality < 2) );

```

```

WRE10 : NOT( (connectivity_name <> 'Start Node') AND
(connectivity_number = 1) );
WRE11 : NOT( (connectivity_name <> 'End Node') AND (connectivity_number = 2) AND
(connecting_element.element_dimensionality = 1) );
WRE12 : connecting_node.parent_model :=: connecting_element.parent_model;
END_ENTITY;
(*)

```

Attribute definitions:

connectivity_number

An integer reference for the element_node_connectivity, which may be used to state the position of the connectivity. For example, for a 1 dimensional element, the connectivity_number = 1 for the ‘start node’ of the element and 2 for the ‘end node’ of the element. In 2 and 3 dimensional elements, the connectivity_number = 3 is used to define the element’s local coordinate system.

connectivity_name

A short human-readable alphanumerical reference for the instance of element_node_connectivity; e.g. ‘start node’, ‘end node’, ‘reference node’, etc.

connecting_node

Declares the instance of node associated with the element_node_connectivity. The connectivity shall be associated with one (and only one) node.

connecting_element

Declares the instance of element associated with the element_node_connectivity. The connectivity shall be associated with one (and only one) element.

eccentricity

Declares the instance of element_eccentricity that may be associated with the element_node_connectivity.

fixity

Declares the instance of release that may be associated with the element_node_connectivity.

Formal propositions:

URE2

The combination of references to instances of node and element (made through connecting_node and connecting_element, respectively) shall be unique to the element_node_connectivity. Thus, there can be only one connection between a node and an element.

WRE9

The value of the attribute connectivity_number shall not be greater than 2 if the value of the attribute element_dimensionality of the connecting_element is less than 2. In other words, the connectivity_number must be either 1 or 2 for one-dimensional elements.

WRE10

The attribute connectivity_name shall not be given any value other than “Start Node”, if the connectivity_number has a value of 1. In other words, the instance of

element_node_connectivity with a connectivity_number of 1 is taken to be at the ‘start’ of the element ($x = 0$).

WRE11

The attribute connectivity_name shall not be given any value other than “End Node”, if the connectivity_number has a value of 2, and the element_dimensionality of the connecting_element is equal to 1. In other words, for a one-dimensional element, the instance of element_node_connectivity with a connectivity_number of 2 is taken to be at the ‘end’ of the element ($x = \text{max}$).

WRE12

The instance of analysis_model referenced by the instance of node (referenced by the attribute connecting_node) must be the same instance of analysis_model referenced by the instance of element (referenced by the attribute connecting_element). In other words, the connectivity must be defined for a node and element in the same analysis model.

Informal propositions:

The limits on connectivities depends upon the type of element as follows:

- For zero dimensional elements (element_point) there will be 1 instance of element_node_connectivity.
- For one dimensional elements (element_curve) there will be 2 instances of element_node_connectivity (e.g. a prismatic element will have only 2 ends.)
- For two dimensional elements (element_surface) there will be 3 (or more) instances of element_node_connectivity. (E.g. a triangular element will have 3 vertices defined by the connectivities, while a quadrilateral element will have 4 vertices defined by the connectivities. In the simplest (unwarped) cases, the connecting nodes will lie in the same plane. Where the connecting nodes do not lie in the same plane the quadrilateral element will be warped. Two dimensional elements may also have reference nodes - not in the same plane - defined by other connectivities.)
- For three dimensional elements (element_volume) there will be 4 (or more) instances of element_node_connectivity (E.g. a tetrahedron element will have 4 vertices defined by the connectivities, while a pyramid element will have 5 vertices.)

These limitations are enforced by the WHERE rules in the subtypes of element.

Notes:

Known as ELT_NODE_CONNECTIVITY in CIS/1.

See diagram 24 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.26 element_point

Entity definition:

A type of element that is single-noded and has no shape. Detailed information is specified through the SUBTYPES. Matrices are used to define stationary masses, grounded springs, or grounded dampers.

EXPRESS specification:

*)

ENTITY element_point

ABSTRACT SUPERTYPE OF (ONEOF

```

        (element_point_grounded_damper,
         element_point_stationary_mass,
         element_point_grounded_spring))
SUBTYPE OF (element);
DERIVE
    connectivities : BAG [1:1] OF element_node_connectivity := USEDIN (SELF,
        'STRUCTURAL_FRAME_SCHEMA.
        ELEMENT_NODE_CONNECTIVITY.CONNECTING_ELEMENT');
WHERE
    WRE13 : SELF.element.element_dimensionality = 0;
END_ENTITY;
(*)

```

Attribute definitions:*connectivities*

This attribute derives the (bag of) one instance of `element_node_connectivity` that references this instance of `element_point` (through the attribute `connecting_element`). There shall be only one `element_node_connectivity` associated with this element.

Formal propositions:*ABSTRACT SUPERTYPE*

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

DERIVE

The derived attribute `connectivities` does not appear in the STEP Part 21 file.

WRE13

The dimensionality of the element shall be zero.

Notes:

New for CIS/2. This entity is based on constructs found in STEP Part 104^[28].

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.27 element_point_grounded_damper**Entity definition:**

A type of point element used to represent a nodal response matrix of type grounded damper. The grounded damper matrix is populated as follows:

$$\begin{bmatrix} dt_x & & & & & \\ & dt_y & & & & \\ & & dt_z & & & \\ & & & dr_x & & \\ & & & & dr_y & \\ & & & & & dr_z \end{bmatrix}$$

EXPRESS specification:

```

*)
ENTITY element_point_grounded_damper
SUBTYPE OF (element_point);
    damping_coefficients : ARRAY [1:6] OF REAL;
END_ENTITY;
(*)

```

Attribute definitions:*damping_coefficients*

The numerical values of the damping in different directions. This is populated as follows:

- Array item 1 is the translational damping in the x direction, denoted dt_x in the matrix above.
- Array item 2 is the translational damping in the y direction, denoted dt_y in the matrix above.
- Array item 3 is the translational damping in the z direction, denoted dt_z in the matrix above.
- Array item 4 is the rotational damping about the x direction, denoted dr_x in the matrix above.
- Array item 5 is the rotational damping about the y direction, denoted dr_y in the matrix above.
- Array item 6 is the rotational damping about the z direction, denoted dr_z in the matrix above.

Notes:

New for CIS/2. This entity is based on constructs found in STEP Part 104^[28].

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.28 element_point_grounded_spring**Entity definition:**

A type of point element used to represent a nodal response matrix of type grounded spring. The grounded damper spring is populated as follows:

$$\begin{bmatrix} kt_x & & & & & \\ & kt_y & & & & \\ & & kt_z & & & \\ & & & kr_x & & \\ & & & & kr_y & \\ & & & & & kr_z \end{bmatrix}$$

EXPRESS specification:

```

*)
ENTITY element_point_grounded_spring
SUBTYPE OF (element_point);
    stiffness_coefficients : ARRAY [1:6] OF REAL;
END_ENTITY;
(*)

```

Attribute definitions:*stiffness_coefficients*

The numerical values of the stiffnesses in different directions. This is populated as follows:

- Array item 1 is the translational stiffness in the x direction, denoted kt_x in the matrix above.
- Array item 2 is the translational stiffness in the y direction, denoted kt_y in the matrix above.
- Array item 3 is the translational stiffness in the z direction, denoted kt_z in the matrix above.
- Array item 4 is the rotational stiffness about the x direction, denoted kr_x in the matrix above.
- Array item 5 is the rotational stiffness about the y direction, denoted kr_y in the matrix above.
- Array item 6 is the rotational stiffness about the z direction, denoted kr_z in the matrix above.

Notes:

New for CIS/2. This entity is based on constructs found in STEP Part 104^[28].

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.29 element_point_stationary_mass**Entity definition:**

A type of point element used to represent a nodal response matrix of type stationary mass. The stationary mass matrix is populated as follows:

$$\begin{bmatrix}
 m_x & & & & & \\
 & m_y & & & & \\
 & & m_z & & & \\
 & & & i_{xx} & i_{xy} & i_{xz} \\
 & & & i_{xy} & i_{yy} & i_{yz} \\
 & & & i_{xz} & i_{yz} & i_{zz}
 \end{bmatrix}$$

EXPRESS specification:

```

*)
ENTITY element_point_stationary_mass
SUBTYPE OF (element_point);
    masses : ARRAY [1:3] OF REAL;
    moments_of_inertia : ARRAY [1:3] OF ARRAY [1:3] OF REAL;
END_ENTITY;
(*

```

Attribute definitions:***masses***

The numerical values of the point masses in the different directions. This is populated as follows:

- Array item 1 is the mass acting in the x direction, denoted m_x in the matrix above.
- Array item 2 is the mass acting in the y direction, denoted m_y in the matrix above.
- Array item 3 is the mass acting in the z direction, denoted m_z in the matrix above.

moments_of_inertia

The numerical values of the moments of inertia of a point (denoted i_{xx} to i_{zz} in the matrix above) in the different directions. The values of the inertias should be set to 0.0 if unwanted. The inertias are evaluated about the centre of mass.

Notes:

New for CIS/2. This entity is based on constructs found in STEP Part 104^[28].

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.30 element_surface**Entity definition:**

A type of element connected to 3 or more nodes, whose dimensionality is two, and whose shape is defined by a thickness, a surface and a boundary. (See Figure 5.21.)

In general, the element_surface is not planar; i.e. it is warped. If there are only three nodes (and connectivities) associated with the element_surface, then by definition, the element will be planar. The special case of a multi-noded planar element is represented by the SUBTYPE element_surface_plane.

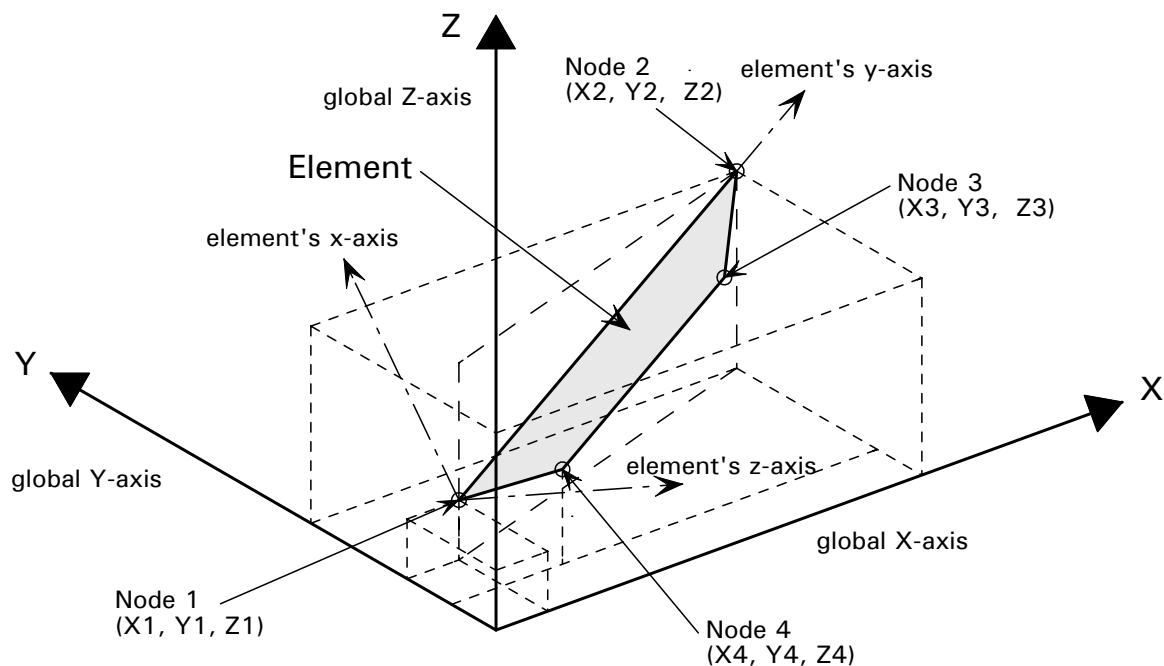


Figure 5.21 *Defining surface elements*

While most steel structures consist of members that can be idealized as one-dimensional curved or line elements, structural components having significant dimensions in two directions (such as plates) are also encountered frequently. In steel structures, plates occur as components of I, H, T, or channel sections as well as in structural hollow sections. When these components are required to be analysed in detail, they should be represented by instances of `element_surface`. Sheets are also used to enclose lift shaft walls or cladding in framed structures.

When using a Cartesian coordinate system, the surface element is defined in the yz -plane with the thickness lying in the x -axis; i.e., the x -axis is perpendicular to the plane of the 'plate'. The first connecting node is initially taken as the origin of the element's local coordinate system. (This is based on the assumption that initially the first connectivity (declared by the instance of `element_node_connectivity` having a `connectivity_number` = 1) is taken to lie on the node; i.e. no eccentricity has been applied.) The element's y -axis passes from the first connectivity to the second, and thus lies initially between the first 2 nodes. The third connectivity defines a point on the yz -plane. The origin and the axes will shift if values are assigned to the `element_eccentricity`. The fourth and subsequent connectivity define the other vertices of the `element_surface`. These may (but need not) lie in the same plane.

The thickness is distributed equally on either side of the surface; i.e. the surface represents the centroidal plane of the element. It is assumed that the thickness of the element is small enough to ensure that the element behaves as a '2D plate' rather than as a '3D solid'. A plate whose thickness is large enough to require consideration of 'through-thickness' effects should be modelled as an `element_volume`. Instances of `element_volume` should also be used to represent plates that do not have a constant thickness.

EXPRESS specification:

*)

ENTITY `element_surface`

SUPERTYPE OF (ONEOF (`element_surface_simple`, `element_surface_complex`))

```

SUBTYPE OF (element);
  thickness : positive_length_measure_with_unit;
DERIVE
  connectivities : SET [3:?] OF element_node_connectivity := bag_to_set (USEDIN (SELF,
    'STRUCTURAL_FRAME_SCHEMA.
    ELEMENT_NODE_CONNECTIVITY.CONNECTING_ELEMENT'));
WHERE
  WRE14 : SELF\element.element_dimensionality = 2;
  WRE15 : SELF\element.parent_model.coordinate_space_dimension > 1;
END_ENTITY;
(*)

```

Attribute definitions:

thickness

Declares the instance of `positive_length_measure_with_unit` associated with (and defining) the `element_surface`, which provides the numerical value - measured in accordance with an appropriate unit system - of the thickness of the element. When using a Cartesian coordinate system, the element is defined as lying in the *yz*-plane and so the thickness will be the dimension in the *x*-direction. It is assumed that the thickness is constant throughout the surface element and is distributed equally on either side of the defined surface, i.e. the surface represents the centroidal plane of the element.

connectivities

This attribute derives the set of three or more distinct instances of `element_node_connectivity` that reference this instance of `element_surface` (through the attribute `connecting_element`). There is no upper limit to the number of connectivities associated with this surface element, although each instance must be `element_node_connectivity` different (no repetition allowed).

Formal propositions:

DERIVE

The derived attribute `connectivities` does not appear in the STEP Part 21 file.

WRE14

The dimensionality of the element shall be equal to two.

WRE15

The value assigned to the attribute `coordinate_space_dimension` of the instance of `analysis_model` (referenced by the attribute `parent_model`) shall be greater than one. In other words, a surface element must be located within either a 2D or a 3D analysis model.

Informal propositions:

Although it could be instanced on its own, it is expected that this entity would be instanced with one of its SUBTYPEs. If it is instanced on its own, then the element has no defined shape, or the shape is implied from the element's connectivity.

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.31 element_surface_complex

Entity definition:

A type of two-dimensional element whose shape is defined using explicit geometry.

EXPRESS specification:

*)

ENTITY element_surface_complex

SUPERTYPE OF (ONEOF(element_surface_plane, element_surface_profiled))

SUBTYPE OF (element_surface);

surface_definition : surface;

END_ENTITY;

(*

Attribute definitions:

surface_definition

Declares the instance of surface associated with the element_surface_complex. Surface is a STEP Part 42 entity, and defines the shape of the element.

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.32 element_surface_plane

Entity definition:

A type of two-dimensional element whose shape is defined within a plane. All the connectivities of the element (declared by associated instances of element_node_connectivity) lie in the same plane. If zero or NULL values have been assigned to the associated element eccentricities, then all the connecting nodes also lie in the same plane.

EXPRESS specification:

*)

ENTITY element_surface_plane

SUBTYPE OF (element_surface_complex);

WHERE

WRE16 : 'STRUCTURAL_FRAME_SCHEMA.PLANE' IN TYPE OF
(SELF\element_surface_complex.surface_definition);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own, although it does inherit several from its SUPERTYPE. This purpose of this entity is to constrain the use of the SUPERTYPE entity element_surface.

Formal propositions:**WRE16**

The type of surface used to define the shape of the element shall be a plane (STEP Part 42 entity).

Notes:

New for CIS/2.

The entity plane is defined in STEP Part 42.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.33 element_surface_profiled**Entity definition:**

A type of two-dimensional element that has a repeating profile that extends into the third (through-thickness) dimension. For example, a profiled sheet.

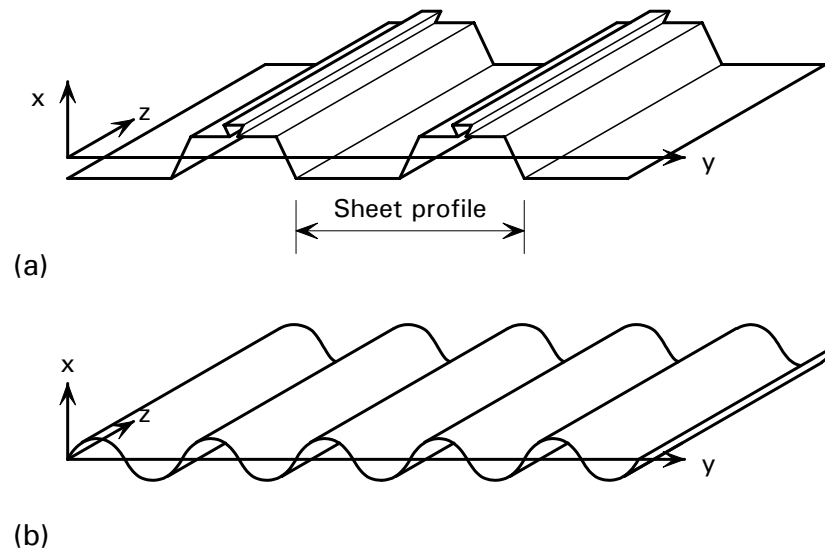


Figure 5.22 *Examples of element_surface_profiled*

EXPRESS specification:

```
*)
ENTITY element_surface_profiled
  SUBTYPE OF (element_surface_complex);
    profile : curve;
END_ENTITY;
(*
```

Attribute definitions:**profile**

Declares the instance of curve associated with the *element_surface_profiled*. (See Figure 5.22.) Curve is a STEP Part 42 entity, and defines the shape of the 2D element in the through thickness (x-direction).

Notes:

New for CIS/2.

The entity curve is defined in STEP Part 42.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

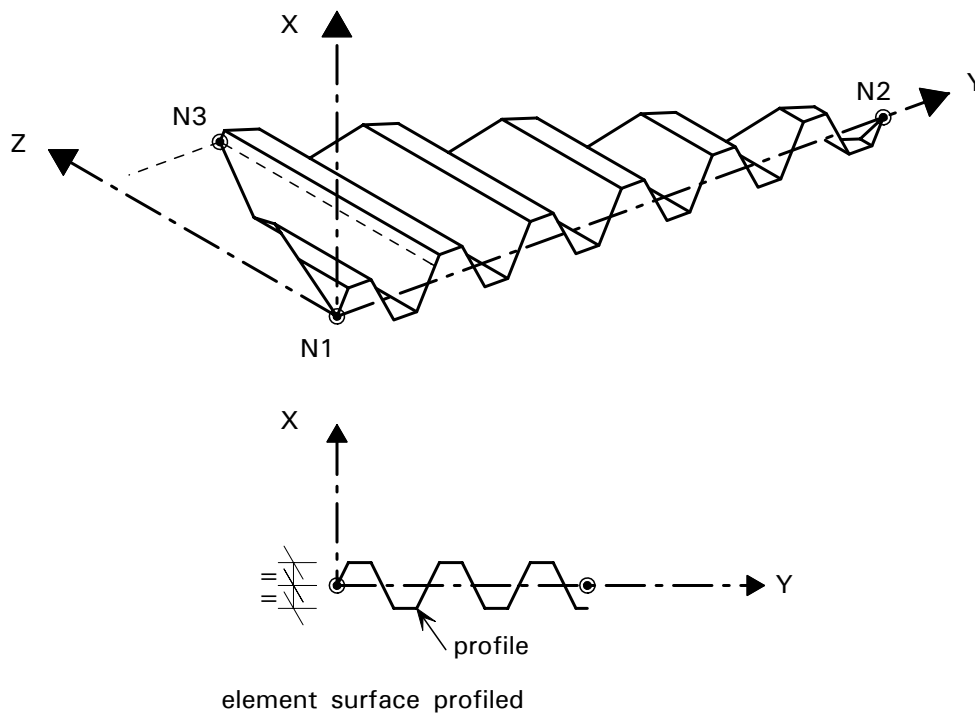


Figure 5.23 *Defining an element_sheet_profiled*

5.3.34 element_surface_simple

Entity definition:

A type of two-dimensional element whose shape is defined as either a triangle or quadrilateral. For a triangular element, all the connectivities of the element (declared by associated instances of element_node_connectivity) lie in the same plane. If zero or NULL values have been assigned to the associated element eccentricities, then all the connecting nodes also lie in the same plane. For a quadrilateral element, the connectivities of the element may (but need not) lie in the same plane. If they do not lie in the same plane, then the quadrilateral element is warped. In this case, no information is provided on the shape of the warping. If this is required, then the element element_surface_complex should be instance in preference to this entity

EXPRESS specification:

*)

```
ENTITY element_surface_simple
SUBTYPE OF (element_surface);
  shape : element_surface_shape;
  assumption : plane_stress_or_strain;
END_ENTITY;
```

(*)

Attribute definitions:*shape*

Specifies the type of simple surface assumed for the `element_surface_simple`, as either a triangle or quadrilateral. (See also type definition for `element_surface_shape`.) If the shape is defined as a triangle, the element has 3 connectivities and there shall be 3 instances of `element_node_connectivity` for each element. If the shape is defined as a quadrilateral, the element has 4 connectivities and there shall be 4 instances of `element_node_connectivity` for each element. The shape need not be a regular polygon. The shape need not be a plane surface, although in the simplest cases, it will be.

assumption

Specifies the type of analytical assumption made for the `element_surface_simple` (See also type definition for `plane_stress_or_strain`.)

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.35 element_volume**Entity definition:**

A type of element that is connected to 4 or more nodes and whose dimensionality is three. The element may or may not be solid, depending on the SUBTYPE used. Simple solid volume elements will be represented using the SUBTYPE `element_volume_simple`. More complicated volume elements, that may be solids or shells, elements will be represented using the SUBTYPE `element_volume_complex`. The definition of the shape of the element is provided by the SUBTYPEs.

The element has an implied 3D coordinate system. The first connecting node is initially taken as the origin of the element's local coordinate system. (This is based on the assumption that initially the first connectivity (declared by the instance of `element_node_connectivity` having a `connectivity_number` = 1) is taken to lie on the node; i.e. no eccentricity has been applied.)

When using a Cartesian coordinate system, the element's y-axis is defined as passing from the first connectivity to the second, and thus lies initially between the first 2 nodes. The third connectivity defines a point on the yz-plane with a positive value of its z coordinate. This may, but need not, lie on the z-axis. The origin and the axes will shift if values are assigned to the `element_eccentricity`. It follows that the x-axis is perpendicular to the yz-plane using a right-handed rule.

EXPRESS specification:

*)

ENTITY `element_volume`

ABSTRACT SUPERTYPE OF (ONEOF(`element_volume_simple`,
`element_volume_complex`))

SUBTYPE OF (`element`);

DERIVE

```

connectivities : SET [4:?] OF element_node_connectivity := bag_to_set (USEDIN (SELF,
'Structural_Frame_Schema.
ELEMENT_NODE_CONNECTIVITY.CONNECTING_ELEMENT'));
WHERE
WRE17 : SELF\element.element_dimensionality = 3;
WRE18 : SELF\element.parent_model.coordinate_space_dimension = 3;
END_ENTITY;
(*)

```

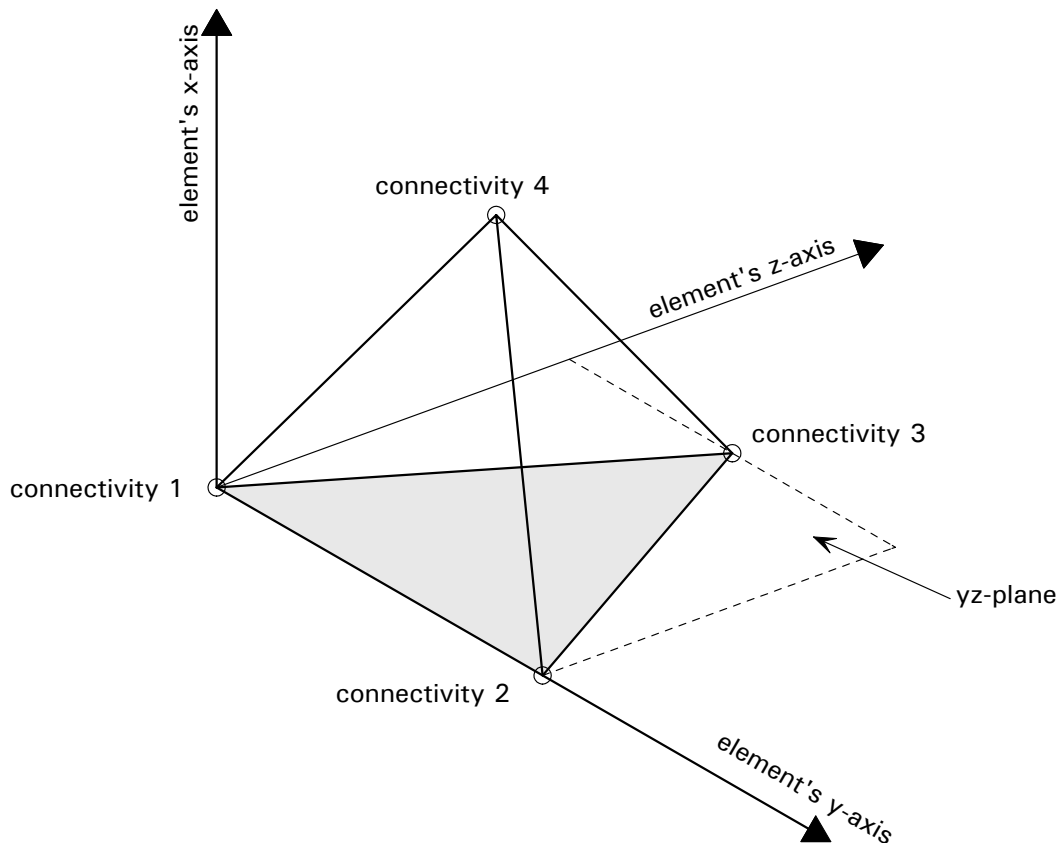


Figure 5.24 *Defining volume elements*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE element. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

connectivities

The attribute derives the set of four or more separate instances of element_node_connectivity that reference this instance of element_volume (through the attribute connecting_element). There is no upper limit to the number of connectivities that reference this volume element, but they must all be different (no repetition allowed).

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

DERIVE

The derived attribute connectivities does not appear in the STEP Part 21 file.

WRE17

The dimensionality of the element shall be equal to three.

WRE18

The value assigned to the attribute `element_dimensionality` (inherited from the SUPERTYPE element) shall = 3. In other words, the volume element is three-dimensional.

WRE19

The value assigned to the attribute `coordinate_space_dimension` of the instance of `analysis_model` referenced by the attribute `parent_model` (inherited from the SUPERTYPE element) shall = 3. In other words, volume elements can only be defined in three-dimensional analysis models.

Informal propositions:

There will be 4 or more instances of `element_node_connectivity` associated with the `element_volume`.

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.36 element_volume_complex

Entity definition:

A type of three-dimensional element whose shape is defined using the constructs for explicit geometry taken from STEP Part 42.

EXPRESS specification:

```
*)
ENTITY element_volume_complex
  SUBTYPE OF (element_volume);
    shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:

shape

Declares the instance of `shape_representation_with_units` associated with the `element_volume_complex`. The LPM entity `shape_representation_with_units` is a specialized SUBTYPE of the STEP Part 41 entity `shape_representation`. The definition of the elements shape requires the use of a set of `geometric_representation_items`, or a set of `topological_representation_items` from STEP Part 42.

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.37 element_volume_simple

Entity definition:

A type of solid three-dimensional element whose shape is defined as either a hexahedron, a wedge, a tetrahedron, or a pyramid. (Shells and solids containing voids must be modelled using element_volume_complex. Note, a cube is considered to be a special case of a hexahedron.

EXPRESS specification:

```
*)
ENTITY element_volume_simple
  SUBTYPE OF (element_volume);
    shape : element_volume_shape;
END_ENTITY;
(*
```

Attribute definitions:

shape

Specifies the type of shape assumed for the element_volume_simple. (See also type definition for element_volume_shape and Figure 5.3.)

Notes:

New for CIS/2.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

The enumeration names of the type element_volume_shape have been modified for the 2nd Edition.

5.3.38 element_with_material

Entity definition:

A type of element that has its material defined. This is, in effect, an association between an element and a material.

EXPRESS specification:

```
*)
ENTITY element_with_material
  SUBTYPE OF (element);
    material_definition : material;
END_ENTITY;
(*
```

Attribute definitions:

material_definition

Declares the instance of material associated with this element_with_material, which is used to defined the material and the material properties of the element.

Informal propositions:

Because the SUPERTYPE is an ANDOR, then any type of element may be instanced with the element_with_material.

Although this entity may be populated on its own, it is more likely to be instanced with one of its sibling SUBTYPES – `element_point`, `element_curve`, `element_surface`, or `element_volume`. In this case, the complex entity instance is mapped to a STEP Part 21 file using external mapping.

Notes:

New for CIS/2; addressed by ELEMENT in CIS/1.

See diagram 25 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.39 node

Entity definition:

A theoretical point in space, used in structural analysis to define where one or more elements converge, and where the structure (as represented by the analytical model) may be restrained. Nodes are defined by a point within the global coordinate system of the analysis model.

EXPRESS specification:

*)

ENTITY node;

node_name : label;

node_coords : point;

restraints : OPTIONAL boundary_condition;

parent_model : analysis_model;

UNIQUE

URN1 : node_name, parent_model;

URN2 : node_coords, parent_model;

WHERE

WRN1 : node_coords.dim = parent_model.coordinate_space_dimension;

END_ENTITY;

(*

Attribute definitions:

node_name

A short alphanumeric (or simply numeric) reference for the node.

node_coords

Declares the instance of point associated with the node, and thus locating the node. The node is defined in the global coordinate system of the analytical model. In a Cartesian coordinate system, the attributes of point provide the x, y, and z coordinates of the node.

restraints

Declares the instance of boundary_condition that may be associated with the node. The boundary_condition defines the support conditions (e.g. pinned, fixed, or spring support) for the node assumed for the analysis. The node can have at most one instance of boundary_condition.

parent_model

Declares the instance of *analysis_model* associated with the node. The node cannot exist without a parent model and it is assumed that the node belongs to the *analysis_model* referenced here.

Formal propositions:*URN1*

The combination of the *node_name* and the *parent_model* shall be unique to the node. Thus, within an analytical model, nodes must all have different names.

URN2

The combination of the *node_coords* and the *parent_model* shall be unique to the node. Thus, within an analytical model, nodes must all have different points defining their location. (It should be noted that the uniqueness applies to the instance of point, rather than to the values of the attributes of point.)

WRN1

The attribute *coordinate_space_dimension* of the instance of *analysis_model* referenced by the attribute *parent_model* must have the same value as that assigned to the attribute *dim* of the instance of point referenced by the attribute *node_coords*. In other words, the position of the node must be defined by a point in a coordinate space appropriate to the analysis model.

Notes:

Known as NODE in CIS/1.

See diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.40 node_dependency**Entity definition:**

An association between two distinct nodes, where one acts as the ‘master’ to the other. The dependency is generic, such that every aspect of the ‘master node’ must be reflected in the ‘slave node’, and any changes made to the master node must be reflected in the slave node.

The dependency is one-way; that is, the slave is dependent on the master, but the master is independent of the slave. Where the nodal dependency is mutual, then two instances of *nodal_dependency* are required.

A node may be the master of any number of other nodes. A node can also be the slave of any number of nodes. In this situation, the conditions for all the master nodes must be reflected in the slave node.

The dependency can be inherited, such that a node that acts as a ‘slave’ to a ‘master node’ can inherit the conditions of the node that acts as a ‘master’ to that ‘master node’. (CIS/2 compatible systems should ensure that the inherited dependencies are mutually compatible.)

This entity is also intended to capture the situation where analysis models do not represent the complete structure. The missing pieces of the structure (e.g. a concrete slab) are simply represented by the nodal dependency. For example, if the master node is displaced by 10mm in the x-direction, then the slave node should also be displaced by

10mm in the x-direction. This assumes that the relative stiffness of the concrete slab is so large that its distortion is small compared to distortion of the steel frame.

EXPRESS specification:

```
*)
ENTITY node_dependency;
    master_node : node;
    slave_node : node;
    dependency_description : OPTIONAL text;
WHERE
    WRN2 : master_node :<>: slave_node;
END_ENTITY;
(*
```

Attribute definitions:

master_node

Declares the first instance of node associated with this instance of nodal_dependency, and which is deemed to act as the ‘master node’.

slave_node

Declares the second instance of node associated with this instance of nodal_dependency, and which is deemed to act as the ‘slave node’. The ‘slave node’ is dependent upon the master. Any changes in the conditions defined for the master node must be reflected in the definition for the slave node.

dependency_description

An optional free text description of the type of the dependency. It may describe why the dependency has been created, and what it represents.

Formal propositions:

WRN2

The instances of node referenced by the attributes master_node and slave_node must be different. In other words, a node cannot be dependent upon itself.

Notes

New for CIS/2.

See Diagram 23 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.41 release

Entity definition:

Static condition defining the degrees of freedom where the element meets a node; i.e. the end fixity of an element; e.g. ‘pinned’, ‘fixed’ etc. The release must be given a name and may be given a description. The release may be linear or non-linear depending on the SUBTYPE instanced.

EXPRESS specification:

```
*)
ENTITY release
ABSTRACT SUPERTYPE OF (ONEOF
```

```

        (release_logical, release_spring_linear, release_spring_non_linear));
    release_name : label;
    release_description : OPTIONAL text;
INVERSE
    release_for_element_nodes : SET [1:?] OF element_node_connectivity
        FOR fixity;
END_ENTITY;
(*)

```

Attribute definitions:

release_name

A short alphanumeric reference for the release.

release_description

A free text description of the release.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

release_for_element_nodes

The release shall apply to a set of one or more element_node_connectivity; i.e. the release cannot exist without it being referenced by at least one element_node_connectivity.

Notes:

Known as RELEASE in CIS/1; SUBTYPES added.

See diagram 24 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.42 release_logical

Entity definition:

A type of release where six degrees of freedom are given logical values (true, false or unknown). This is a simple representation of an end fixity. Analysis models defined in three dimensions should have instances of element_node_connectivity that reference instances of release_logical that have all of their attributes assigned values as either TRUE or FALSE. Those defined in one plane only (i.e. out of plane conditions ignored) should have instances of element_node_connectivity that reference instances of release_logical with some of their attributes assigned a value of UNKNOWN.

Each of the six attributes may be assigned a value of either TRUE, FALSE, or UNKNOWN. This corresponds to an end fixity for the element (defined in the local coordinate system of the element) that is either 'free', 'fixed', or 'unknown'. The value will be 'unknown' if that particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model.

EXPRESS specification:

```

*)
ENTITY release_logical
SUBTYPE OF (release);
    release_axial_force : LOGICAL;
    release_y_force : LOGICAL;
    release_z_force : LOGICAL;
    release_torsional_moment : LOGICAL;
    release_y_bending_moment : LOGICAL;
    release_z_bending_moment : LOGICAL;
END_ENTITY;
(*)

```

Attribute definitions:*release_axial_force*

Declares the value of the degree of freedom for the release for linear displacement in the x direction, in the element's local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the x direction is unrestrained (i.e. 'released') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

release_y_force

Declares the value of the degree of freedom for the release for linear displacement in the y direction, in the element's local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the y direction is unrestrained (i.e. 'released') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

release_z_force

Declares the value of the degree of freedom for the release for linear displacement in the z direction, in the element's local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If linear displacement in the z direction is unrestrained (i.e. 'released') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

release_torsional_moment

Declares the value of the degree of freedom for the release for rotation about the x axis of the element's local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the x axis is unrestrained (i.e. 'released') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

release_y_bending_moment

Declares the value of the degree of freedom for the release for rotation about the y axis of the element's local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the y axis is unrestrained (i.e. 'released') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the

value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

release_z_bending_moment

Declares the value of the degree of freedom for the release for rotation about the z axis of the element's local coordinate system. The value may be either TRUE, FALSE, or UNKNOWN. If rotation about the z axis is unrestrained (i.e. 'released') this attribute is assigned the value TRUE. If it is restrained (i.e. 'fixed'), this attribute is assigned the value FALSE. If it is unknown (because the degree of freedom is not being considered) this attribute is assigned the value UNKNOWN.

Informal propositions:

Although it is possible to assign a value of UNKNOWN to all of the attributes, this is not recommended, as the populated instance would have little semantic value.

Notes

New for CIS/2.

See Diagram 24 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.43 release_spring_linear

Entity definition:

A type of release, providing the values of the stiffnesses of a linear spring end fixity, where the values of the spring stiffnesses are assumed to be constant regardless of the load condition. Taken together, these values define the degrees of freedom where the element meets a node; i.e. the end fixity of an element; e.g. 'pinned', 'fixed' etc.

EXPRESS specification:

*)

ENTITY release_spring_linear

SUPERTYPE OF (release_warping)

SUBTYPE OF (release);

release_axial_force : OPTIONAL linear_stiffness_measure_with_unit;

release_y_force : OPTIONAL linear_stiffness_measure_with_unit;

release_z_force : OPTIONAL linear_stiffness_measure_with_unit;

release_torsional_moment : OPTIONAL rotational_stiffness_measure_with_unit;

release_y_bending_moment : OPTIONAL rotational_stiffness_measure_with_unit;

release_z_bending_moment : OPTIONAL rotational_stiffness_measure_with_unit;

WHERE

WRR21 : EXISTS (release_axial_force) OR

EXISTS (release_y_force) OR

EXISTS (release_z_force) OR

EXISTS (release_torsional_moment) OR

EXISTS (release_y_bending_moment) OR

EXISTS (release_z_bending_moment);

END_ENTITY;

(*)

Attribute definitions:***release_axial_force***

Declares the first instance of `linear_stiffness_measure_with_unit` that may be used to define the `release_spring_linear`. This provides the numerical value with an appropriate unit of the release (or end fixity of the element) in the linear x-direction, in the element's local coordinate system; (i.e. the restraint against axial force). If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a 'free' release, this attribute should be assigned a value equivalent to a zero spring stiffness.

release_y_force

Declares the second instance of `linear_stiffness_measure_with_unit` that may be used to define the `release_spring_linear`. This provides the numerical value with an appropriate unit of the release (or end fixity of the element) in the linear y-direction, in the element's local coordinate system; (i.e. the restraint against in-plane force). If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a 'free' release, this attribute should be assigned a value equivalent to a zero spring stiffness.

release_z_force

Declares the third instance of `linear_stiffness_measure_with_unit` that may be used to define the `release_spring_linear`. This provides the numerical value with an appropriate unit of the release (or end fixity of the element) in the linear z-direction, in the element's local coordinate system; (i.e. the restraint against out-of-plane force). If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a 'free' release, this attribute should be assigned a value equivalent to a zero spring stiffness.

release_torsional_moment

Declares the first instance of `rotational_stiffness_measure_with_unit` that may be used to define the `release_spring_linear`. This provides the numerical value with an appropriate unit of the release (or end fixity of the element) about the x- axis, in the element's local coordinate system; (i.e. the restraint against torsional bending). If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a 'free' release, this attribute should be assigned a value equivalent to a zero spring stiffness.

release_y_bending_moment

Declares the second instance of `rotational_stiffness_measure_with_unit` that may be used to define the `release_spring_linear`. This provides the numerical value with an

appropriate unit of the release (or end fixity of the element) about the y-axis, in the element's local coordinate system; (i.e. the restraint against major axis bending). If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ release, this attribute should be assigned a value equivalent to a zero spring stiffness.

release_z_bending_moment

Declares the third instance of `rotational_stiffness_measure_with_unit` that may be used to define the `release_spring_linear`. This provides the numerical value with an appropriate unit of the release (or end fixity of the element) about the z-axis, in the element's local coordinate system; (i.e. the restraint against minor axis bending). If provided, the spring stiffness value must be greater than or equal to zero. (See definition of `linear_stiffness_measure`.) This attribute should remain unpopulated (and appear in a STEP Part 21 file with a null value - \$) only when this particular degree of freedom is not being considered – as may be the case for a two-dimensional analysis model. When modelling a ‘free’ release, this attribute should be assigned a value equivalent to a zero spring stiffness.

Formal propositions

WRR21

At least one of the attributes `release_axial_force`, `release_y_force`, `release_z_force`, `release_torsional_moment`, `release_y_bending_moment`, or `release_z_bending_moment` must be assigned a value. In other words, the release must have a spring stiffness in at least one linear direction or about one axis.

Notes:

New for CIS/2, but covered by RELEASE in CIS/1.

See diagram 24 of the EXPRESS-G diagrams in Appendix B.

Attribute definitions clarified for 2nd Edition.

5.3.44 release_spring_non_linear

Entity definition:

A type of release where the values of the spring stiffnesses are assumed to vary depending on the load condition. The non-linear spring release is defined as a series of linear spring releases with change values between consecutive conditions. In this way, a lower and upper bound are specified for the range in which the linear spring end fixity is valid.

Taken together, the series of linear spring releases define the degrees of freedom where the element meets a node; i.e. the end fixity of an element.

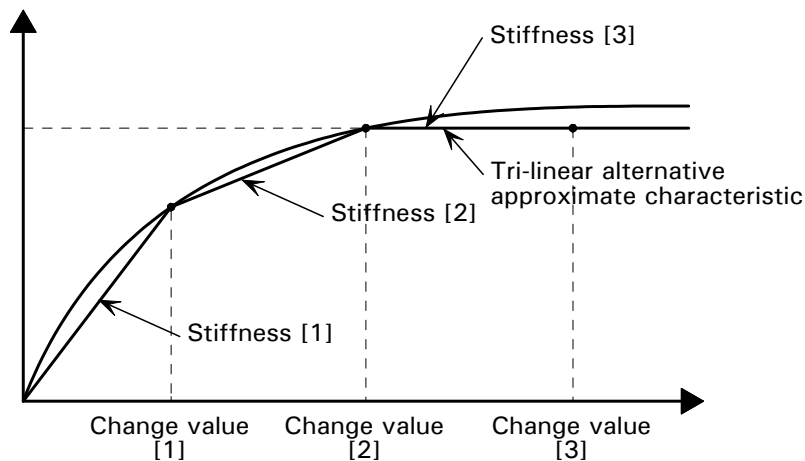


Figure 5.25 *Tri-linear approximation of a non-linear spring release*

EXPRESS specification:

```

*)
ENTITY release_spring_non_linear
SUBTYPE OF (release);
  change_values : LIST [2:?] OF measure_with_unit;
  values : LIST [2:?] OF release_spring_linear;
DERIVE
  number_of_values : INTEGER := SIZEOF(change_values);
WHERE
  WRR23 : SIZEOF(values) = SIZEOF(change_values);
END_ENTITY;
(*

```

Attribute definitions:

change_values

Declares the list of instances of `measure_with_unit` associated with the instance of `release_spring_non_linear`. There must be at least two instances of `measure_with_unit` associated with each instance of `release_spring_non_linear`. The order of the list is important as consecutive instances in the list represent the change values for consecutive conditions. Each instance in the list provides the numerical value with an appropriate unit of the upper bound for the range of validity of the previous instance of `release_spring_linear` and a lower bound for the range of validity of the next instance of `release_spring_linear`.

The lower bound for the first instance of `release_spring_linear` (referenced by the attribute `values`) is assumed to be zero. Thus, the first instance of `measure_with_unit` (referenced by the attribute `change_values`) provides the upper bound for the range of validity of the first instance of `release_spring_linear`. It also provides the lower bound for the range of validity of the second instance of `release_spring_linear`.

values

Declares the list of instances of `release_spring_linear` that are associated with (and are used to define) the `release_spring_non_linear`. There must be at least two instances of `release_spring_linear` associated with each instance of `release_spring_non_linear`. The order of the list is important as consecutive instances in the list represent consecutive conditions.

number_of_values

This attribute derives the integer value of the number of instances of `release_spring_linear` that are used to define the `release_spring_non_linear`.

Formal propositions:*DERIVE*

The derived attribute `number_of_values` does not appear in the STEP Part 21 file.

WRR23

The size of the aggregation associated with the attribute `change_values` shall be equal to the size of the aggregation associated with the attribute `values`. In other words, a non-linear spring must be defined as a sequence of at least two linear springs each provided with a change value.

Notes:

New for CIS/2.

See diagram 24 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

5.3.45 release_warping**Entity definition:**

A type of release that is defined as a linear spring release with the seventh degree of freedom populated. The value of this warping release (for rotation about a 'warping axis') is defined by reference to an instance of `rotational_stiffness_measure_with_unit`.

EXPRESS specification:

*)

ENTITY `release_warping`

SUBTYPE OF (`release_spring_linear`);

`release_warping_moment` : `rotational_stiffness_measure_with_unit`;

END_ENTITY;

(*)

Attribute definitions:*release_warping_moment*

Declares the instance of `rotational_stiffness_measure_with_unit` used to define the `release_warping`. This provides the numerical value with an appropriate unit of the release (or end fixity of the element) under warping, in the element's local coordinate system; (i.e. the restraint against warping).

Notes

New for CIS/2.

See Diagram 24 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6 LPM/6 LOADING

6.1 Loading concepts and assumptions

6.1.1 Conceptual overview

Loads may be applied to the nodes and elements. Elements may be subjected to concentrated, distributed or thermal loads. Loading is specified in a number of different ways using terminology adopted from Eurocode 1^[11]. The loading takes the form of basic load cases, which result from physical actions (e.g. permanent, variable, accidental, or seismic). Applied loads can be described as static or dynamic. Static loads are defined in terms of displacements, forces, or pressures, while dynamic loads are defined in terms of velocities or accelerations, associated with sets of static loads. The basic load cases are combined and factored to form loading combinations.

6.2 Loading type definitions

The following types have been modified for the 2nd Edition of CIS/2:

- projected_or_true_length

6.2.1 action_source_accidental

Type definition:

Classifies the type of accidental action (load) by its source; that is the action may be specified as being caused by either:

- fire,
- impulsive force (such as shock wave from an explosion),
- an impact of a solid object, or
- its source may be undefined.

EXPRESS specification:

*)

```
TYPE action_source_accidental
= ENUMERATION OF
    (fire, impulse, impact, undefined);
END_TYPE;
(*)
```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

6.2.2 action_source_permanent

Type definition:

Classifies the type of permanent action (load) by its source; that is the action may be specified as being caused by either:

- dead loads (e.g. walls, floors, and permanent finishes, fixtures and fittings),
- by the structure's self weight,

- a prestress applied to the structure,
- by lack of fit (arising from inaccuracies in the dimensions of the as-built structural members), or
- the source of the permanent action may be undefined.

EXPRESS specification:

*)

TYPE action_source_permanent

= ENUMERATION OF

(dead, self_weight, prestress, lack_of_fit, undefined);

END_TYPE;

(*

Notes:

New for CIS/2. Unchanged in 2nd Edition.

6.2.3 action_source_variable_long_term

Type definition:

Classifies the type of long term variable action (load) by its source; that is the action may be specified as being caused by:

- live loads (e.g. load imposed during the normal use of the structure by its occupants, furniture, movable items, etc),
- system imperfection (arising from changes in the structural members during the structure's life time),
- settlement (arising from ground movements during the structure's life time),
- temperature effect (arising from thermal changes during the structure's life time),
- or the source of the long term variable action may be undefined.

EXPRESS specification:

*)

TYPE action_source_variable_long_term

= ENUMERATION OF

(live, system_imperfection, settlement, temperature_effect, undefined);

END_TYPE;

(*

Notes:

New for CIS/2. Unchanged in 2nd Edition.

6.2.4 action_source_variable_short_term

Type definition:

Classifies the type of short term variable action (load) by its source; that is the action may be specified as being caused by:

- buoyancy (applicable to floating of submerged structures),
- wind,
- snow,

- ice (e.g. ice build up on cables or transmission towers),
- current (i.e. constantly moving water - applicable to floating of submerged structures),
- wave, (i.e. cyclical moving water applicable to floating of submerged structures),
- rain,
- or the source of the short term variable action may be undefined.

EXPRESS specification:

*)

TYPE action_source_variable_short_term

= ENUMERATION OF

(buoyancy, wind, snow, ice, current, wave, rain, undefined);

END_TYPE;

(*

Notes:

New for CIS/2. Unchanged in 2nd Edition.

6.2.5 action_source_variable_transient

Type definition:

Classifies the type of transient variable action (load) by its source; that is the action may be specified as being caused by:

- transport (from fabrication shop to site),
- erection (from initial to final position),
- propping (temporary support of structural members during construction), or
- the source of the transient variable action may be undefined.

EXPRESS specification:

*)

TYPE action_source_variable_transient

= ENUMERATION OF

(transport, erection, propping, undefined);

END_TYPE;

(*

Notes:

New for CIS/2.

6.2.6 direct_or_indirect_action

Type definition:

Classifies the type of action as either direct or indirect; i.e. the structure may be loaded directly or indirectly through the components it supports such as cladding. A direct action may be a force (load) applied to the structure. An indirect action may be an imposed or constrained deformation or an imposed acceleration caused by, for example, temperature changes, moisture variation, uneven settlement or earthquakes.

EXPRESS specification:

```
*)
TYPE direct_or_indirect_action
= ENUMERATION OF
    (direct_action, indirect_action);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

6.2.7 global_or_local_load

Type definition:

Classifies the load applied to an element as being defined in the global coordinate system, or the element's local coordinate system.

EXPRESS specification:

```
*)
TYPE global_or_local_load
= ENUMERATION OF
    (global_load, local_load);
END_TYPE;
(*
```

Notes:

Known as global_or_local in CIS/1.

6.2.8 projected_or_true_length

Type definition:

Classifies the distributed load applied to an element as being applied in a true direction (where the load length is equal to the actual loaded length of element) or projected onto an element (such that the loaded length is increased and the actual magnitude of the distributed load is decreased).

EXPRESS specification:

```
*)
TYPE projected_or_true_length
= ENUMERATION OF
    (projected_length, true_length);
END_TYPE;
(*
```

Notes:

Known as projected_or_true in CIS/1.

Modified for 2nd Edition – known as projected_or_true in CIS/2

6.2.9 spatial_variation

Type definition:

Classifies the physical action by its spatial variation; i.e. the ability of the action to move. Actions are classified as being either fixed (unable to move) or free to move. According to Eurocode 1, a fixed action is an action that has a fixed distribution over the structure such that the magnitude and direction of the action are determined unambiguously for the whole structure if this magnitude and direction are determined at one point on the structure. A free action is an action that may have any spatial distribution over the structure within given limits.

EXPRESS specification:

```
*)
TYPE spatial_variation
= ENUMERATION OF
    (free_action, fixed_action);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged in 2nd Edition.

6.2.10 static_or_dynamic

Type definition:

Classifies the type of physical action by its nature as either static, dynamic, or quasi-dynamic. According to Eurocode 1, a static action is an action that does not cause significant acceleration of the structure or structural members, while a dynamic action is one that does cause significant acceleration of the structure or structural members. A quasi-dynamic action is a dynamic action that can be described by static models in which the dynamic effects are included.

EXPRESS specification:

```
*)
TYPE static_or_dynamic
= ENUMERATION OF
    (static, dynamic, quasi_dynamic);
END_TYPE;
(*
```

Notes:

Known as ANALYSIS_TYPE in CIS/1, but now used in a different entity.

Unchanged in 2nd Edition.

6.3 Loading entity definitions

The following entities have been modified for the 2nd Edition of CIS/2:

- load
- load_case
- load_element_distributed
- loading_combination

The following entities have been added for the 2nd Edition of CIS/2:

- load_connection
- load_member
- load_member_concentrated
- load_member_distributed
- load_member_distributed_curve
- load_member_distributed_curve_line
- load_member_distributed_surface
- load_member_distributed_surface_uniform
- load_member_distributed_surface_varying

6.3.1 applied_load

Entity definition:

An abstract concept allowing the high level definition of values of the loads that are applied to an analysis model. The SUBTYPEs give the detailed values, while other entities reference this entity. Applied loads can be described as static or dynamic. Static loads are defined in terms of displacements, forces, or pressures, while dynamic loads are defined in terms of velocities or accelerations, associated with sets of static loads.

EXPRESS specification:

```
*)
ENTITY applied_load
ABSTRACT SUPERTYPE OF (ONEOF(
    applied_load_static,
    applied_load_dynamic));
    applied_load_name : label;
END_ENTITY;
(*
```

Attribute definitions:

applied_load_name

A short human-readable alphanumeric reference for the applied_load.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2.

See diagram 28 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.2 applied_load_dynamic**Entity definition:**

A type of applied_load where the loading is dependent on and variable with time. Dynamic loads are defined in terms of velocities or accelerations, associated with sets of static loads. Dynamic loads may be given initial and final values as well as maximum and minimum values of a static load. Dynamic loads may also be given a duration, frequency and a number of cycles.

EXPRESS specification:

*)

ENTITY applied_load_dynamic

SUPERTYPE OF (ONEOF

(applied_load_dynamic_acceleration,
applied_load_dynamic_velocity))

SUBTYPE OF (applied_load);

initial_value : OPTIONAL applied_load_static;

final_value : OPTIONAL applied_load_static;

maximum_value : OPTIONAL applied_load_static;

minimum_value : OPTIONAL applied_load_static;

number_of_cycles : OPTIONAL count_measure;

load_duration : OPTIONAL time_measure_with_unit;

load_frequency : OPTIONAL frequency_measure_with_unit;

WHERE

WRA10 : EXISTS (initial_value) OR

EXISTS (final_value) OR

EXISTS (maximum_value) OR

EXISTS (minimum_value) OR

EXISTS (number_of_cycles) OR

EXISTS (load_duration) OR

EXISTS (load_frequency) OR

('STRUCTURAL_FRAME_SCHEMA.

APPLIED_LOAD_DYNAMIC_ACCELERATION' IN TYPE OF (SELF)) OR

('STRUCTURAL_FRAME_SCHEMA.

APPLIED_LOAD_DYNAMIC_VELOCITY' IN TYPEOF (SELF));

END_ENTITY;

(*

Attribute definitions:*initial_value*

Declares the instance of `applied_load_static` that may be used to define the initial value of the `applied_load_dynamic`.

final_value

Declares the instance of `applied_load_static` that may be used to define the final value of the `applied_load_dynamic`.

maximum_value

Declares the instance of `applied_load_static` that may be used to define the maximum value of the `applied_load_dynamic`.

minimum_value

Declares the instance of `applied_load_static` that may be used to define the minimum value of the `applied_load_dynamic`.

number_of_cycles

An integer or real number value of the number of cycles of loading used to define the `applied_load_dynamic`. A cycle of loading may be considered to be an applied load moving from maximum to maximum (through minimum) or from minimum to minimum (through maximum). The count of how many times this cycle occurs gives the `number_of_cycles`.

load_duration

Declares the instance of `time_measure_with_unit` that that may be used to define the load duration of the `applied_load_dynamic`. The instance of `time_measure_with_unit` (from STEP Part 41) provides the real number value of the amount of time with an appropriate unit. In simple terms, this is how long the dynamic load is applied.

load_frequency

Declares the instance of `frequency_measure_with_unit` that that may be used to define the load frequency of the `applied_load_dynamic`. The instance of `frequency_measure_with_unit` provides the real number value of the frequency with an appropriate unit. In simple terms, this is how many times in a given time period the maximum value of the dynamic load is applied. This is related to the `number_of_cycles` and the `load_duration`.

Formal propositions:*WRA10*

One of the attributes `initial_value`, `final_value`, `maximum_value`, `minimum_value`, `number_of_cycles`, `load_duration`, or `load_frequency` must be given a value if this entity is populated on its own. In other words, the attributes of this entity can only all be assigned as NULL values in the SUBTYPEs of this entity. If this entity is populated on its own, one of its attributes must be populated.

Notes:

New for CIS/2.

See diagram 28 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.3 applied_load_dynamic_acceleration

Entity definition:

A type of applied_load_dynamic where the set of applied_load_static used to define the applied_load_dynamic is given a pre-set acceleration. This pre-set acceleration is given in 3 linear directions and about 3 rotational axes. (Any warping effects are combined with the torsion effects as a pre-set acceleration about the appropriate axis.)

A pre-set acceleration is one that is applied to an analysis model, rather than one that is the result of the analysis of the analysis model under consideration. This could be, for example, the excitation of a slender member under wind load, the simulation of seismic load, or the direct effect of an item of machinery supported by a member. The values of the pre-set acceleration may be derived from the results of analyses of other analysis models. For the purposes of the analysis model to which the pre-set acceleration is applied, the acceleration is from an external, rather than an internal, source.

EXPRESS specification:

*)

ENTITY applied_load_dynamic_acceleration

SUBTYPE OF (applied_load_dynamic);

 preset_acceleration_ax : OPTIONAL linear_acceleration_measure_with_unit;

 preset_acceleration_ay : OPTIONAL linear_acceleration_measure_with_unit;

 preset_acceleration_az : OPTIONAL linear_acceleration_measure_with_unit;

 preset_acceleration_arx : OPTIONAL rotational_acceleration_measure_with_unit;

 preset_acceleration_ary : OPTIONAL rotational_acceleration_measure_with_unit;

 preset_acceleration_arz : OPTIONAL rotational_acceleration_measure_with_unit;

WHERE

 WRA11 : EXISTS (preset_acceleration_ax) OR

 EXISTS (preset_acceleration_ay) OR

 EXISTS (preset_acceleration_az) OR

 EXISTS (preset_acceleration_arx) OR

 EXISTS (preset_acceleration_ary) OR

 EXISTS (preset_acceleration_arz);

END_ENTITY;

(*

Attribute definitions:

preset_acceleration_ax

Declares the first instance of linear_acceleration_measure_with_unit that may be used to define the applied_load_dynamic_acceleration. This provides the numerical value with an appropriate unit of the acceleration in linear x-direction.

preset_acceleration_ay

Declares the second instance of linear_acceleration_measure_with_unit that may be used to define the applied_load_dynamic_acceleration. This provides the numerical value with an appropriate unit of the acceleration in linear y-direction.

preset_acceleration_az

Declares the third instance of linear_acceleration_measure_with_unit that may be used to define the applied_load_dynamic_acceleration. This provides the numerical value with an appropriate unit of the acceleration in linear z-direction.

preset_acceleration_arx

Declares the first instance of `rotational_acceleration_measure_with_unit` that may be used to define the `applied_load_dynamic_acceleration`. This provides the numerical value with an appropriate unit of the angular acceleration about the x-axis.

preset_acceleration_ary

Declares the second instance of `rotational_acceleration_measure_with_unit` that may be used to define the `applied_load_dynamic_acceleration`. This provides the numerical value with an appropriate unit of the angular acceleration about the y-axis.

preset_acceleration_arz

Declares the third instance of `rotational_acceleration_measure_with_unit` that may be used to define the `applied_load_dynamic_acceleration`. This provides the numerical value with an appropriate unit of the angular acceleration about the z-axis.

Formal propositions:*WRA 11*

At least one of the attributes `preset_acceleration_ax`, `preset_acceleration_ay`, `preset_acceleration_az`, `preset_acceleration_arx`, `preset_acceleration_ary`, or `preset_acceleration_arz` must be assigned a value. In other words, an `applied_load_dynamic_acceleration` cannot be instanced with all of its attributes set to NULL values.

Notes:

New for CIS/2.

See diagram 29 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.4 applied_load_dynamic_velocity**Entity definition:**

A type of `applied_load_dynamic` where the set of `applied_load_static` used to define the `applied_load_dynamic` is given a pre-set velocity. This pre-set velocity is given in 3 linear directions and about 3 rotational axes. (Any warping effects are combined with the torsion effects as a pre-set velocity about the appropriate axis.)

A pre-set velocity is one that is applied to an analysis model, rather than one that is the result of the analysis of the analysis model under consideration. This could be, for example, the excitation of a slender member under wind load, the simulation of seismic load, or the direct effect of an item of machinery supported by a member. The values of the pre-set velocity may be derived from the results of analyses of other analysis models. For the purposes of the analysis model to which the pre-set velocity is applied, the velocity is from an external, rather than an internal, source.

EXPRESS specification:

*)

ENTITY `applied_load_dynamic_velocity`

SUBTYPE OF (`applied_load_dynamic`);

`preset_velocity_vx` : OPTIONAL `linear_velocity_measure_with_unit`;

`preset_velocity_vy` : OPTIONAL `linear_velocity_measure_with_unit`;

`preset_velocity_vz` : OPTIONAL `linear_velocity_measure_with_unit`;

```

    preset_velocity_vrx : OPTIONAL rotational_velocity_measure_with_unit;
    preset_velocity_vry : OPTIONAL rotational_velocity_measure_with_unit;
    preset_velocity_vrz : OPTIONAL rotational_velocity_measure_with_unit;
WHERE
    WRA12 : EXISTS (preset_velocity_vx) OR
            EXISTS (preset_velocity_vy) OR
            EXISTS (preset_velocity_vz) OR
            EXISTS (preset_velocity_vrx) OR
            EXISTS (preset_velocity_vry) OR
            EXISTS (preset_velocity_vrz);
END_ENTITY;
(*)

```

Attribute definitions:

preset_velocity_vx

Declares the first instance of `linear_velocity_measure_with_unit` that may be used to define the `applied_load_dynamic_velocity`. This provides the numerical value with an appropriate unit of the velocity in linear x-direction.

preset_velocity_vy

Declares the second instance of `linear_velocity_measure_with_unit` that may be used to define the `applied_load_dynamic_velocity`. This provides the numerical value with an appropriate unit of the velocity in linear y-direction.

preset_velocity_vz

Declares the third instance of `linear_velocity_measure_with_unit` that may be used to define the `applied_load_dynamic_velocity`. This provides the numerical value with an appropriate unit of the velocity in linear z-direction.

preset_velocity_vrx

Declares the first instance of `rotational_velocity_measure_with_unit` that may be used to define the `applied_load_dynamic_velocity`. This provides the numerical value with an appropriate unit of the angular velocity about the x-axis.

preset_velocity_vry

Declares the second instance of `rotational_velocity_measure_with_unit` that may be used to define the `applied_load_dynamic_velocity`. This provides the numerical value with an appropriate unit of the angular velocity about the y-axis.

preset_velocity_vrz

Declares the third instance of `rotational_velocity_measure_with_unit` that may be used to define the `applied_load_dynamic_velocity`. This provides the numerical value with an appropriate unit of the angular velocity about the z-axis.

Formal propositions:

WRA12

At least one of the attributes `preset_velocity_vx`, `preset_velocity_vy`, `preset_velocity_vz`, `preset_velocity_vrx`, `preset_velocity_vry`, or `preset_velocity_vrz` must be assigned a value. In other words, an `applied_load_dynamic_velocity` cannot be instanced with all of its attributes set to NULL values.

Notes:

New for CIS/2.

See diagram 29 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.5 applied_load_static**Entity definition:**

A type of applied_load where the loading is not dependent on or variable with time. Static loads are defined in terms of displacements, forces, or pressures. SUBTYPEs of this entity provide the numerical values to define a static load.

EXPRESS specification:

```
*)
ENTITY applied_load_static
ABSTRACT SUPERTYPE OF (ONEOF
    (applied_load_static_displacement,
    applied_load_static_force,
    applied_load_static_pressure))
SUBTYPE OF (applied_load);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own, however, it does inherit several from its SUPERTYPE. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Notes:

New for CIS/2.

See diagram 28 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.6 applied_load_static_displacement**Entity definition:**

A type of applied_load_static where the loading is defined in terms of pre-set displacements along 3 directions and about 3 axes. (Any warping effects are combined with the torsion effects as a pre-set displacement about the appropriate axis.)

A pre-set displacement that is one that is applied to an analysis model, rather than one that is the result of the analysis of the analysis model under consideration. This could be, for example, the simulation of differential settlement of a support, or the distortion of members due to imperfections or manufacturing tolerances. The values of the pre-set displacement may be derived from the results of analyses of other analysis models. For

the purposes of the analysis model to which the pre-set displacement is applied, the velocity is from an external, rather than an internal, source.

Where the pre-set displacement is applied to a two-dimensional analysis model, the attributes representing the out of plane displacements shall be assigned NULL values (rather than zero values).

Where the load is applied to a three-dimensional analysis model, all of the attributes should be assigned real values (even if the value of the displacement is zero).

EXPRESS specification:

*)

ENTITY applied_load_static_displacement

SUBTYPE OF (applied_load_static);

 preset_displacement_dx : OPTIONAL length_measure_with_unit;

 preset_displacement_dy : OPTIONAL length_measure_with_unit;

 preset_displacement_dz : OPTIONAL length_measure_with_unit;

 preset_displacement_rx : OPTIONAL plane_angle_measure_with_unit;

 preset_displacement_ry : OPTIONAL plane_angle_measure_with_unit;

 preset_displacement_rz : OPTIONAL plane_angle_measure_with_unit;

WHERE

 WRA13 : EXISTS (preset_displacement_dx) OR

 EXISTS (preset_displacement_dy) OR

 EXISTS (preset_displacement_dz) OR

 EXISTS (preset_displacement_rx) OR

 EXISTS (preset_displacement_ry) OR

 EXISTS (preset_displacement_rz);

END_ENTITY;

(*)

Attribute definitions:

preset_displacement_dx

Declares the first instance of length_measure_with_unit that may be used to define the applied_load_static_displacement. This entity (taken from STEP Part 41) provides the numerical value with an appropriate unit of the pre-set linear displacement in the x-direction.

preset_displacement_dy

Declares the second instance of length_measure_with_unit that may be used to define the applied_load_static_displacement. This entity (taken from STEP Part 41) provides the numerical value with an appropriate unit of the pre-set linear displacement in the y-direction.

preset_displacement_dz

Declares the third instance of length_measure_with_unit that may be used to define the applied_load_static_displacement. This entity (taken from STEP Part 41) provides the numerical value with an appropriate unit of the pre-set linear displacement in the z-direction.

preset_displacement_rx

Declares the first instance of plane_angle_measure_with_unit that may be used to define the applied_load_static_displacement. This entity (taken from STEP Part 41) provides the numerical value with an appropriate unit of the pre-set rotation about the x-axis.

preset_displacement_ry

Declares the second instance of *plane_angle_measure_with_unit* that may be used to define the *applied_load_static_displacement*. This entity (taken from STEP Part 41) provides the numerical value with an appropriate unit of the pre-set rotation about the y-axis.

preset_displacement_rz

Declares the third instance of *plane_angle_measure_with_unit* that may be used to define the *applied_load_static_displacement*. This entity (taken from STEP Part 41) provides the numerical value with an appropriate unit of the pre-set rotation about the z-axis.

Formal propositions:**WRA13**

At least one of the attributes *preset_displacement_dx*, *preset_displacement_dy*, *preset_displacement_dz*, *preset_displacement_rx*, *preset_displacement_ry*, or *preset_displacement_rz* must be assigned a value. In other words, an *applied_load_static_displacement* cannot be instanced with all of its attributes set to NULL values.

Notes:

New for CIS/2.

See diagram 28 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.7 applied_load_static_force**Entity definition:**

A type of *applied_load_static* where the loading is defined in terms of forces along 3 directions and moments about 3 axes.

Where the load is applied to a two-dimensional analysis model, the attributes representing the out of plane forces or moments shall be assigned NULL values (rather than zero values).

Where the load is applied to a three-dimensional analysis model, all of the attributes should be assigned real values (even if the value of the force is zero).

EXPRESS specification:

*)

ENTITY *applied_load_static_force*

SUBTYPE OF (*applied_load_static*);

applied_force_fx : OPTIONAL *force_measure_with_unit*;
applied_force_fy : OPTIONAL *force_measure_with_unit*;
applied_force_fz : OPTIONAL *force_measure_with_unit*;
applied_moment_mx : OPTIONAL *moment_measure_with_unit*;
applied_moment_my : OPTIONAL *moment_measure_with_unit*;
applied_moment_mz : OPTIONAL *moment_measure_with_unit*;

WHERE

WRA14 : EXISTS (*applied_force_fx*) OR
 EXISTS (*applied_force_fy*) OR
 EXISTS (*applied_force_fz*) OR
 EXISTS (*applied_moment_mx*) OR

```

    EXISTS (applied_moment_my) OR
    EXISTS (applied_moment_mz);
END_ENTITY;
(*)

```

Attribute definitions:

applied_force_fx

Declares the first instance of `force_measure_with_unit` that may be used to define the `applied_load_static_force`. This entity provides the numerical value with an appropriate unit of the applied force in the x-direction.

applied_force_fy

Declares the second instance of `force_measure_with_unit` that may be used to define the `applied_load_static_force`. This entity provides the numerical value with an appropriate unit of the applied force in the y-direction.

applied_force_fz

Declares the third instance of `force_measure_with_unit` that may be used to define the `applied_load_static_force`. This entity provides the numerical value with an appropriate unit of the applied force in the z-direction.

applied_moment_mx

Declares the first instance of `moment_measure_with_unit` that may be used to define the `applied_load_static_force`. This entity provides the numerical value with an appropriate unit of the applied moment about the x-axis.

applied_moment_my

Declares the second instance of `moment_measure_with_unit` that may be used to define the `applied_load_static_force`. This entity provides the numerical value with an appropriate unit of the applied moment about the y-axis.

applied_moment_mz

Declares the third instance of `moment_measure_with_unit` that may be used to define the `applied_load_static_force`. This entity provides the numerical value with an appropriate unit of the applied moment about the z-axis.

Formal propositions:

WRA14

At least one of the attributes `applied_force_fx`, `applied_force_fy`, `applied_force_fz`, `applied_moment_mx`, `applied_moment_my`, or `applied_moment_mz` must be assigned a value. In other words, an `applied_load_static_force` cannot be instanced with all of its attributes set to NULL values.

Notes:

New for CIS/2.

See diagram 28 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.8 applied_load_static_pressure

Entity definition:

A type of applied_load_static where forces are applied over given areas.

EXPRESS specification:

```
*)
ENTITY applied_load_static_pressure
SUBTYPE OF (applied_load_static);
    applied_pressure_px : OPTIONAL pressure_measure_with_unit;
    applied_pressure_py : OPTIONAL pressure_measure_with_unit;
    applied_pressure_pz : OPTIONAL pressure_measure_with_unit;
WHERE
    WRA15 : EXISTS (applied_pressure_px) OR
            EXISTS (applied_pressure_py) OR
            EXISTS (applied_pressure_pz);
END_ENTITY;
(*
```

Attribute definitions:

applied_pressure_px

Declares the first instance of pressure_measure_with_unit that may be used to define the applied_load_static_pressure. This entity provides the numerical value with an appropriate unit of the pressure applied in the x-direction and projected onto the yz-plane.

applied_pressure_py

Declares the second instance of pressure_measure_with_unit that may be used to define the applied_load_static_pressure. This entity provides the numerical value with an appropriate unit of the pressure applied in the y-direction and projected onto the xz-plane.

applied_pressure_pz

Declares the third instance of pressure_measure_with_unit that may be used to define the applied_load_static_pressure. This entity provides the numerical value with an appropriate unit of the pressure applied in the z-direction and projected onto the xy-plane.

Formal propositions:

WRA15

At least one of the attributes applied_pressure_px, applied_pressure_py, or applied_pressure_pz must be assigned a value. In other words, an applied_load_static_pressure cannot be instanced with all of its attributes set to NULL values.

Informal propositions:

A pressure load can only be applied to a two-dimensional or a three-dimensional element. Thus, this entity should not be associated with instances of element_point or element_curve.

Notes:

New for CIS/2.

See diagram 28 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.9 load

Entity definition:

An abstract concept allowing the high level definition of values of the loads that are applied to an analysis model. The SUBTYPEs give the detailed values, while other entities reference this entity. Loads may be applied to nodes or elements. A load is assigned to a load_case and is given a name. A load may also be a description.

EXPRESS specification:

*)

ENTITY load

ABSTRACT SUPERTYPE OF (ONEOF(
load_element, load_node, load_member, load_connection));

parent_load_case : load_case;

load_name : label;

load_description : OPTIONAL text;

END_ENTITY;

(*

Attribute definitions:

parent_load_case

Declares the instance of load_case associated with the load. This load_case is defined as being the parent of the load.

load_name

A short human-readable alphanumeric reference for the load.

load_description

A free text description of the load.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Notes:

New for CIS/2.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – subtypes added.

6.3.10 load_case

Entity definition:

A combination of individual loads optionally multiplied by a common load factor, which are used in the definition of loading_combinations. This may be a set of analytical loads applied to elements and nodes (via the entities load_element and load_node), or a set of design loads applied to members and connections (via the entities load_member and load_connection). The load_case must be given a name and must be classified by the

type of physical action under consideration. The load_case may also be assigned to a set of analysis methods.

The 2nd Edition of CIS/2 includes both design and analytical load models. Where Rules in this entity prevent analytical loads and design loads being mixed in the same load case.

EXPRESS specification:

```

*)
ENTITY load_case
SUPERTYPE OF (load_case_documented);
    load_case_name : label;
    load_case_factor : OPTIONAL REAL;
    governing_analyses : SET [0:?] OF analysis_method;
    time_variation : physical_action;
DERIVE
    load_components : SET [1:?] OF load := bag_to_set
        (USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.
        LOAD.PARENT_LOAD_CASE'));
INVERSE
    loads : SET [1:?] OF load FOR parent_load_case;
WHERE
    WRL60 : NOT ((SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_ELEMENT') IN
        TYPE OF(components))) > 0 )
        AND (SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_MEMBER') IN
        TYPE OF(components))) > 0 ));
    WRL61 : NOT ((SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_NODE') IN
        TYPE OF(components))) > 0 )
        AND (SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_CONNECTION') IN
        TYPE OF(components))) > 0 ));
    WRL62 : NOT ((SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_NODE') IN
        TYPE OF(components))) > 0 )
        AND (SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_MEMBER') IN
        TYPE OF(components))) > 0 ));
    WRL63 : NOT ((SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_ELEMENT') IN
        TYPE OF(components))) > 0 )
        AND (SIZEOF(QUERY(components <* load_components |
        ('STRUCTURAL_FRAME_SCHEMA.LOAD_CONNECTION') IN
        TYPE OF(components))) > 0 ));
END_ENTITY;
(

```

Attribute definitions:

load_case_name

A short alphanumeric reference for the load_case.

load_case_factor

A numerical multiplier (or factor) that is globally applied to the values of the loads contained within the load_case used during the analysis.

governing_analyses

Declares the set of instances of analysis_method that may be associated with (and govern) the load_case. The load_case may be governed by a set of zero, one or more analysis_methods.

This attribute is defined as a SET whose lower bound (minimum number of items in the aggregation) is zero. In other words, an empty set is allowed. Semantically, this allows the application to leave the aggregation empty. Nevertheless, since the attribute is not optional, null attributes are not allowed; as a minimum, the attribute should be populated as an empty set.

time_variation

Declares the instance of physical_action associated with (and classifying) the load_case. The SUBTYPES of the entity physical_action classify the load_case by the dependency on time of the physical actions. These physical actions can be permanent, variable, accidental, or seismic.

DERIVE

load_components

Derives the set of one or more instances of load that use this load_case as their parent load case.

Formal propositions:

loads

The load_case shall be the parent of set of one of more loads. The load_case cannot exist without being referenced by a load.

WRL60

The set of instances of load shall not include those of type load_element and load_member.

WRL61

The set of instances of load shall not include those of type load_node and load_connection.

WRL62

The set of instances of load shall not include those of type load_node and load_member.

WRL63

The set of instances of load shall not include those of type load_element and load_connection.

Notes:

Known as BASIC_LOAD_CASE in CIS/1.

See diagram 30 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – Derived attribute and Where Rules added.

All of the Where Rules simply prevent analytical loads and design loads being mixed in the same load case.

6.3.11 load_case_documented

Entity definition:

A type of load_case that is assigned a document reference, through the STEP Part 41 entity document_usage_constraint. The instance of document_usage_constraint represents an extract from a specified document; for example, a clause in a national standard or a chapter of a design guide. It also includes the free text description of the relevant part of the document, which in this case, is deemed to govern the definition of the load case, the combination of loads, and any factors applied to the load.

EXPRESS specification:

```
*)
ENTITY load_case_documented
SUBTYPE OF (load_case);
    code_ref : document_usage_constraint;
END_ENTITY;
(*
```

Attribute definitions:

code_ref

Declares the instance of document_usage_constraint associated with the load_case_documented. This provides the extract from the appropriate national standard or code of practice that governs the definition of the load case.

Notes

New for CIS/2.

See Diagram 30 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.12 load_combination_occurrence

Entity definition:

One of the terms of a loading combination. A multiplier (or factor) determines the extent to which the load_case participates in the loading_combination. This is effectively an associative entity that resolves the many-to-many relationship between load_case and loading_combination.

EXPRESS specification:

```
*)
ENTITY load_combination_occurrence;
    load_combination_factor : OPTIONAL REAL;
    parent_load_combination : loading_combination;
    component_load_case : load_case;
UNIQUE
    URL1 : parent_load_combination, component_load_case;
```


END_ENTITY;

(*

Attribute definitions:

load_combination_factor

The value of the factor (or multiplier) which may be globally applied to a combination of load_cases to produce a load_combination_occurrence. (Corresponds to the partial factor of safety for load cases - see Clause 2.3.3. in EC3^[12]).

parent_load_combination

Declares the instance of the loading_combination, to which the load_combination_occurrence belongs. The load_combination_occurrence shall be used with one (and only one) loading combination.

component_load_case

Declares the instance of the load_case associated with the load_combination_occurrence. The load_combination_occurrence shall be used with one (and only one) load_case.

Formal propositions:

URL 1

The combination of the parent_load_combination, component_load_case (i.e. the instances of loading_combination and load_case) associated with the load_combination_occurrence shall be unique to this instance.

Informal propositions:

Because an INVERSE clause is not specified in the load_case, the default cardinality applies such that any number of instances of load_combination_occurrence can be associated with a load_case. In other words, a load_case may be associated with zero, one or many instances of load_combination_occurrence. However, a loading combination must be associated at least one instance of load_combination_occurrence.

Notes:

Known as LOAD_COMBINATION_OCCURRENCE in CIS/1.

See diagram 30 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.13 load_connection

Entity definition:

A type of load where the load is applied to a specific structural connection. It is effectively an associative entity resolving a many-to-many relationship between assembly_design_structural_connection and applied_load. As a type of load, it inherits a parent load case, as well as a number, label and description for the load.

EXPRESS specification:

*)

ENTITY load_connection

SUBTYPE OF (load);

supporting_connection : assembly_design_structural_connection;

load_values : applied_load;

WHERE

```

WRL59 : NOT
('STRUCTURAL_FRAME_SCHEMA.APPLIED_LOAD_STATIC_PRESSURE' IN TYPE
OF(load_values));
END_ENTITY;
(*)

```

Attribute definitions:

supporting_connection

Declares the instance of `assembly_design_structural_connection` associated with (and therefore supporting) the `load_connection`.

load_values

Declares the instance of `applied_load` associated with (and therefore defining the values of) the `load_connection`. As `applied_load` is abstract, the SUBTYPEs are used. For example, when the `applied_load` referred to is the SUBTYPE `applied_load_static_force`, the `load_connection` is defined in terms of 3 linear forces (F_x , F_y , F_z) and 3 moments (M_x , M_y , M_z).

Formal propositions:

WRL59

The `applied_load` referenced by the attribute `load_values` shall not be of the type `applied_load_static_pressure`.

Informal propositions:

A `load_node` cannot exist without corresponding instances of `applied_load` and `assembly_design_structural_connection`. Because INVERSE clauses are not specified in the `assembly_design_structural_connection` and the `applied_load`, the default cardinality applies such that any number of instances of `load_connection` can be associated with these items. In other words, a `load_case` may be associated with zero, one or many instances of `load_connection`, and an `assembly_design_structural_connection` may be associated with zero, one or many instances of `load_connection`.

Notes:

New for CIS/2 2nd Edition.

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

6.3.14 load_element

Entity definition:

A type of load applied to a particular element. The `load_element` may be described in terms of a concentrated, distributed or thermal load. The load can be applied in either local or global coordinate systems. It can also be classified as a destabilizing, for the purposes of member design. As a type of load, it inherits a parent load case, as well as a name and a description for the load.

EXPRESS specification:

```

*)
ENTITY load_element
ABSTRACT SUPERTYPE OF (ONEOF
    (load_element_distributed,
    load_element_thermal,
    load_element_concentrated))

```

```

SUBTYPE OF (load);
  supporting_element : element;
  load_position_label : OPTIONAL label;
  reference_system : OPTIONAL text;
  destabilizing_load : OPTIONAL BOOLEAN;
  global_or_local : global_or_local_load;
END_ENTITY;
(*)

```

Attribute definitions:

supporting_element

Declares the instance of element associated with the load. While the element may be associated with (and thus support) zero, one or many instances of load_element, the load_element must be associated with (and located on) one (and only one) element.

load_position_label

A short text description of the position on the element at which the load is applied, e.g. mid-span. For distributed loads the position corresponds to the point where the initial value of the load is specified.

reference_system

A text description of the referencing system used to describe the loads on the element.

destablizing_load

Declares the classification of whether the load is destabilizing (TRUE) or not (FALSE), for the purposes of member design.

global_or_local

Declares the classification of the direction of the loading applied to this element. The loading direction can be specified as either being in the global coordinate system (X, Y, Z), or the in the element's local coordinate system (x, y, z).

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

Known as ELEMENT_LOAD in CIS/1.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.15 load_element_concentrated

Entity definition:

A type of load_element where the load is described as being applied at a point within the element. The value of the load is defined by a reference to a single instance of applied_load. For two-dimensional analysis models, some of the attributes representing the (out-of-plane) magnitude of the applied load may be assigned NULL values. For three-dimensional analysis models, all of the attributes representing the magnitude of the applied load should be assigned values, even if the value is zero.

If the load is not applied in the direction of one of the 3 (local or global) axes, then the values are the resolution of the load. If the load is applied in the direction of one of the 3 axes, then the remaining values shall be set to 0.0 (or NULL for two-dimensional analysis models).

The applied load can be described as static or dynamic. Static loads are defined in terms of displacements, forces, or pressures, while dynamic loads are defined in terms of velocities or accelerations, associated with sets of static loads.

EXPRESS specification:

*)

ENTITY load_element_concentrated

SUBTYPE OF (load_element);

load_position : point;

load_value : applied_load;

WHERE

WRL40 : NOT ('STRUCTURAL_FRAME_SCHEMA.

APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(load_value));

END_ENTITY;

(*

Attribute definitions:

load_position

Declares the instance of point used to define the position of the concentrated load on the element in question. The point entity (from STEP Part 42) specifies the numerical value (in accordance with the unit system) of the position within the element, measured from the start of the element adjacent to the first connecting node. For one-dimensional elements, the value corresponding to the point's x-coordinate specifies the linear distance to the applied load measured along the element's x-axis from the element's start end.

load_value

Declares the instance of applied_load used to define the values of the load_element_concentrated. As applied_load is abstract, the SUBTYPEs are used. For example, when the applied load referred to is the SUBTYPE applied_load_static_force, a concentrated element load is defined in terms of three linear forces (Fx, Fy, Fz) and three moments (Mx, My, Mz). It should be noted that a load_element_concentrated cannot exist without a corresponding instance of one of the SUBTYPEs of applied_load.

Formal propositions:

WRL40

The applied_load used to define the values of the load_element_concentrated shall not be of the type applied_load_static_pressure.

Notes:

Known as CONC_1D_ELMNT_LOAD in CIS/1.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.16 load_element_distributed

Entity definition:

A type of load_element where the load is considered to be distributed. This is a generic entity that is intended to cover all distributed loading applied to all elements with a dimensionality greater than zero; including uniformly distributed, triangular, and the general patch load.

EXPRESS specification:

```

*)
ENTITY load_element_distributed
ABSTRACT SUPERTYPE OF (ONEOF
    (load_element_distributed_surface,
    load_element_distributed_curve))
SUBTYPE OF (load_element);
    projected_or_true : projected_or_true_length;
WHERE
    WRL18 : NOT ('STRUCTURAL_FRAME_SCHEMA.ELEMENT_POINT' IN
        TYPE OF(SELFlload_element.supporting_element));
END_ENTITY;
(*

```

Attribute definitions:

projected_or_true

Declares the classification of the distributed load applied to an element as being applied in a true direction (where the load length is equal to the actual loaded length of element) or projected onto an element (such that the loaded length is increased and the actual magnitude of the distributed load is decreased).

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

WRL18

The type of element associated with this load shall not be an element_point. That is, the element shall be one, two or three-dimensional but not zero-dimensional.

Notes:

Known as DISTRIBUTED_1D_ELMNT_LOAD in CIS/1.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – type projected_or_true_length renamed.

6.3.17 load_element_distributed_curve

Entity definition:

A type of load_element_distributed where the loading is defined as being applied along a curve. The curve is defined using explicit geometry taken from STEP Part 42. This can apply to one, two or three dimensional elements but not zero-dimensional elements.

EXPRESS specification:

```

*)
ENTITY load_element_distributed_curve
SUPERTYPE OF (load_element_distributed_curve_line)
SUBTYPE OF (load_element_distributed);
    start_load_value : applied_load;
    end_load_value : applied_load;
    curve_definition : curve;
WHERE
    WRL41 : NOT ('STRUCTURAL_FRAME_SCHEMA.
        APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(start_load_value));
    WRL42 : NOT ('STRUCTURAL_FRAME_SCHEMA.
        APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(end_load_value));
    WRL44 : TYPE OF(start_load_value) = TYPEOF(end_load_value);
END_ENTITY;
(*)

```

Attribute definitions:*start_load_value*

Declares the first instance of *applied_load* used to define the values at the beginning of the curve defining the *load_element_distributed_curve*. As *applied_load* is abstract, the SUBTYPES are used. For example, when the *applied_load* referred to is the SUBTYPE *applied_load_static_force*, the start value of the distributed element load is defined in terms of 3 linear forces (Fx, Fy, Fz) and 3 moments (Mx, My, Mz). The instance of *applied_load* defines the value for a unit length of the associated curve. The unit length is measured in the same units as those used to define the curve. Therefore the distributed load may need to be modified to suit the definition of the curve. For example, if the curve has been defined in millimetres then a distributed load of 60kN/m must be expressed as a load of 60N. Similarly, if the curve has been defined in inches, a distributed load of 60lbf/ft must be expressed as a load of 5lbf.

end_load_value

Declares the second instance of *applied_load* used to define the values at the end of the curve defining the *load_element_distributed_curve*. (See also definition for *start_load_value*.)

curve_definition

Declares the instance of *curve* used to define the *load_element_distributed_curve*. The curve entity (taken from STEP Part 42) specifies the line along which the load is applied. The curve may (but need not be) a straight line. The curve may be bounded, trimmed, or segmented. The curve is defined in the local coordinate system of the element.

Formal propositions:*WRL41*

The *applied_load* referenced by the attribute *start_load_value* shall not be of the type *applied_load_static_pressure*.

WRL42

The *applied_load* referenced by the attribute *end_load_value* shall not be of the type *applied_load_static_pressure*.

WRL44

The `applied_load` referenced by the attribute `start_load_value` shall be of the same type as that referenced by the attribute `end_load_value`.

Informal propositions:

It should be noted that a `load_element_distributed_curve` cannot exist without corresponding instances of one of the SUBTYPES of `applied_load`. Normally there would be two separate instances, but two references to the same instance may be used when the start and end values are the same.

It is assumed that the curve is defined in the same geometric context as the element; i.e. the same coordinate system and dimensionality. If the curve does not lie on the element, then the load is projected onto it from the curve. It is assumed that this projection meets the element – either in whole or in part.

Notes:

New for CIS/2.

The entity curve is defined in STEP Part 42.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

6.3.18 load_element_distributed_curve_line**Entity definition:**

A type of `load_element_distributed_curve` where the curve is defined by a (straight) line. The line (taken from STEP Part 42) is a SUBTYPE of curve.

EXPRESS specification:

*)

ENTITY `load_element_distributed_curve_line`

SUBTYPE OF (`load_element_distributed_curve`);

WHERE

WRL19 : 'STRUCTURAL_FRAME_SCHEMA.LINE' IN TYPE OF
(SELF\load_element_distributed_curve.curve_definition);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `load_element_distributed_curve`.

Formal propositions:**WRL19**

The curve used to define the `load_element_distributed_curve_line` shall be of the type 'line'.

Informal propositions:

The line is defined in the local coordinate system of the element. Thus, the `cartesian_point` referenced by the line shall lie on the element and be defined relative to the element's coordinate system.

The instances of applied load (inherited from the SUPERTYPE `load_element_distributed_curve`) define the values for unit lengths of the associated line. The lengths are measured in the same units as those used to define the line. Therefore, the distributed load may need to be modified to suit the definition of the line. For example, if the line has been defined in millimetres then a distributed load of 60kN/m must be expressed as a load of 60N. Similarly, if the line has been defined in inches, a distributed load of 60lbf/ft must be expressed as a load of 5lbf.

Notes:

New for CIS/2, but partially addressed by `DISTRIBUTED_1D_ELMNT_LOAD` in CIS/1.

The entity line is defined in STEP Part 42.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

6.3.19 `load_element_distributed_surface`

Entity definition:

A type of `load_element_distributed` where the loading is defined over a surface. The surface is defined by reference to the entity `bounded_surface` (taken from STEP Part 42). This may be applied to two or three dimensional elements, but not to zero or one-dimensional elements. The values of the loading are defined by the SUBTYPES.

EXPRESS specification:

```
*)
ENTITY load_element_distributed_surface
ABSTRACT SUPERTYPE OF (ONEOF
    (load_element_distributed_surface_uniform,
     load_element_distributed_surface_varying))
SUBTYPE OF (load_element_distributed);
    surface_definition : bounded_surface;
WHERE
WRL20 : NOT ('STRUCTURAL_FRAME_SCHEMA.ELEMENT_CURVE' IN TYPE OF
    (SELF\ load_element.supporting_element));
END_ENTITY;
(*
```

Attribute definitions:

surface_definition

Declares the instance of `bounded_surface` used to define the boundary of the loading for the `load_element_distributed_surface`.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

WRL20

The supporting element under consideration shall not be of the type 'element_curve'. That is, the element shall be two or three-dimensional but not zero or one-dimensional.

Informal propositions:

It is assumed that the `bounded_surface` is defined in the same geometric context as the element. If the `bounded_surface` does not lie on the surface of the element, then the load is projected onto the element's surface from the `bounded_surface`. It is assumed that this projection meets the surface of the element – either in whole or in part.

Notes:

New for CIS/2.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

6.3.20 load_element_distributed_surface_uniform**Entity definition:**

A type of `load_element_distributed_surface` where the load value is defined by a single reference to an instance of `applied_load_static_pressure` and is assumed to be constant across the `bounded_surface`.

EXPRESS specification:

```
*)
ENTITY load_element_distributed_surface_uniform
  SUBTYPE OF (load_element_distributed_surface);
    load_value : applied_load_static_pressure;
END_ENTITY;
(*
```

Attribute definitions:*load_value*

Declares the instance of `applied_load_static_pressure` used to define the loading values of the `load_element_distributed_surface_uniform`.

Informal propositions:

The load value is assumed to be constant across the `bounded_surface` of the `load_element_distributed_surface_uniform`.

Notes:

New for CIS/2.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

6.3.21 load_element_distributed_surface_varying**Entity definition:**

A type of `load_element_distributed_surface` where the load value is defined by multiple references to instances of `applied_load_static_pressure` and is assumed to vary across the `bounded_surface`.

EXPRESS specification:

```
*)
ENTITY load_element_distributed_surface_varying
  SUBTYPE OF (load_element_distributed_surface);
```

```

    load_values : SET [4:4] OF applied_load_static_pressure;
WHERE
WRL45 : 'STRUCTURAL_FRAME_SCHEMA.
    RECTANGULAR_TRIMMED_SURFACE' IN TYPE OF
    (SELF\load_element_distributed_surface.surface_definition);
END_ENTITY;
(*)

```

Attribute definitions:

load_values

Declares the set of four instances of `applied_load_static_pressure` used to define the loading values of the `load_element_distributed_surface_varying`. Each of these is placed at a position on the element corresponding to the vertices of the `rectangular_trimmed_surface`.

The values of the parameters of the `applied_load_static_pressure` apply to a unit area of the `bounded_surface` on the element and are measured in the same units that define the `bounded_surface`. The values of the applied distributed load may need to be converted to suit the `bounded_surface`. A linear distribution of pressure is assumed to occur between each vertex of the `bounded_surface`.

Formal propositions:

WRL45

The `bounded_surface` referenced by the attribute `surface_definition` (inherited from the SUPERTYPE `load_element_distributed_surface`) shall be of the type `rectangular_trimmed_surface`.

Informal propositions:

It should be noted that a `load_element_distributed_surface_varying` cannot exist without four corresponding instances of `applied_load_static_pressure`.

Notes:

New for CIS/2.

The entity `rectangular_trimmed_surface` is defined in STEP Part 42.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

6.3.22 load_element_thermal

Entity definition:

A type of `load_element` where the load is described in terms of temperature gradients. The temperature gradients are specified by a list of between 1 and 3 real numbers corresponding to the three dimensions of the element.

EXPRESS specification:

```

*)
ENTITY load_element_thermal
SUBTYPE OF (load_element);
    temperature_gradients : LIST [1:3] OF measure_with_unit;
END_ENTITY;
(*)

```

Attribute definitions:*temperature_gradients*

Declares the instances of `measure_with_unit` used to define the `load_element_thermal`. There will be at least one and at most 3 references to instances of `measure_with_unit`. Where the temperature gradient is constant in all 3 directions, the 3 references may be made to the same instance of `measure_with_unit`. The `measure_with_unit` (taken from STEP Part 41) provides a generic mechanism with a real number value and an appropriate unit.

Notes:

New for CIS/2.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

6.3.23 load_member**Entity definition:**

A type of load applied to a particular member. The `load_member` may be described in terms of a concentrated or distributed load. The load can be applied in either local or global coordinate systems. It can also be classified as a destabilizing, for the purposes of member design. As a type of load, it inherits a parent load case, as well as a name and a description for the load. This entity is effectively an association between an `assembly_design_structural_member` and an `applied_load`. This may be used to describe loads applied to members in the design model and thus assist in the derivation of the analysis model. It is similar to the entity `load_element`, but is not a substitute for it.

EXPRESS specification:

*)

ENTITY `load_member`

ABSTRACT SUPERTYPE OF (ONEOF(`load_member_distributed`,
`load_member_concentrated`))

SUBTYPE OF (`load`);

`supporting_member` : `assembly_design_structural_member`;

`load_position_label` : OPTIONAL label;

`reference_system` : OPTIONAL text;

`destablizing_load` : OPTIONAL BOOLEAN;

`global_or_local` : `global_or_local_load`;

END_ENTITY;

(*

Attribute definitions:*supporting_member*

Declares the instance of `assembly_design_structural_member` associated with the load. While the member may be associated with (and thus support) zero, one or many instances of `load_member`, the `load_member` must be associated with (and located on) one (and only one) element.

load_position_label

A short text description of the position on the member at which the load is applied, e.g. mid-span. For distributed loads the position corresponds to the point where the initial value of the load is specified.

reference_system

A text description of the referencing system used to describe the loads on the member.

destablizing_load

Declares the classification of whether the load is destabilizing (TRUE) or not (FALSE), for the purposes of member design.

global_or_local

Declares the classification of the direction of the loading applied to this member. The loading direction can be specified as either being in the global coordinate system (X, Y, Z), or the in the member's local coordinate system (x, y, z).

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.24 load_member_concentrated**Entity definition:**

A type of load_member where the load is described as being applied at a point within the member. The value of the load is defined by a reference to a single instance of applied_load. For two-dimensional design models, some of the attributes representing the (out-of-plane) magnitude of the applied load may be assigned NULL values. For three-dimensional design models, all of the attributes representing the magnitude of the applied load should be assigned values, even if the value is zero.

If the load is not applied in the direction of one of the 3 (local or global) axes, then the values are the resolution of the load. If the load is applied in the direction of one of the 3 axes, then the remaining values shall be set to 0.0 (or NULL for two-dimensional design models).

The applied load can be described as static or dynamic. Static loads are defined in terms of displacements, forces, or pressures, while dynamic loads are defined in terms of velocities or accelerations, associated with sets of static loads.

EXPRESS specification:

*)

```
ENTITY load_member_concentrated
  SUBTYPE OF (load_member);
    load_position : point;
    load_value : applied_load;
  WHERE
```

```

WRL53 : NOT ('STRUCTURAL_FRAME_SCHEMA.
APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(load_value));
END_ENTITY;
(*)

```

Attribute definitions:

load_position

Declares the instance of point used to define the position of the concentrated load on the member in question. The point entity (from STEP Part 42) specifies the numerical value (in accordance with the unit system) of the position within the member, measured from the start of the member from its origin. For linear members, the value corresponding to the point's x-coordinate specifies the linear distance to the applied load measured along the member's x-axis from the member's origin.

load_value

Declares the instance of applied_load used to define the values of the load_member_concentrated. As applied_load is abstract, the SUBTYPEs are used. For example, when the applied load referred to is the SUBTYPE applied_load_static_force, a concentrated member load is defined in terms of three linear forces (Fx, Fy, Fz) and three moments (Mx, My, Mz). It should be noted that a load_member_concentrated cannot exist without a corresponding instance of one of the SUBTYPEs of applied_load.

Formal propositions:

WRL53

The applied_load used to define the values of the load_member_concentrated shall not be of the type applied_load_static_pressure.

Notes:

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.25 load_member_distributed

Entity definition:

A type of load_member where the load is considered to be distributed. This is a generic entity that is intended to cover all distributed loading applied to all members with a dimensionality greater than zero; including uniformly distributed, triangular, and the general patch load.

EXPRESS specification:

```

*)
ENTITY load_member_distributed
ABSTRACT SUPERTYPE OF (ONEOF
    (load_member_distributed_surface,
    load_member_distributed_curve))
SUBTYPE OF (load_member);
    projected_or_true : projected_or_true_length;
END_ENTITY;
(*)

```

Attribute definitions:*projected_or_true*

Declares the classification of the distributed load applied to a member as being applied in a true direction (where the load length is equal to the actual loaded length of member) or projected onto a member (such that the loaded length is increased and the actual magnitude of the distributed load is decreased).

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition

6.3.26 load_member_distributed_curve**Entity definition:**

A type of load_member_distributed where the loading is defined as being applied along a curve. The curve is defined using explicit geometry taken from STEP Part 42. This can apply to one, two or three dimensional members.

EXPRESS specification:

```

*)
ENTITY load_member_distributed_curve
SUPERTYPE OF (load_member_distributed_curve_line)
SUBTYPE OF (load_member_distributed);
    start_load_value : applied_load;
    end_load_value : applied_load;
    curve_definition : curve;
WHERE
    WRL54 : NOT ('STRUCTURAL_FRAME_SCHEMA.
    APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(start_load_value));
    WRL55 : NOT ('STRUCTURAL_FRAME_SCHEMA.
    APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(end_load_value));
    WRL56 : TYPE OF(start_load_value) = TYPEOF(end_load_value);
END_ENTITY;
(*

```

Attribute definitions:*start_load_value*

Declares the first instance of applied_load used to define the values at the beginning of the curve defining the load_member_distributed_curve. As applied_load is abstract, the SUBTYPES are used. For example, when the applied load referred to is the SUBTYPE applied_load_static_force, the start value of the distributed element load is defined in terms of 3 linear forces (Fx, Fy, Fz) and 3 moments (Mx, My, Mz). The instance of applied load defines the value for a unit length of the associated curve. The unit length is measured in the same units as those used to define the curve. Therefore the distributed load may need to be modified to suit the definition of the curve. For example, if the

curve has been defined in millimetres then a distributed load of 60kN/m must be expressed as a load of 60N. Similarly, if the curve has been defined in inches, a distributed load of 60lbf/ft must be expressed as a load of 5lbf.

end_load_value

Declares the second instance of `applied_load` used to define the values at the end of the curve defining the `load_member_distributed_curve`. (See also definition for `start_load_value`.)

curve_definition

Declares the instance of `curve` used to define the `load_member_distributed_curve`. The `curve` entity (taken from STEP Part 42) specifies the line along which the load is applied. The curve may (but need not be) a straight line. The curve may be bounded, trimmed, or segmented. The curve is defined in the local coordinate system of the member.

Formal propositions:

WRL54

The `applied_load` referenced by the attribute `start_load_value` shall not be of the type `applied_load_static_pressure`.

WRL55

The `applied_load` referenced by the attribute `end_load_value` shall not be of the type `applied_load_static_pressure`.

WRL56

The `applied_load` referenced by the attribute `start_load_value` shall be of the same type as that referenced by the attribute `end_load_value`.

Informal propositions:

It should be noted that a `load_member_distributed_curve` cannot exist without corresponding instances of one of the SUBTYPES of `applied_load`. Normally there would be two separate instances, but two references to the same instance may be used when the start and end values are the same.

It is assumed that the curve is defined in the same geometric context as the member; i.e. the same coordinate system and dimensionality. If the curve does not lie on the member, then the load is projected onto it from the curve. It is assumed that this projection meets the member – either in whole or in part.

Notes:

The entity `curve` is defined in STEP Part 42.

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.27 load_member_distributed_curve_line

Entity definition:

A type of `load_member_distributed_curve` where the curve is defined by a (straight) line. The line (taken from STEP Part 42) is a SUBTYPE of `curve`.

EXPRESS specification:

```

*)
ENTITY load_member_distributed_curve_line
SUBTYPE OF (load_member_distributed_curve);
WHERE
    WRL57 : 'STRUCTURAL_FRAME_SCHEMA.LINE' IN TYPE OF
        (SELFload_member_distributed_curve.curve_definition);
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity load_member_distributed_curve.

Formal propositions:

WRL57

The curve used to define the load_member_distributed_curve_line shall be of the type 'line'.

Informal propositions:

The line is defined in the local coordinate system of the member. Thus, the cartesian_point referenced by the line shall lie on the member and be defined relative to the member's coordinate system.

The instances of applied load (inherited from the SUPERTYPE load_member_distributed_curve) define the values for unit lengths of the associated line. The lengths are measured in the same units as those used to define the line. Therefore, the distributed load may need to be modified to suit the definition of the line. For example, if the line has been defined in millimetres then a distributed load of 60kN/m must be expressed as a load of 60N. Similarly, if the line has been defined in inches, a distributed load of 60lbf/ft must be expressed as a load of 5lbf.

Notes:

The entity line is defined in STEP Part 42.

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.28 load_member_distributed_surface**Entity definition:**

A type of load_member_distributed where the loading is defined over a surface. The surface is defined by reference to the entity bounded_surface (taken from STEP Part 42). This may be applied to two or three dimensional members. The values of the loading are defined by the SUBTYPES.

EXPRESS specification:

```

*)
ENTITY load_member_distributed_surface
ABSTRACT SUPERTYPE OF (ONEOF
    (load_member_distributed_surface_uniform,

```



```

        load_member_distributed_surface_varying))
SUBTYPE OF (load_member_distributed);
    surface_definition : bounded_surface;
END_ENTITY;
(*)

```

Attribute definitions:*surface_definition*

Declares the instance of `bounded_surface` used to define the boundary of the loading for the `load_member_distributed_surface`.

Formal propositions:*ABSTRACT SUPERTYPE*

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Informal propositions:

It is assumed that the `bounded_surface` is defined in the same geometric context as the member. If the `bounded_surface` does not lie on the surface of the member, then the load is projected onto the member's surface from the `bounded_surface`. It is assumed that this projection meets the surface of the member – either in whole or in part.

Notes:

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.29 load_member_distributed_surface_uniform**Entity definition:**

A type of `load_member_distributed_surface` where the load value is defined by a single reference to an instance of `applied_load_static_pressure` and is assumed to be constant across the `bounded_surface`.

EXPRESS specification:

```

*)
ENTITY load_member_distributed_surface_uniform
SUBTYPE OF (load_member_distributed_surface);
    load_value : applied_load_static_pressure;
END_ENTITY;
(*)

```

Attribute definitions:*load_value*

Declares the instance of `applied_load_static_pressure` used to define the loading values of the `load_member_distributed_surface_uniform`.

Informal propositions:

The load value is assumed to be constant across the `bounded_surface` of the `load_member_distributed_surface_uniform`.

Notes:

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.30 load_member_distributed_surface_varying**Entity definition:**

A type of `load_member_distributed_surface` where the load value is defined by multiple references to instances of `applied_load_static_pressure` and is assumed to vary across the `bounded_surface`.

EXPRESS specification:

*)

ENTITY `load_member_distributed_surface_varying`

SUBTYPE OF (`load_member_distributed_surface`);

`load_values` : SET [4:4] OF `applied_load_static_pressure`;

WHERE

WRL58 : 'STRUCTURAL_FRAME_SCHEMA.RECTANGULAR_TRIMMED_SURFACE' IN
TYPE OF (`SELF`/`load_element_distributed_surface.surface_definition`);

END_ENTITY;

(*

Attribute definitions:***load_values***

Declares the set of four instances of `applied_load_static_pressure` used to define the loading values of the `load_element_distributed_surface_varying`. Each of these is placed at a position on the element corresponding to the vertices of the `rectangular_trimmed_surface`.

The values of the parameters of the `applied_load_static_pressure` apply to a unit area of the `bounded_surface` on the element and are measured in the same units that define the `bounded_surface`. The values of the applied distributed load may need to be converted to suit the `bounded_surface`. A linear distribution of pressure is assumed to occur between each vertex of the `bounded_surface`.

Formal propositions:**WRL58**

The `bounded_surface` referenced by the attribute `surface_definition` (inherited from the SUPERTYPE `load_member_distributed_surface`) shall be of the type `rectangular_trimmed_surface`.

Informal propositions:

It should be noted that a `load_member_distributed_surface_varying` cannot exist without four corresponding instances of `applied_load_static_pressure`.

Notes:

The entity `rectangular_trimmed_surface` is defined in STEP Part 42.

See diagram 76 of the EXPRESS-G diagrams in Appendix B.

New for 2nd Edition.

6.3.31 load_node

Entity definition:

A type of load where the load is applied to a specific node. It is effectively an associative entity resolving a many-to-many relationship between node and applied_load. As a type of load, it inherits a parent load case, as well as a number, label and description for the load.

EXPRESS specification:

```
*)
ENTITY load_node
SUBTYPE OF (load);
    supporting_node: node;
    load_values : applied_load;
WHERE
    WRL43 : NOT ('STRUCTURAL_FRAME_SCHEMA.
    APPLIED_LOAD_STATIC_PRESSURE' IN TYPE OF(load_values));
END_ENTITY;
(*
```

Attribute definitions:

supporting_node

Declares the instance of node associated with (and therefore supporting) the load_node.

load_values

Declares the instance of applied_load associated with (and therefore defining the values of) the load_node. As applied_load is abstract, the SUBTYPEs are used. For example, when the applied load referred to is the SUBTYPE applied_load_static_force, the load_node is defined in terms of 3 linear forces (Fx, Fy, Fz) and 3 moments (Mx, My, Mz).

Formal propositions:

WRL43

The applied_load referenced by the attribute load_values shall not be of the type applied_load_static_pressure.

Informal propositions:

A load_node cannot exist without corresponding instances of node and applied_load. Because INVERSE clauses are not specified in the node and the applied_load, the default cardinality applies such that any number of instances of load_node can be associated with these items. In other words, a load_case may be associated with zero, one or many instances of load_node, and a node may be associated with zero, one or many instances of load_node.

Notes:

Known as NODE_LOAD in CIS/1.

See diagram 27 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd edition.

6.3.32 loaded_product

Entity definition:

An association between a structural_frame_product and an applied_load. This is a generic construct allowing three other generic constructs to be combined. This may be used in place of an analytical model to describe the situation where products are loaded directly. (It should be noted that in LPM/6 assembly and part are types of product.).

EXPRESS specification:

```
*)
ENTITY loaded_product;
    product : structural_frame_product;
    load_definition : applied_load;
    time_variation : OPTIONAL physical_action;
END_ENTITY;
(*
```

Attribute definitions:

product

Declares the instance of structural_frame_product associated with the loaded_product. That is, the product to which the load is applied.

load_definition

Declares the instance of applied_load associated with the loaded_product. This provides the definition of the load that is applied to the product under consideration.

time_variation

Declares the instance of physical_action that may be associated with the loaded_product. The SUBTYPEs of the entity physical_action classify the loading by the dependency on time of the physical actions. These physical actions can be permanent, variable, accidental, or seismic.

Informal propositions:

A loaded_product cannot exist without corresponding instances of structural_frame_product and applied_load. Because INVERSE clauses are not specified in the structural_frame_product and the applied_load, the default cardinality applies such that any number of instances of loaded_product can be associated with these items. In other words, a structural_frame_product may be associated with zero, one or many instances of loaded_product, and a applied_load may be associated with zero, one or many instances of loaded_product.

Notes:

New for CIS/2.

See diagram 20 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition

6.3.33 loading_combination

Entity definition:

Combination of load cases that assumed to be acting simultaneously. The loading combination is formed by collating instances of load_case via the

load_combination_occurrence. The loading_combination must be given a unique alphanumeric reference and may also be given a description. An example is shown in Figure 6.1.

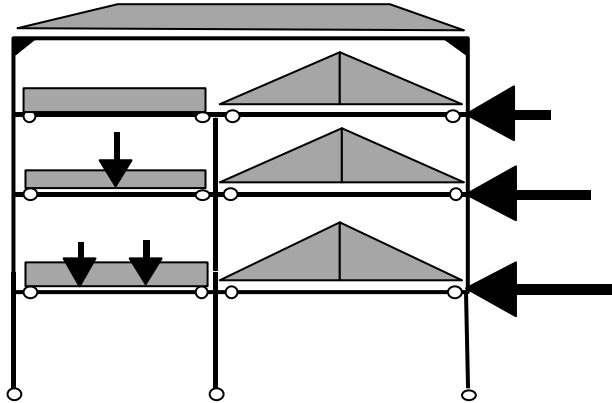


Figure 6.1 *Defining a loading combination*

EXPRESS specification:

```

*)
ENTITY loading_combination;
    loading_combination_name : label;
    loading_combination_description : OPTIONAL text;
    loaded_model : OPTIONAL analysis_model;
DERIVE
    cases : SET [1:?] OF load_combination_occurrence := bag_to_set
        (USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.
            LOAD_COMBINATION_OCCURRENCE.PARENT_LOAD_COMBINATION'));
INVERSE
    load_cases : SET [1:?] OF load_combination_occurrence FOR parent_load_combination;
END_ENTITY;
(*

```

Attribute definitions:

loading_combination_name

A short alphanumeric reference for the loading_combination.

loading_combination_description

A free text description of the loading_combination.

loaded_model

Declares the instance of analysis_model that may be associated with (and therefore loaded by) the loading_combination. The instance of analysis_model referenced here must correspond to the instance of analysis_model that may be derived from the associated nodes and elements that are loaded by the loads in this loading combination.

cases

Derives the set of one or more instances of load_combination_occurrence that make reference to this instance of loading_combination. This derived attribute complements the inverse clause.

Formal propositions:**load_cases**

The loading_combination shall be the parent of at least one component load_combination_occurrence. The loading_combination cannot exist without being referenced by at least one instance of load_combination_occurrence.

Notes:

Known as LOADING_COMBINATION in CIS/1.

See diagram 30 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – Derive clause added.

6.3.34 physical_action**Entity definition:**

A physical_action is a force applied to the structure, an imposed deformation or an imposed acceleration. This is a conceptual mechanism used to classify physical actions by their source. A physical_action can be described as being either permanent, variable, accidental, or seismic. The SUBTYPES of physical_action provide further classification. The physical_action must be classified by:

- its nature (static, dynamic, or quasi-dynamic),
- its spatial variation (fixed or free), and
- its type (direct or indirect).

As part of the design loading specification, it provides a set of descriptions of the loads predicted to act on the structure of a building (or building complex) together with values for factors of safety. This set of descriptions would normally be in the form of text, and numerical information. It serves to inform structural modellers, detail designers, concept designers, architects, building services designers, and constructors.

A physical_action may be given a basic magnitude and a derived magnitude. The physical_action may also be given a list of derivation_factors and derivation_factor_labels.

EXPRESS specification:

*)

ENTITY physical_action

SUPERTYPE OF (ONEOF

(physical_action_permanent,
physical_action_variable,
physical_action_accidental,
physical_action_seismic));

action_nature : static_or_dynamic;

action_spatial_variation : spatial_variation;

action_type : direct_or_indirect_action;

basic_magnitude : OPTIONAL measure_with_unit;

derived_magnitude : OPTIONAL measure_with_unit;

derivation_factors : LIST [0:?] OF REAL;

derivation_factor_labels : LIST [0:?] OF label;

WHERE

```

WRP11 : SIZEOF (derivation_factors) = SIZEOF (derivation_factor_labels);
END_ENTITY;
(*)

```

Attribute definitions:

action_nature

Declares the classification of the type of physical action by its nature as either static, dynamic, or quasi-dynamic. According the Eurocode 1^[11], a static action is an action that does not cause significant acceleration of the structure or structural members, while a dynamic action is one that does cause significant acceleration of the structure or structural members. A quasi-dynamic action is a dynamic action that can be described by static models in which the dynamic effects are included.

action_spatial_variation

Declares the classification of the physical action by its spatial variation; i.e. the ability of the action to move. Actions are classified as being either fixed (unable to move) or free to move. According to Eurocode 1, a fixed action is an action that has a fixed distribution over the structure such that the magnitude and direction of the action are determined unambiguously for the whole structure if this magnitude and direction are determined at one point on the structure. A free action is an action that may have any spatial distribution over the structure within given limits.

action_type

Declares the classification of the type of action as either direct or indirect; i.e. the structure may be loaded directly or indirectly through the components it supports such as cladding. A direct action may be a force (load) applied to the structure. An indirect action may be an imposed or constrained deformation or an imposed acceleration caused by, for example, temperature changes, moisture variation, uneven settlement or earthquakes.

basic_magnitude

Declares the first reference to an instance of *measure_with_unit* used to define the physical action. The *measure_with_unit* (taken from STEP Part 41) provides a generic mechanism with a real number value and an appropriate unit. For example, this may be the basic wind pressure for the site as given by the appropriate code of practice.

derived_magnitude

Declares the second reference to an instance of *measure_with_unit* used to define the physical action. The *measure_with_unit* (taken from STEP Part 41) provides a generic mechanism with a real number value and an appropriate unit.

The derived magnitude would normally be derived from the *basic_magnitude* using the *derivation_factors* supplied. For example, this may be the design wind pressure for the structure taking into account the form of the structure and its surrounding environment. In this case, the basic wind pressure would be multiplied by a number of factors as given by the appropriate code of practice.

derivation_factors

An ordered list (which may include repetitions) of real numbers that are used to define the physical action. The list of real numbers are multiplied together with the *basic_magnitude* to produce the *derived_magnitude*.

derivation_factor_labels

An ordered list (which may include repetitions) of short text references for the *derivation_factors*.

It is assumed that the size of the list of *derivation_factors* is equal to the size of the list of *derivation_factor_labels*. Because these lists are ordered it is assumed that the list of labels is in the same order as the list of factors. For example, the first label corresponds to the first factor.

Formal propositions:*WRP11*

The size of the list of instances referenced by the attribute *derivation_factors* shall equal the size of the list of instances referenced by the attribute *derivation_factor_labels*.

Notes:

New for CIS/2; partially addressed by BASIC_LOAD_CASE in CIS/1.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

The attributes *derivation_factor* and *derivation_factor_labels* are defined as LISTs whose lower bounds (minimum number of items in the aggregation) are zero. In other words, empty lists are allowed. This allows the application to leave the aggregation empty. Nevertheless, since the attributes are not optional, null values are not allowed. As a minimum, the attributes should be populated as empty lists.

6.3.35 physical_action_accidental**Entity definition:**

A type of physical action, usually of short duration, which is unlikely to occur with a significant magnitude over the period of time under consideration during the design working life. An accidental action can be expected in many cases to cause severe consequences unless special measures are taken.

EXPRESS specification:

```
*)
ENTITY physical_action_accidental
  SUBTYPE OF (physical_action);
    action_source : action_source_accidental;
END_ENTITY;
(*
```

Attribute definitions:*action_source*

Declares the classification of the type of accidental action (load) by its source; that is the action may be specified as being caused by either:

- fire,
- impulsive force (such as shock wave from an explosion),
- an impact of a solid object, or
- its source may be undefined.

Notes:

New for CIS/2.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.36 physical_action_permanent

Entity definition:

A type of physical_action which is likely to act throughout a given design situation and for which the variation of magnitude with time is negligible in relation to the mean value, or for which the variation is always in the same direction (monotonic) until the action attains a certain limit value.

EXPRESS specification:

```
*)
ENTITY physical_action_permanent
SUBTYPE OF (physical_action);
    action_source : action_source_permanent;
END_ENTITY;
(*
```

Attribute definitions:

action_source

Declares the classification of the type of permanent action (load) by its source; that is the action may be specified as being caused by either:

- dead loads (e.g. walls, floors, and permanent finishes, fixtures and fittings),
- by the structure's self weight,
- a prestress applied to the structure,
- by lack of fit (arising from inaccuracies in the dimensions of the as-built structural members), or
- the source of the permanent action may be undefined.

Notes:

New for CIS/2; partially addressed by DEAD_LOAD in CIS/1.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.37 physical_action_seismic

Entity definition:

A type of physical_action which arises due to earthquake ground motions.

EXPRESS specification:

```
*)
ENTITY physical_action_seismic
SUBTYPE OF (physical_action);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `physical_action`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Notes:

New for CIS/2.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.38 `physical_action_variable`**Entity definition:**

A type of `physical_action` which is unlikely to act throughout a given design situation or for which the variation in magnitude with time is neither negligible in relation to the mean value nor monotonic. Variable actions are defined as either long term, transient, or short term.

EXPRESS specification:

```

*)
ENTITY physical_action_variable
ABSTRACT SUPERTYPE OF (ONEOF
    (physical_action_variable_long_term,
    physical_action_variable_transient,
    physical_action_variable_short_term))
SUBTYPE OF (physical_action);
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `physical_action`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity, and to collect the SUBTYPEs under a common category.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Notes:

New for CIS/2; partially addressed by `LIVE_LOAD` in CIS/1.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.39 physical_action_variable_long_term

Entity definition:

A type of physical_action_variable where the given design situation is persistent (the conditions of normal use during the life of the structure). These are generally loads imposed on the structure by its use.

EXPRESS specification:

```
*)
ENTITY physical_action_variable_long_term
SUBTYPE OF (physical_action_variable);
    action_source : action_source_variable_long_term;
END_ENTITY;
(*
```

Attribute definitions:

action_source

Declares the classification of the type of long term variable action (load) by its source; that is the action may be specified as being caused by:

- live loads (e.g. load imposed during the normal use of the structure by its occupants, furniture, movable items, etc.),
- system imperfection (arising from changes in the structural members during the structure's life time),
- settlement (arising from ground movements during the structure's life time),
- temperature effect (arising from thermal changes during the structure's life time),
- or the source of the long term variable action may be undefined.

Notes:

New for CIS/2; partially addressed by LIVE_LOAD in CIS/1.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.40 physical_action_variable_short_term

Entity definition:

A type of physical_action_variable where the given design situation is not persistent. These are actions that will arise only occasionally during the life of the structure. These are generally environmental loads, that is loads imposed on the structure by its environment. The duration of the loading 'event' is normally short when compared to the life of the structure.

EXPRESS specification:

```
*)
ENTITY physical_action_variable_short_term
SUBTYPE OF (physical_action_variable);
    action_source : action_source_variable_short_term;
END_ENTITY;
(*
```

Attribute definitions:***action_source***

Declares the classification of the type of short term variable action (load) by its source; that is the action may be specified as being caused by:

- buoyancy (applicable to floating of submerged structures),
- wind,
- snow,
- ice (e.g. ice build up on cables or transmission towers),
- current (i.e. constantly moving water - applicable to floating or submerged structures),
- wave, (i.e. cyclical moving water applicable to floating or submerged structures),
- rain,
- or the source of the short term variable action may be undefined.

Notes:

New for CIS/2; partially addressed by ENVIRONMENTAL_LOAD in CIS/1.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

6.3.41 physical_action_variable_transient**Entity definition:**

A type of physical_action_variable where the given design situation is transient (referring to temporary conditions applicable to the structure, e.g. during execution or repair. These are actions that will arise only occasionally during the life of the structure. These are generally temporary loads, that is loads imposed on the structure by particular events such as erection, transportation, propping, or repair. The duration of the loading ‘event’ is normally short when compared to the life of the structure.

EXPRESS specification:

```
*)
ENTITY physical_action_variable_transient
  SUBTYPE OF (physical_action_variable);
    action_source : action_source_variable_transient;
END_ENTITY;
(*
```

Attribute definitions:***action_source***

Declares the classification of the type of transient variable action (load) by its source; that is the action may be specified as being caused by:

- transport (from fabrication shop to site),
- erection (from initial to final position),
- propping (temporary support of structural members during construction), or

- the source of the transient variable action may be undefined.

Notes:

New for CIS/2; partially addressed by LIVE_LOAD in CIS/1.

See diagram 26 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

7 LPM/6 STRUCTURAL RESPONSE

7.1 Structural Response concepts and assumptions

7.1.1 Conceptual overview

This subject area contains constructs that represent the response of the structure (in the form of an analysis model) to the applied loads. Results of the analysis may be given at nodes, element ends, or at a point within the element. These analysis results can be described in terms of displacements, forces, velocities or accelerations, and may be combined as sets. A structural response model (as illustrated in Figure 7.1) basically consists of one or more sets of analysis results. These analysis results sets are categorized (through subtyping) as either basic results (for a basic load case), or combined results (for a loading combination). Where the results cover the ranges of extremes (based on a number of loading combinations) the set is categorized as an envelope result.

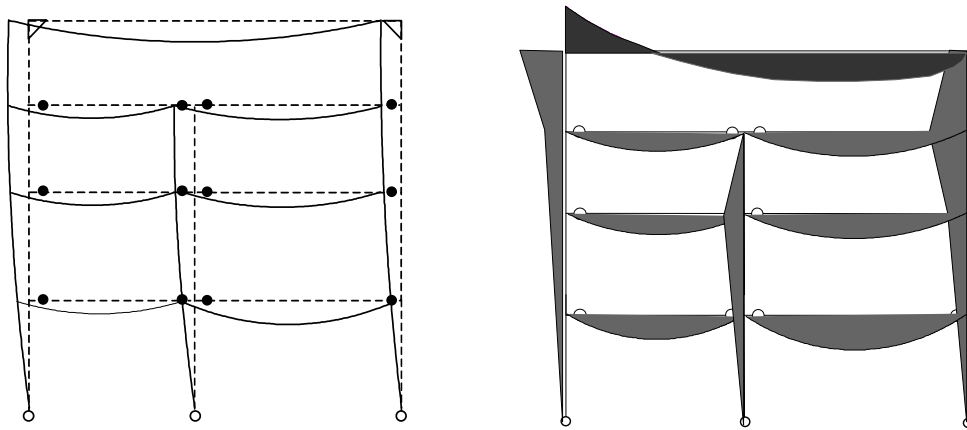


Figure 7.1 *Examples of structural response models*

The sign convention is as described for the structural analysis model shown in Figure 5.2.

Analysis results are categorized (through subtyping) as either a result for an element (`analysis_result_element`), for a node (`analysis_result_node`), or for the point where an element meets a node (`analysis_result_element_node`).

7.2 Structural Response type definitions

No types have been added or modified for the 2nd Edition in this subject area.

7.2.1 `elastic_or_plastic_resistance`

Type definition:

Classifies the resistance as either elastic or plastic resistance. This is used as a data type in the entity resistance. If the resistance is specified as being ‘elastic’, the instance of resistance is defining the capacity of the structural component in an elastic state; i.e. one in which the material has not reached yield and Hook’s Law still applies. If it is specified as being ‘plastic’, the instance of resistance is defining the capacity of the structural component in a plastic state; i.e. one in which the material has been taken beyond its yield point and Hook’s Law is no longer applicable.

EXPRESS specification:

```
*)
TYPE elastic_or_plastic_resistance
= ENUMERATION OF
    (elastic_resistance, plastic_resistance);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged for 2nd Edition.

7.2.2 global_or_local_resistance

Type definition:

Classifies the resistance as either global or local resistance. This is used as a data type in the entity resistance. If the resistance is specified as being ‘global’, the instance of resistance is defining the overall capacity of the structural component. If it is specified as being ‘local’, the instance of resistance is defining the capacity of the structural component at a particular position on that component. (The position is defined elsewhere.)

EXPRESS specification:

```
*)
TYPE global_or_local_resistance
= ENUMERATION OF
    (global_resistance, local_resistance);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged for 2nd Edition.

7.2.3 maximum_or_minimum

Type definition:

Classifies the type of value as either maximum or minimum.

EXPRESS specification:

```
*)
TYPE maximum_or_minimum
= ENUMERATION OF
    (maximum, minimum);
END_TYPE;
(*
```

Notes:

Known as maximum_or_minimum in CIS/1.

7.3 Structural Response entity definitions

No entities have been added or modified for the 2nd Edition in this subject area.

7.3.1 analysis_result

Entity definition:

An analysis_result is the mechanism in the LPM that identifies a single result of a particular analysis procedure performed using a particular analysis method. Its SUBTYPES are used to categorize the result as belonging to either a node, an element, or an element node (where an element meets a node). It will be instanced as one of its SUBTYPES.

EXPRESS specification:

```
*)
ENTITY analysis_result
ABSTRACT SUPERTYPE OF (ONEOF
    (analysis_result_node,
     analysis_result_element_node,
     analysis_result_element));
analysis_result_name : label;
sign_convention : OPTIONAL text;
results_for_analysis : analysis_method;
UNIQUE
    URA1 : analysis_result_name;
END_ENTITY;
(*
```

Attribute definitions:

analysis_result_name

A short alphanumerical reference for the analysis_result.

sign_convention

An optional text description of the sign convention used for the analysis_result. (See Figure 5.2 for the sign convention used in LPM/6.)

results_for_analysis

Declares the instance of analysis_method associated with this analysis_result. The result must be declared as belonging to one (and only one) analysis_method.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

URA 1

The analysis_result_name shall be unique to the analysis_result.

Notes:

Known as ANALYSIS_RESULT in CIS/1.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.2 analysis_result_element

Entity definition:

A type of analysis_result where the result of the analysis is given for an element. Because there are several different types of element, the analysis_result_element must be instanced through one of its SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY analysis_result_element
ABSTRACT SUPERTYPE OF (ONEOF
    (analysis_result_element_curve,
     analysis_result_element_surface,
     analysis_result_element_point,
     analysis_result_element_volume))
SUBTYPE OF (analysis_result);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE analysis_result. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity, and to collate the SUBTYPEs under a common category.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Notes:

Known as ELEMENT_RESULT in CIS/1.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.3 analysis_result_element_curve

Entity definition:

A type of analysis_result where the result of the analysis is given for a curved element. This entity captures an association between the entity element_curve and the entity reaction. It represents a single result from the analysis at one position along the length of the element. The result (i.e. the reaction) must be given a position along the length of the curve of the element. This position may be given a label.

EXPRESS specification:

```
*)
ENTITY analysis_result_element_curve
SUBTYPE OF (analysis_result_element);
    result_for_element_curve : element_curve;
    x_increasing : BOOLEAN;
    result_values : reaction;
```

```

    result_position : length_measure_with_unit;
    position_label : OPTIONAL label;
END_ENTITY;
(*)

```

Attribute definitions:

result_for_element_curve

Declares the instance of `element_curve` associated with the `analysis_result_element_curve`. The `analysis_result_element_curve` must be associated with one (and only one) `element_curve`.

x_increasing

Specifies whether the analysis result has been calculated on the x-increasing (TRUE) or x-decreasing (FALSE) side of the `result_position`. For example, if the element is viewed such that its x-axis runs from left to right, and the element result is for a position below a load point, then if `x_increasing` is TRUE the result is for the part of the element to the right of the load. Similarly, `x_increasing` is FALSE if the result is for the element part to the left of the load. (The shape of the shear force diagram, for example, changes at a load point.)

result_values

Declares the instance of reaction associated with the `analysis_result_element_curve`. The `analysis_result_element_curve` must be associated with one (and only one) reaction (see definition for reaction).

result_position

Declares the instance of `length_measure_with_unit` associated with the `analysis_result_element_curve`, which specifies the position on the curved element at which the analysis result has been calculated. The `analysis_result_element_curve` must be associated with one (and only one) `length_measure_with_unit`; i.e. the analysis result is for one particular position on the element.

position_label

An optional short text description for the position on the element for which the results have been calculated; e.g. 'load-point'.

Notes:

New for CIS/2; but partially addressed by `ELEMENT_RESULT` in CIS/1.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.4 analysis_result_element_node

Entity definition:

A type of `analysis_result` where the result of the analysis is given for the point where an element meets a particular node. This entity captures an association between the entity `element_node_connectivity` and the entity reaction. It represents one result of the analysis for one connectivity.

EXPRESS specification:

*)
 ENTITY analysis_result_element_node
 SUBTYPE OF (analysis_result);
 result_for_element_node : element_node_connectivity;
 result_values : reaction;
 END_ENTITY;
 (*

Attribute definitions:*result_for_element_node*

Declares the instance of element_node_connectivity associated with the analysis_result_element_node. The analysis_result_element_node must be associated with one (and only one) element_node_connectivity.

result_values

Declares the instance of reaction associated with the analysis_result_element_node. The analysis_result_element_node must be associated with one (and only one) reaction (see definition for reaction).

Notes:

Known as ELMNT_NODE_RESULT in CIS/1.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.5 analysis_result_element_point**Entity definition:**

A type of analysis_result where the result of the analysis is given for a point element. This entity captures an association between the entity element_point and the entity reaction. It represents one result for the analysis for the point element.

EXPRESS specification:

*)
 ENTITY analysis_result_element_point
 SUBTYPE OF (analysis_result_element);
 result_for_element_point : element_point;
 result_values : reaction;
 END_ENTITY;
 (*

Attribute definitions:*result_for_element_point*

Declares the instance of element_point associated with the analysis_result_element_point. The analysis_result_element_point must be associated with one (and only one) element_point.

result_values

Declares the instance of reaction associated with the analysis_result_element_point. The analysis_result_element_point must be associated with one (and only one) reaction (see definition for reaction).

Notes:

New for CIS/2.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.6 analysis_result_element_surface**Entity definition:**

A type of analysis_result where the result of the analysis is given for a surface element. This entity will normally be instantiated as one of its SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY analysis_result_element_surface
ABSTRACT SUPERTYPE OF (ONEOF
    (analysis_result_element_surface_stresses,
     analysis_result_element_surface_tractions))
SUBTYPE OF (analysis_result_element);
    result_for_element_surface : element_surface;
    result_position : point;
    position_label : OPTIONAL label;
END_ENTITY;
(*
```

Attribute definitions:*result_for_element_surface*

Declares the instance of element_surface associated with the analysis_result_element_surface. The analysis_result_element_surface must be associated with one (and only one) element_surface.

result_position

Declares the instance of point associated with the analysis_result_element_surface, which specifies the position on the surface element at which the analysis result has been calculated. The analysis_result_element_surface must be associated with one (and only one) point; i.e. the analysis result is for one particular position on the element. The point is defined in the local coordinate system of the element.

position_label

An optional short text description for the position on the element for which the results have been calculated; e.g. 'load-point'.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instantiated on its own - it must be instantiated with one of its SUBTYPEs.

Notes:

New for CIS/2.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.7 analysis_result_element_surface_stresses**Entity definition:**

A type of analysis result for a surface element where the results of the analysis are expressed in terms of three stresses. This entity makes three associations with the entity `pressure_measure_with_unit`, which defines the value and the unit of each of the three stresses.

Any point on the surface element (such as a plate) will be subjected to 6 stresses: 3 direct and 3 shear stresses. However, when considering the response of a plate, the approach customarily employed is termed 'plane stress idealization'. As the thickness of the surface element (or plate) is small compared to its other 2 dimensions, the stresses having components in the through thickness direction are negligible. Although the out of plane displacement is not zero, it is assumed to be small enough to ignore out of plane displacement effects.

EXPRESS specification:

*)

ENTITY `analysis_result_element_surface_stresses`

SUBTYPE OF (`analysis_result_element_surface`);

`direct_stress_sigma_y` : `pressure_measure_with_unit`;

`membrane_stress_tau_yz` : `pressure_measure_with_unit`;

`direct_stress_sigma_z` : `pressure_measure_with_unit`;

END_ENTITY;

(*)

Attribute definitions:*direct_stress_sigma_y*

Declares the first instance of `pressure_measure_with_unit` associated with the `analysis_result_element_surface_stresses`, which specifies the value, with an appropriate unit, of the direct (or normal) stress in the y-direction for the surface element.

membrane_stress_tau_yz

Declares the second instance of `pressure_measure_with_unit` associated with the `analysis_result_element_surface_stresses`, which specifies the value, with an appropriate unit, of the membrane shear stress in the yz-plane for the surface element.

direct_stress_sigma_z

Declares the third instance of `pressure_measure_with_unit` associated with the `analysis_result_element_surface_stresses`, which specifies the value, with an appropriate unit, of the direct (or normal) stress in the z-direction for the surface element.

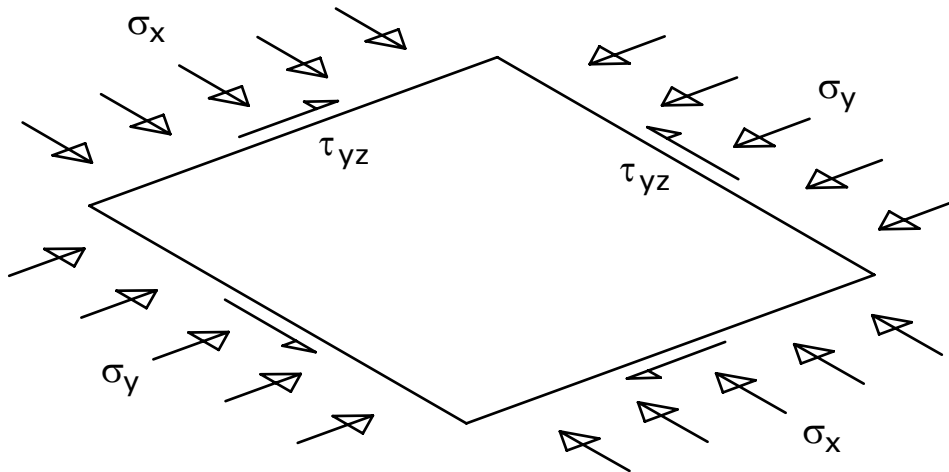


Figure 7.2 Analysis results for a surface element (stresses)

Notes:

New for CIS/2.

See diagram 32 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.8 analysis_result_element_surface_tractions

Entity definition:

A type of analysis result for a surface element where the results of the analysis are expressed in terms of forces per unit length. This entity makes four associations with the entity `force_measure_with_unit`, which define the values and the units of the two bending tractions and the two torsional tractions. (The moment per length (e.g. kNm/m) has been resolved to a force unit (e.g. kN).) It also makes four associations with the entity `force_per_length_measure_with_unit`, which define the values and the units of the two thrusts and the two shear forces in the y and z directions.

In most cases the thrusts will be set to zero as the out of plane displacements will be assumed to be small. However, this construct is intended to allow the results of the analysis to be represented when the analyst has used large displacement theory.

EXPRESS specification:

*)

```
ENTITY analysis_result_element_surface_tractions
SUBTYPE OF (analysis_result_element_surface);
  thrust_tz : OPTIONAL force_per_length_measure_with_unit;
  bending_traction_my : force_measure_with_unit;
  thrust_ty : OPTIONAL force_per_length_measure_with_unit;
  torsional_traction_mzy : force_measure_with_unit;
  torsional_traction_myz : force_measure_with_unit;
  shear_traction_qz : force_per_length_measure_with_unit;
  shear_traction_qy : force_per_length_measure_with_unit;
  bending_traction_mz : force_measure_with_unit;
```

END_ENTITY;

(*

Attribute definitions:***thrust_tz***

Declares the first instance of `force_per_length_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the thrust (or in plane force traction) in the z-direction for the surface element.

bending_traction_my

Declares the first instance of `force_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the flexural traction (bending moment per unit width) in the y-direction about the z-axis for the surface element.

thrust_ty

Declares the second instance of `force_per_length_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the thrust (or in plane force traction) in the y-direction for the surface element.

torsional_traction_mzy

Declares the second instance of `force_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the torsional traction (torque per unit width) in the z-direction about the y-axis for the surface element.

torsional_traction_myz

Declares the third instance of `force_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the torsional traction in the y direction about the z-axis for the surface element.

shear_traction_qz

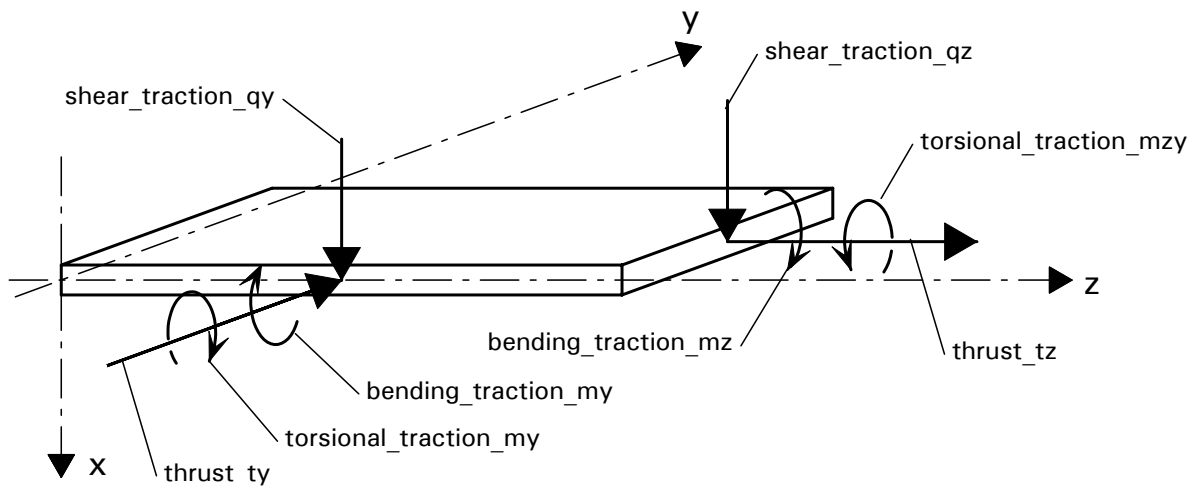
Declares the third instance of `force_per_length_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the shear traction in the y-direction for the surface element.

shear_traction_qy

Declares the fourth instance of `force_per_length_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the shear traction in the z-direction for the surface element.

bending_traction_mz

Declares the fourth instance of `force_measure_with_unit` associated with the `analysis_result_element_surface_tractions`, which specifies the value, with an appropriate unit, of the flexural traction (bending moment per unit width) in the z-direction about y-axis for the surface element.



Note: Complementary forces and moments not shown for clarity
 Direction of forces refer to positive direction rather than actual direction of forces

Figure 7.3 Analysis results for a surface element (tractions)

Notes:

New for CIS/2.

See diagram 32 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.9 analysis_result_element_volume

Entity definition:

A type of analysis_result where the result of the analysis is given for a volume element. This will normally be instantiated as one of its SUBTYPEs.

EXPRESS specification:

```

*)
ENTITY analysis_result_element_volume
ABSTRACT SUPERTYPE OF (analysis_result_element_volume_stress_tensor)
SUBTYPE OF (analysis_result_element);
  result_for_element_volume : element_volume;
  result_position : point;
  position_label : OPTIONAL label;
END_ENTITY;
(*
  
```

Attribute definitions:

result_for_element_volume

Declares the instance of element_volume associated with this analysis_result_element_volume. The analysis_result_element_volume must be associated with one (and only one) element_volume.

result_position

Declares the instance of point associated with the analysis_result_element_volume, which specifies the position within or on the surface of the volume element at which the analysis result has been calculated. The analysis_result_element_volume must be associated with

one (and only one) point; i.e. the analysis result is for one particular position on the element. The point is defined in the local coordinate system of the element.

position_label

An optional short text description for the position on or within the element for which the results have been calculated.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.10 analysis_result_element_volume_stress_tensor

Entity definition:

A type of *analysis_result_element_volume* where the results of the analysis are represented by 3 values for normal stress and 6 values for shear stress. This entity associates the volume element with 9 instances of *pressure_measure_with_unit* to give the appropriate values for the stress tensor matrix. This represents the most general state of stress acting on an element.

The values of the attributes of this entity could be used to populate a matrix as follows:

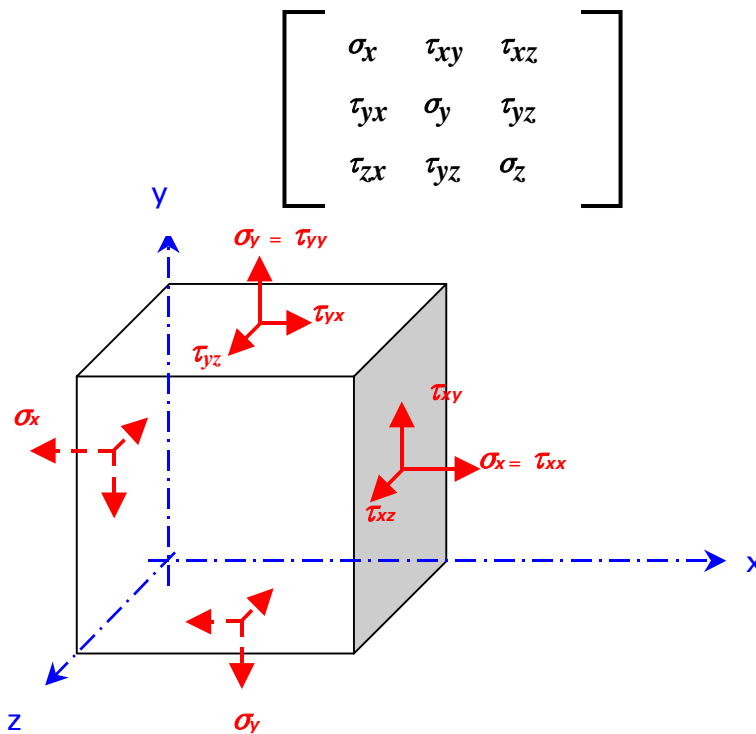


Figure 7.4 Analysis results for a volume element (stress tensor)

EXPRESS specification:

*)

ENTITY analysis_result_element_volume_stress_tensor

SUBTYPE OF (analysis_result_element_volume);

shear_stress_tau_zy : pressure_measure_with_unit;

shear_stress_tau_xz : pressure_measure_with_unit;

normal_stress_sigma_z : pressure_measure_with_unit;

normal_stress_sigma_y : pressure_measure_with_unit;

normal_stress_sigma_x : pressure_measure_with_unit;

shear_stress_tau_zx : pressure_measure_with_unit;

shear_stress_tau_yz : pressure_measure_with_unit;

shear_stress_tau_yx : pressure_measure_with_unit;

shear_stress_tau_xy : pressure_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:*shear_stress_tau_zy*

Declares the first instance of pressure_measure_with_unit associated with the analysis_result_element_volume_stress_tensor, which specifies the value, with an appropriate unit, of the shear stress (τ) in the y-direction for the volume element.

shear_stress_tau_xz

Declares the second instance of pressure_measure_with_unit associated with the analysis_result_element_volume_stress_tensor, which specifies the value, with an appropriate unit, of the shear stress (τ) in the x-direction for the volume element.

normal_stress_sigma_z :

Declares the third instance of pressure_measure_with_unit associated with the analysis_result_element_volume_stress_tensor, which specifies the value, with an appropriate unit, of the normal stress (σ) in the z-direction for the volume element.

normal_stress_sigma_y

Declares the fourth instance of pressure_measure_with_unit associated with the analysis_result_element_volume_stress_tensor, which specifies the value, with an appropriate unit, of the normal stress (σ) in the y-direction for the volume element.

normal_stress_sigma_x

Declares the fifth instance of pressure_measure_with_unit associated with the analysis_result_element_volume_stress_tensor, which specifies the value, with an appropriate unit, of the normal stress (σ) in the x-direction for the volume element.

shear_stress_tau_zx

Declares the sixth instance of pressure_measure_with_unit associated with the analysis_result_element_volume_stress_tensor, which specifies the value, with an appropriate unit, of the shear stress (τ) in the z-direction for the volume element.

shear_stress_tau_yz

Declares the seventh instance of `pressure_measure_with_unit` associated with the `analysis_result_element_volume_stress_tensor`, which specifies the value, with an appropriate unit, of the shear stress (τ) in the y-direction for the volume element.

shear_stress_tau_yx

Declares the eighth instance of `pressure_measure_with_unit` associated with the `analysis_result_element_volume_stress_tensor`, which specifies the value, with an appropriate unit, of the shear stress (τ) in the y-direction for the volume element.

shear_stress_tau_xy

Declares the ninth instance of `pressure_measure_with_unit` associated with the `analysis_result_element_volume_stress_tensor`, which specifies the value, with an appropriate unit, of the shear stress (τ) in the y-direction for the volume element.

Notes:

New for CIS/2.

See diagram 32 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.11 analysis_result_node**Entity definition:**

A type of analysis result where the results of the analysis are given for a node of the analysis model. This entity associates an instance of the entity node with an instance of the entity reaction, which provides the values with appropriate units for the analysis results. (See also definition for reaction.)

EXPRESS specification:

*)

```
ENTITY analysis_result_node
SUBTYPE OF (analysis_result);
    result_for_node : node;
    result_values : reaction;
END_ENTITY;
(*
```

Attribute definitions:*result_for_node*

Declares the instance of node associated with this analysis result. There must be one (and only one) instance of node referenced by `analysis_result_node`.

result_values

Declares the instance of reaction associated with this analysis result. There must be one (and only one) instance of reaction referenced by `analysis_result_node`.

Notes:

Known as `NODE_RESULT` in CIS/1.

See diagram 31 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.12 analysis_results_set

Entity definition:

This entity allows analysis results to be grouped together as sets. These sets can (but need not) be categorized (through its SUBTYPES) as being a set of results for either basic loads or combined loads. These SUBTYPES associate the set with a basic load case or a combined load case, respectively. Where the results cover the ranges of extremes (based on a number of loading combinations) the set is categorized as an envelope result. The remaining SUBTYPE allow the set to be categorized as a set of redistributed results, where redistribution factor have been applied to the results.

The analysis_results_set must be given a name and must contain at least one analysis result.

EXPRESS specification:

```
*)
ENTITY analysis_results_set
SUPERTYPE OF (ONEOF
    (analysis_results_set_basic,
    analysis_results_set_combined,
    analysis_results_set_envelope,
    analysis_results_set_redistributed));
    results_set_name : label;
    component_results : SET [1:?] OF analysis_result;
END_ENTITY;
(*
```

Attribute definitions:

results_set_name

A short text reference used to identify the analysis_results_set.

component_results

Declares the set of one or more instances of analysis_result associated with the analysis_results_set. There must be at least one instance of analysis_result associated with each instance of analysis_results_set.

Notes:

Known as ANALYSIS_RESULTS_SET in CIS/1.

See diagram 35 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.13 analysis_results_set_basic

Entity definition:

A type of analysis_result_set where the analysis results are all referring to the response of the structure to an applied basic load case. This entity associates the set of analysis results with an appropriate (basic) load case. (See also definition of load_case.)

EXPRESS specification:

```

*)
ENTITY analysis_results_set_basic
SUBTYPE OF (analysis_results_set);
    basic_load_case : load_case;
END_ENTITY;
(*)

```

Attribute definitions:*basic_load_case*

Declares the instance of load_case associated with the analysis_results_set_basic. There must be one (and only one) instance of load_case associated an instance of analysis_results_set_basic.

Notes:

Known as BASIC_RESULTS in CIS/1.

See diagram 35 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.14 analysis_results_set_combined**Entity definition:**

A type of analysis_result_set where the analysis results are all referring to the response of the structure to a particular loading combination. This entity associates the set of analysis results with an appropriate loading_combination. (See also definitions of load_case and loading_combination.)

EXPRESS specification:

```

*)
ENTITY analysis_results_set_combined
SUBTYPE OF (analysis_results_set);
    loading_combination_ref : loading_combination;
END_ENTITY;
(*)

```

Attribute definitions:*loading_combination_ref*

Declares the instance of loading_combination associated with the analysis_results_set_basic. There must be one (and only one) instance of loading_combination associated an instance of analysis_results_set_combined.

Notes:

Known as COMBINED_RESULTS in CIS/1.

See diagram 35 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.15 analysis_results_set_envelope

Entity definition:

A type of analysis_result_set where the analysis results represent the response of the structure to a number of loading combinations such that the results are extreme (maximum or minimum) values. This entity associates the set of analysis results with an appropriate set of results for particular loading combinations. (See also definitions of load_case, loading_combination, and analysis_results_set_combined.)

EXPRESS specification:

```
*)
ENTITY analysis_results_set_envelope
SUBTYPE OF (analysis_results_set);
    max_or_min : maximum_or_minimum;
    component_combinations : SET [1:?] OF analysis_results_set_combined;
END_ENTITY;
(*
```

Attribute definitions:

max_or_min

Declares the classification of the set of analysis results as being either maximum or minimum numerical values.

component_combinations

Declares the set of sets analysis results for loading combinations that make up the envelope. Each instance of analysis_results_set_envelope must be associated with at least one instance of analysis_results_set_combined.

Notes:

Known as ENVELOPE_RESULTS in CIS/1.

See diagram 35 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.16 analysis_results_set_redistributed

Entity definition:

A type of analysis_result_set where the analysis results represent a redistributed response of the structure. This entity associates the set of analysis results with an appropriate list of factors. These factors would normally represent the numerical values of the ratios by which the analysis results have been modified. (See also definitions of load_case and loading_combination.)

EXPRESS specification:

```
*)
ENTITY analysis_results_set_redistributed
SUBTYPE OF (analysis_results_set);
    redistribution_factors : LIST [1:?] OF ratio_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:***redistribution_factors***

Declares the list instances of `ratio_measure_with_unit` which represent the numerical values of the ratios by which the analysis results have been modified. There must be at least one instance of `ratio_measure_with_unit` associated with each instance of `analysis_results_set_redistributed`.

Formal propositions:

It should be noted that because the aggregation data type is a LIST rather than a SET, repetitions are allowed, such that the `redistribution_factors` can be defined by several references to the same instance of `ratio_measure_with_unit`. Further, order is important in a LIST.

Informal propositions:

It is implied that the list of instances of `ratio_measure_with_unit` refer to the numerical values of the ratios by which each `analysis_result` in the set of `component_results` (inherited from the SUPERTYPE `analysis_results_set`), has been modified.

Notes:

New for CIS/2.

See diagram 35 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.17 design_result**Entity definition:**

This entity allows the results of a member or connection design process to be represented. As an abstract SUPERTYPE it must be instanced as one of its SUBTYPES, which categorize the result as being for a structural member, structural connection, design joint system, or design part. The SUBTYPE `design_result_mapped` allows a set of analysis results (`analysis_results_set`) to be associated with the design result. A design result must be given a name. (See also `analysis_results_set`.)

EXPRESS specification:

*)

ENTITY `design_result`

ABSTRACT SUPERTYPE OF (ONEOF

`design_result_connection`,
`design_result_joint_system`,
`design_result_member`,
`design_result_part`) ANDOR
`design_result_mapped` ANDOR
`design_result_resolved`);

`design_result_name` : label;

`design_resistance` : resistance;

END_ENTITY;

(*

Attribute definitions:*design_result_name*

A short text description that is used to reference the design_result.

design_resistance

Declares the resistance associated with the design result. There must be one (and only one) instance of resistance referenced by each instance of design_result. The instance of resistance provides the capacity of the structural component.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it can be instanced with several of its SUBTYPES. In such an event, the complex entity instance created is encoded in the STEP Part 21 file using external mapping.

Notes:

New for CIS/2.

See diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.18 design_result_connection**Entity definition:**

A type of design_result that creates the associated between an instance of assembly_design_structural_connection and an instance of resistance to provide the capacity of a structural connection. (See also the definitions for the entities resistance and assembly_design_structural_connection.)

EXPRESS specification:

*)

ENTITY design_result_connection

SUBTYPE OF (design_result);

 result_for_connection : assembly_design_structural_connection;

 result_position : OPTIONAL point;

 position_label : OPTIONAL label;

END_ENTITY;

(*

Attribute definitions:*result_for_connection*

Declares the structural connection associated with the design result for which the resistance is described. There must be one (and only one) instance of assembly_design_structural_connection referenced by each instance of design_result_connection.

result_position

Declares the instance of point that may be associated with the design_result. There may be one (and only one) instance of point referenced by each instance of design_result. The point will be defined in the local coordinate system of the structural member or structural connection under consideration.

position_label

An optional short text description that may be used to describe the position on the structural member or structural connection for which the design_result is provided.

Informal propositions:

As the INVERSE clause has not been declared in the entity assembly_design_structural_connection, the default cardinality [0:?] applies and there can be any number of design_results associated with a assembly_design_structural_connection. This implies that a structural connection can have any number of resistances. In this way, the design_result_connection acts as an intersection entity to resolve the many-to-many relationship between assembly_design_structural_connection and resistance.

Notes:

New for CIS/2.

See diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.19 design_result_joint_system**Entity definition:**

A type of design_result that creates the associated between an instance of design_joint_system and an instance of resistance to provide the capacity 'as designed' of a joint system. (See also the definitions for the entities resistance and design_joint_system.)

EXPRESS specification:

*)

```
ENTITY design_result_joint_system
  SUBTYPE OF (design_result);
    result_for_joint_system : design_joint_system;
END_ENTITY;
(*
```

Attribute definitions:*result_for_joint_system*

Declares the instance of design_joint_system associated with the design_result_joint_system. There must be one (and only one) instance of design_joint_system referenced by each instance of design_result_joint_system.

Informal propositions:

As the INVERSE clause has not been declared in the entity design_result_joint_system, the default cardinality [0:?] applies and there can be any number of design_results associated with a design_joint_system. This implies that a joint system can have any number of resistances. In this way, the design_result_joint_system acts as an intersection

entity to resolve the many-to-many relationship between `design_joint_system` and resistance.

A position is not given for the design result. It is assumed that the instance of `design_result_joint_system` provides the design capacity of the whole joint system.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.20 design_result_mapped

Entity definition:

A type of `design_result` that is associated with a set of analysis results (through the entity `analysis_results_set`). This entity would normally be instantiated with one of its sibling SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY design_result_mapped
  SUBTYPE OF (design_result);
    origin_of_forces : analysis_results_set;
END_ENTITY;
(*
```

Attribute definitions:

origin_of_forces

Declares the instance of `analysis_results_set` associated with the `design_result_mapped`. This may be used to provide the set of forces that have been used in the derivation of the design forces for which the structural component has been designed. This also allows the mapping back from the design model to the analysis model.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.21 design_result_member

Entity definition:

A type of `design_result` where the set of analysis results is associated with a structural member and a set of resistances. (See also `assembly_design_structural_member` and resistance.)

EXPRESS specification:

```
*)
ENTITY design_result_member
  SUBTYPE OF (design_result);
    result_for_member : assembly_design_structural_member;
    result_position : point;
```

```

    position_label : label;
END_ENTITY;
(*)

```

Attribute definitions:

result_for_member

Declares the structural member associated with the design result for which the resistance is described. There must be one (and only one) instance of assembly_design_structural_member referenced by each instance of design_result_member.

result_position

Declares the instance of point that may be associated with the design_result. There may be one (and only one) instance of point referenced by each instance of design_result. The point will be defined in the local coordinate system of the structural member under consideration.

position_label

An optional short text description that may be used to describe the position on the structural member for which the design_result is provided.

Informal propositions:

As the INVERSE clause has not been declared in the entity assembly_design_structural_member, the default cardinality [0:?] applies and there can be any number of design_results associated with a assembly_design_structural_member. This implies that a structural connection can have any number of resistances.

Notes:

New for CIS/2.

See diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.22 design_result_part

Entity definition:

A type of design_result that creates the associated between an instance of design_part and an instance of resistance to provide the ‘as designed’ capacity of a part. (See also the definitions for the entities resistance and design_part.)

EXPRESS specification:

```

*)
ENTITY design_result_part
SUBTYPE OF (design_result);
    result_for_part : design_part;
END_ENTITY;
(*)

```

Attribute definitions:**result_for_part**

Declares the instance of design_part associated with the design_result_part. There must be one (and only one) instance of design_part referenced by each instance of design_result_part.

Informal propositions:

As the INVERSE clause has not been declared in the entity design_result_part, the default cardinality [0:?] applies and there can be any number of design_results associated with a design_part. This implies that a joint system can have any number of resistances. In this way, the design_result_part acts as an intersection entity to resolve the many-to-many relationship between design_joint_system and resistance.

A position is not given for the design result. It is assumed that the instance of design_result_part provides the design capacity of the whole part.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.23 design_result_resolved**Entity definition:**

A type of design_result that is associated with a set of forces (through the entity reaction_force). This entity would normally be instanced with one of its sibling SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY design_result_resolved
SUBTYPE OF (design_result);
    design_forces : reaction_force;
    design_factor : REAL;
END_ENTITY;
(*
```

Attribute definitions:**design_forces**

Declares the instance of reaction_force associated with the design_result. There must be one (and only one) instance of reaction_force referenced by each instance of design_result_resolved. This may be used to provide the design forces (a set of 3 forces and 3 moments) that have derived from the results of the analysis and have been used in the design of the structural component. This allows the mapping from the analysis model to the design model to be simplified and resolved.

design_factor

The numerical value of the multiplier that is globally applied to all the attribute values of the instance of reaction_force associated with the design_result_resolved.

When using limit state design, this factor would normally be set to 1.0 if the reaction_force represented the resolved results of an analysis for a factored loading combination at ultimate limit state. If using the results of an unfactored loading combination, this attribute would provide a value that would allow for the equivalent to all the partial factors of safety for loading multiplied together.

Informal propositions:

The ANDOR SUPERTYPE construct allows this entity to be instanced with one of its sibling SUBTYPES. For example, when instanced with the entity design_result_member, the complex instance represents the as designed capacity for a structural member subjected to the resolved design forces. This instance would be encoded in a STEP Part 21 file using external mapping.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.24 reaction

Entity definition:

An abstract SUPERTYPE that is used to define particular sets of values of the response of the structure (in terms of forces, moments, displacements, etc.) to the applied loading. Reactions can be seen as the analysed response of the analysis model to physical actions imposed on it (defined in the loading model). This entity must be instanced as one of its SUBTYPES, which categorize the reaction in terms of force, displacement, velocity, or acceleration. The remaining SUBTYPES allow the dynamic response of a structure and the equilibrium checks on the analysis model to be represented.

EXPRESS specification:

```
*)
ENTITY reaction
ABSTRACT SUPERTYPE OF (ONEOF
    (reaction_force,
     reaction_displacement,
     reaction_velocity,
     reaction_acceleration,
     reaction_dynamic,
     reaction_equilibrium));
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. The purpose of this entity is to collect together the SUBTYPES.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2.

See diagram 33 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.25 reaction_acceleration**Entity definition:**

A type of reaction that is described in terms of 3 linear accelerations and 3 rotational accelerations. This entity makes 3 references to linear_acceleration_measure_with_unit and 3 references to rotational_acceleration_measure_with_unit, which provide the numerical values with appropriate units of the response acceleration (or rate of change of velocity). (Any warping effects are subsumed in the torsion effects and represented by the appropriate moment.)

EXPRESS specification:

*)

ENTITY reaction_acceleration

SUBTYPE OF (reaction);

reaction_acceleration_ax : OPTIONAL linear_acceleration_measure_with_unit;
 reaction_acceleration_ay : OPTIONAL linear_acceleration_measure_with_unit;
 reaction_acceleration_az : OPTIONAL linear_acceleration_measure_with_unit;
 reaction_acceleration_arx : OPTIONAL rotational_acceleration_measure_with_unit;
 reaction_acceleration_ary : OPTIONAL rotational_acceleration_measure_with_unit;
 reaction_acceleration_arz : OPTIONAL rotational_acceleration_measure_with_unit;

WHERE

WRR8 : EXISTS (reaction_acceleration_ax) OR
 EXISTS (reaction_acceleration_ay) OR
 EXISTS (reaction_acceleration_az) OR
 EXISTS (reaction_acceleration_arx) OR
 EXISTS (reaction_acceleration_ary) OR
 EXISTS (reaction_acceleration_arz);

END_ENTITY;

(*

Attribute definitions:**reaction_acceleration_ax**

Declares the first instance of linear_acceleration_measure_with_unit associated with the reaction_acceleration, which is used to provide the numerical value with an appropriate unit of the response acceleration in the linear x-direction.

reaction_acceleration_ay

Declares the second instance of linear_acceleration_measure_with_unit associated with the reaction_acceleration, which provides the numerical value with an appropriate unit of the response acceleration in the linear y-direction.

reaction_acceleration_az

Declares the third instance of linear_acceleration_measure_with_unit associated with the reaction_acceleration, which provides the numerical value with an appropriate unit of the response acceleration in the linear z-direction.

reaction_acceleration_arx

Declares the first instance of *rotational_acceleration_measure_with_unit* associated with the *reaction_acceleration*, which provides the numerical value with an appropriate unit of the response acceleration about the x-axis.

reaction_acceleration_ary

Declares the second instance of *rotational_acceleration_measure_with_unit* associated with the *reaction_acceleration*, which provides the numerical value with an appropriate unit of the response acceleration about the y-axis.

reaction_acceleration_arz

Declares the third instance of *rotational_acceleration_measure_with_unit* associated with the *reaction_acceleration*, which provides the numerical value with an appropriate unit of the response acceleration about the z-axis.

Formal propositions:*WRR8*

At least one of the attributes *reaction_acceleration_ax*, *reaction_acceleration_ay*, *reaction_acceleration_az*, *reaction_acceleration_arx*, *reaction_acceleration_ary*, or *reaction_acceleration_arz* must be assigned a value. In other words, an instance of *reaction_acceleration* cannot exist without one of its attributes being populated.

For 3D analysis models, it is expected that all of the attributes of this entity will be given a value, even if that value is zero. Where the value is unknown or not applicable, as will be the case for a 2D analysis model, the attribute should be given a NULL value.

Notes:

New for CIS/2.

See diagram 34 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.26 reaction_displacement**Entity definition:**

A type of reaction that is described in terms of 3 lengths and 3 angles. This entity makes 3 references to *length_measure_with_unit* and 3 references to *plane_angle_measure_with_unit*, which provide the numerical values with appropriate units of the response displacement. (Any warping effects are subsumed in the torsion effects and represented by the appropriate displacement.)

EXPRESS specification:

*)

ENTITY *reaction_displacement*

SUBTYPE OF (*reaction*);

reaction_displacement_dx : OPTIONAL *length_measure_with_unit*;

reaction_displacement_dy : OPTIONAL *length_measure_with_unit*;

reaction_displacement_dz : OPTIONAL *length_measure_with_unit*;

reaction_displacement_rx : OPTIONAL *plane_angle_measure_with_unit*;

reaction_displacement_ry : OPTIONAL *plane_angle_measure_with_unit*;

reaction_displacement_rz : OPTIONAL *plane_angle_measure_with_unit*;

WHERE

```

WRR9 : EXISTS (reaction_displacement_dx) OR
        EXISTS (reaction_displacement_dy) OR
        EXISTS (reaction_displacement_dz) OR
        EXISTS (reaction_displacement_rx) OR
        EXISTS (reaction_displacement_ry) OR
        EXISTS (reaction_displacement_rz);
END_ENTITY;
(*)

```

Attribute definitions:

reaction_displacement_dx

Declares the first instance of `length_measure_with_unit` associated with the `reaction_displacement`, which is used to provide the numerical value with an appropriate unit of the response displacement in the linear x-direction.

reaction_displacement_dy

Declares the second instance of `length_measure_with_unit` associated with the `reaction_displacement`, which is used to provide the numerical value with an appropriate unit of the response displacement in the linear y-direction.

reaction_displacement_dz

Declares the third instance of `length_measure_with_unit` associated with the `reaction_displacement`, which is used to provide the numerical value with an appropriate unit of the response displacement in the linear z-direction.

reaction_displacement_rx

Declares the first instance of `plane_angle_measure_with_unit` associated with the `reaction_displacement`, which is used to provide the numerical value with an appropriate unit of the response rotation about the x-axis.

reaction_displacement_ry

Declares the second instance of `plane_angle_measure_with_unit` associated with the `reaction_displacement`, which is used to provide the numerical value with an appropriate unit of the response rotation about the y-axis.

reaction_displacement_rz

Declares the third instance of `plane_angle_measure_with_unit` associated with the `reaction_displacement`, which is used to provide the numerical value with an appropriate unit of the response rotation about the z-axis.

Formal propositions:

WRR9

At least one of the attributes `reaction_displacement_dx`, `reaction_displacement_dy`, `reaction_displacement_dz`, `reaction_displacement_rx`, `reaction_displacement_ry`, or `reaction_displacement_rz` must be assigned a value. In other words, an instance of `reaction_displacement` cannot exist without one of its attributes being populated.

For 3D analysis models, it is expected that all of the attributes of this entity will be given a value, even if that value is zero. Where the value is unknown or not applicable, as will be the case for a 2D analysis model, the attribute should be given a NULL value.

Notes:

New for CIS/2.

See diagram 33 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.27 reaction_dynamic

Entity definition:

A type of reaction that is described in terms of a angle, a length and a frequency. This entity makes 1 reference to length_measure_with_unit, 1 reference to plane_angle_measure_with_unit, and 1 reference to frequency_measure_with_unit, which provide the numerical values with appropriate units of the dynamic response of the structure to the applied loading.

EXPRESS specification:

*)

ENTITY reaction_dynamic

SUBTYPE OF (reaction);

phase_angle : OPTIONAL plane_angle_measure_with_unit;

response_amplitude : OPTIONAL length_measure_with_unit;

natural_frequency : OPTIONAL frequency_measure_with_unit;

WHERE

WRR10 : EXISTS (phase_angle) OR
EXISTS (response_amplitude) OR
EXISTS (natural_frequency);

END_ENTITY;

(*

Attribute definitions:

phase_angle

Declares the instance of plane_angle_measure_with_unit associated with the reaction_dynamic, which is used to provide the numerical value with an appropriate unit of the phase angle of the dynamic response.

response_amplitude

Declares the instance of length_measure_with_unit associated with the reaction_dynamic, which is used to provide the numerical value with an appropriate unit of the amplitude of the dynamic response.

natural_frequency

Declares the instance of frequency_measure_with_unit associated with the reaction_dynamic, which is used to provide the numerical value with an appropriate unit of the natural frequency of the dynamic response.

Formal propositions:

WRR10

At least one of the attributes phase_angle, response_amplitude, or natural_frequency must be assigned a value. In other words, an instance of reaction_dynamic cannot exist without one of its attributes being populated. Where the value is unknown or not applicable, the attribute should be given a NULL value.

Notes:

New for CIS/2.

See diagram 34 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.28 reaction_equilibrium

Entity definition:

A type of reaction where the equilibrium of the structure (as represented by the analysis model) is checked. This entity makes 6 references to ratio_measure_with_unit, which provide the numerical values with an appropriate unit of the equilibrium in the 3 linear directions and about 3 rotational axes. For any point in equilibrium the sum total of forces should theoretically be zero. Thus, the values of the attributes will be close to zero.

EXPRESS specification:

*)

ENTITY reaction_equilibrium

SUBTYPE OF (reaction);

equilibrium_dx : OPTIONAL ratio_measure_with_unit;

equilibrium_dy : OPTIONAL ratio_measure_with_unit;

equilibrium_dz : OPTIONAL ratio_measure_with_unit;

equilibrium_mx : OPTIONAL ratio_measure_with_unit;

equilibrium_my : OPTIONAL ratio_measure_with_unit;

equilibrium_mz : OPTIONAL ratio_measure_with_unit;

WHERE

WRR11 : EXISTS (equilibrium_dx) OR
EXISTS (equilibrium_dy) OR
EXISTS (equilibrium_dz);

WRR12 : EXISTS (equilibrium_mx) OR
EXISTS (equilibrium_my) OR
EXISTS (equilibrium_mz);

END_ENTITY;

(*

Attribute definitions:

equilibrium_dx

Declares the first instance of ratio_measure_with_unit associated with the reaction_equilibrium, which is used to provide the numerical value with an appropriate unit of the equilibrium of force components in the linear x-direction.

equilibrium_dy

Declares the second instance of ratio_measure_with_unit associated with the reaction_equilibrium, which is used to provide the numerical value with an appropriate unit of the equilibrium of force components in the linear y-direction.

equilibrium_dz

Declares the third instance of ratio_measure_with_unit associated with the reaction_equilibrium, which is used to provide the numerical value with an appropriate unit of the equilibrium of force components in the linear z-direction.

equilibrium_mx

Declares the fourth instance of `ratio_measure_with_unit` associated with the `reaction_equilibrium`, which is used to provide the numerical value with an appropriate unit of the equilibrium of force components about the x-axis.

equilibrium_my

Declares the fifth instance of `ratio_measure_with_unit` associated with the `reaction_equilibrium`, which is used to provide the numerical value with an appropriate unit of the equilibrium of force components about the y-axis.

equilibrium_mz

Declares the sixth instance of `ratio_measure_with_unit` associated with the `reaction_equilibrium`, which is used to provide the numerical value with an appropriate unit of the equilibrium of force components about the z-axis.

Formal propositions:*WRR11*

At least one of the attributes `equilibrium_dx`, `equilibrium_dy`, or `equilibrium_dz` must be assigned a value. In other words, an instance of `reaction_equilibrium` cannot exist without a value being provided for the equilibrium of force components in a linear direction.

For 3D analysis models, it is expected that all of the attributes of this entity will be given a value. Where the value is unknown or not applicable, as will be the case for a 2D analysis model, the attribute should be given a NULL value.

WRR12

At least one of the attributes `equilibrium_mx`, `equilibrium_my`, or `equilibrium_mz` must be assigned a value. In other words, an instance of `reaction_equilibrium` cannot exist without a value being provided for the equilibrium of force components about one axis.

For 3D analysis models, it is expected that all of the attributes of this entity will be given a value. Where the value is unknown or not applicable, as will be the case for a 2D analysis model, the attribute should be given a NULL value.

Notes:

New for CIS/2.

See diagram 33 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.29 reaction_force**Entity definition:**

A type of reaction that is described in terms of 3 forces and 3 moments. This entity makes 3 references to `force_measure_with_unit` and 3 references to `moment_measure_with_unit`, which provide the numerical values with appropriate units of the response forces and moments. (Any warping effects are subsumed in the torsion effects and represented by the appropriate moment.)

EXPRESS specification:

```

*)
ENTITY reaction_force
SUBTYPE OF (reaction);
    reaction_force_fx : OPTIONAL force_measure_with_unit;
    reaction_force_fy : OPTIONAL force_measure_with_unit;
    reaction_force_fz : OPTIONAL force_measure_with_unit;
    reaction_force_mx : OPTIONAL moment_measure_with_unit;
    reaction_force_my : OPTIONAL moment_measure_with_unit;
    reaction_force_mz : OPTIONAL moment_measure_with_unit;
WHERE
    WRR13 : EXISTS (reaction_force_fx) OR
            EXISTS (reaction_force_fy) OR
            EXISTS (reaction_force_fz) OR
            EXISTS (reaction_force_mx) OR
            EXISTS (reaction_force_my) OR
            EXISTS (reaction_force_mz);
END_ENTITY;
(*)

```

Attribute definitions:*reaction_force_fx*

Declares the first instance of *force_measure_with_unit* associated with this *reaction_force*, which is used to provide the numerical value with an appropriate unit of the response force in the linear x-direction.

reaction_force_fy

Declares the second instance of *force_measure_with_unit* associated with this *reaction_force*, which is used to provide the numerical value with an appropriate unit of the response force in the linear y-direction.

reaction_force_fz

Declares the third instance of *force_measure_with_unit* associated with this *reaction_force*, which is used to provide the numerical value with an appropriate unit of the response force in the linear z-direction.

reaction_force_mx

Declares the first instance of *moment_measure_with_unit* associated with this *reaction_force*, which is used to provide the numerical value with an appropriate unit of the response moment about the x-axis.

reaction_force_my

Declares the second instance of *moment_measure_with_unit* associated with this *reaction_force*, which is used to provide the numerical value with an appropriate unit of the response moment about the y-axis.

reaction_force_mz

Declares the third instance of *moment_measure_with_unit* associated with this *reaction_force*, which is used to provide the numerical value with an appropriate unit of the response moment about the z-axis.

Formal propositions:**WRR13**

At least one of the attributes `reaction_force_fx`, `reaction_force_fy`, `reaction_force_fz`, `reaction_force_mx`, `reaction_force_my`, or `reaction_force_mz` must be assigned a value. In other words, an instance of `reaction_force` cannot exist without one of its attributes being populated.

For 3D analysis models, it is expected that all of the attributes of this entity will be given a value, even if that value is zero. Where the value is unknown or not applicable, as will be the case for a 2D analysis model, the attribute should be given a NULL value.

Notes:

New for CIS/2.

See diagram 33 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.30 reaction_velocity**Entity definition:**

A type of reaction that is described in terms of 3 linear velocities and 3 rotational velocities. This entity makes 3 references to `linear_velocity_measure_with_unit` and 3 references to `rotational_velocity_measure_with_unit`, which provide the numerical values with appropriate units of the response velocity (or rate of change of displacement). (Any warping effects are subsumed in the torsion effects and represented by the appropriate velocity.)

EXPRESS specification:

*)

ENTITY `reaction_velocity`

SUBTYPE OF (`reaction`);

`reaction_velocity_vx` : OPTIONAL `linear_velocity_measure_with_unit`;

`reaction_velocity_vy` : OPTIONAL `linear_velocity_measure_with_unit`;

`reaction_velocity_vz` : OPTIONAL `linear_velocity_measure_with_unit`;

`reaction_velocity_vrx` : OPTIONAL `rotational_velocity_measure_with_unit`;

`reaction_velocity_vry` : OPTIONAL `rotational_velocity_measure_with_unit`;

`reaction_velocity_vrz` : OPTIONAL `rotational_velocity_measure_with_unit`;

WHERE

WRR14 : EXISTS (`reaction_velocity_vx`) OR

EXISTS (`reaction_velocity_vy`) OR

EXISTS (`reaction_velocity_vz`) OR

EXISTS (`reaction_velocity_vrx`) OR

EXISTS (`reaction_velocity_vry`) OR

EXISTS (`reaction_velocity_vrz`);

END_ENTITY;

(*

Attribute definitions:**reaction_velocity_vx**

Declares the first instance of `linear_velocity_measure_with_unit` associated with this `reaction_velocity`, which is used to provide the numerical value with an appropriate unit of the response velocity in the linear x-direction.

reaction_velocity_vy

Declares the second instance of *linear_velocity_measure_with_unit* associated with this *reaction_velocity*, which is used to provide the numerical value with an appropriate unit of the response velocity in the linear y-direction.

reaction_velocity_vz

Declares the third instance of *linear_velocity_measure_with_unit* associated with this *reaction_velocity*, which is used to provide the numerical value with an appropriate unit of the response velocity in the linear z-direction.

reaction_velocity_vrx

Declares the first instance of *rotational_velocity_measure_with_unit* associated with this *reaction_velocity*, which is used to provide the numerical value with an appropriate unit of the response velocity about the x-axis.

reaction_velocity_vry

Declares the second instance of *rotational_velocity_measure_with_unit* associated with this *reaction_velocity*, which is used to provide the numerical value with an appropriate unit of the response velocity about the y-axis.

reaction_velocity_vrz

Declares the third instance of *rotational_velocity_measure_with_unit* associated with this *reaction_velocity*, which is used to provide the numerical value with an appropriate unit of the response velocity about the z-axis.

Formal propositions:*WRR14*

At least one of the attributes *reaction_velocity_vx*, *reaction_velocity_vy*, *reaction_velocity_vz*, *reaction_velocity_vrx*, *reaction_velocity_vry*, or *reaction_velocity_vrz* must be assigned a value. In other words, an instance of *reaction_velocity* cannot exist without one of its attributes being populated.

For 3D analysis models, it is expected that all of the attributes of this entity will be given a value, even if that value is zero. Where the value is unknown or not applicable, as will be the case for a 2D analysis model, the attribute should be given a NULL value.

Notes:

New for CIS/2.

See diagram 34 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.31 resistance**Entity definition:**

This entity allows the capacity of the structural component to be represented. The entity can represent either the elastic resistance or the plastic resistance. This entity is used with the entity *design_result* to represent the ‘as required’ or ‘as provided’ resistance (or capacity) of a structural component. (See also the definition for the entity *design_result* and its SUBTYPES). This entity must be instanced with one of its SUBTYPES. These SUBTYPES provide the bending, shear, and axial capacity of a component to be represented through their references to *moment_measure_with_unit* and *force_measure_with_unit*.

EXPRESS specification:

```

*)
ENTITY resistance
ABSTRACT SUPERTYPE OF (ONEOF(
    resistance_bending,
    resistance_shear,
    resistance_axial));
    resistance_type : label;
    resistance_description : OPTIONAL text;
    resistance_factor : REAL;
    elastic_or_plastic : elastic_or_plastic_resistance;
    local_or_global : global_or_local_resistance;
INVERSE
    results : SET [1:?] OF design_result FOR design_resistance;
END_ENTITY;
(*

```

Attribute definitions:*resistance_type*

An optional short text reference that may be used to describe the type of resistance being specified; e.g. ‘as required resistance’, ‘as provided capacity’.

resistance_description

An optional text description that may be used to provide more detailed information about the required resistance or provided capacity of the component under consideration. The governing criteria may also be specified here.

resistance_factor

The numerical value of the common multiplier (or factor) that is globally applied to all the attribute values of the instances of *moment_measure_with_unit* and *force_measure_with_unit* associated with the resistance.

When using limit state design, this factor would normally be set to 1.0 if the instances of *moment_measure_with_unit* and *force_measure_with_unit* represent the resolved capacity of the component at ultimate limit state. If those instances represent the unfactored capacity of the component, this attribute would provide a value that would allow for the equivalent to all the partial factors of safety for materials multiplied together.

elastic_or_plastic

Declares the classification of the type of the resistance as either elastic or plastic. If the resistance is specified as being ‘elastic’, the instance of resistance is defining the capacity of the structural component in an elastic state; i.e. one in which the material has not reached yield and Hook’s Law still applies. If the resistance is specified as being ‘plastic’, the instance of resistance is defining the capacity of the structural component in a plastic state; i.e. one in which the material has been taken beyond its yield point and Hook’s Law is no longer applicable.

local_or_global

Declares the classification of the resistance as either global or local resistance. If the resistance is specified as being ‘global’, the instance of resistance is defining the overall capacity of the structural component. If it is specified as being ‘local’, the instance of

resistance is defining the capacity of the structural component at a particular position on that component.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

results

The resistance must be associated with at least one instance of design_result. That is, the resistance cannot exist without being referenced by at least one design_result. The upper limit is unbounded. This allows an instance of resistance to be referenced by any number of instances of design_results, and thereby provide the capacity for any number of structural components.

Notes:

New for CIS/2.

See diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.32 resistance_axial

Entity definition:

A type of resistance that provides the axial capacity of a structural component through two references to force_measure_with_unit, which provide the tensile and compressive resistance.

EXPRESS specification:

*)

ENTITY resistance_axial

SUBTYPE OF (resistance);

tensile_resistance_ptx : OPTIONAL force_measure_with_unit;

compressive_resistance_pcx : OPTIONAL force_measure_with_unit;

WHERE

WRR30 : EXISTS(tensile_resistance_ptx) OR EXISTS(compressive_resistance_pcx);

END_ENTITY;

(*

Attribute definitions:

tensile_resistance_ptx

Declares the first instance of force_measure_with_unit associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist tension.

compressive_resistance_pcx

Declares the second instance of force_measure_with_unit associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist compression.

Formal propositions:**WRR30**

One or other of the attributes `tensile_resistance_ptx` or `compressive_resistance_pcx` must be provided with a value. In other words, an `resistance_axial` cannot exist without either the tensile or the compressive resistance being specified.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.33 resistance_bending**Entity definition:**

A type of resistance that provides the bending capacity of a structural component through four references to `moment_measure_with_unit`, which provide the values for component's resistance to bending, bucking and torsion.

EXPRESS specification:

*)

ENTITY `resistance_bending`

SUBTYPE OF (`resistance`);

`torsional_resistance_mx` : OPTIONAL `moment_measure_with_unit`;

`bending_resistance_my` : OPTIONAL `moment_measure_with_unit`;

`bending_resistance_mz` : OPTIONAL `moment_measure_with_unit`;

`buckling_resistance_mb` : OPTIONAL `moment_measure_with_unit`;

WHERE

`WRR28` : EXISTS(`torsional_resistance_mx`) OR EXISTS(`bending_resistance_my`) OR

EXISTS(`bending_resistance_mz`) OR

EXISTS(`buckling_resistance_mb`);

END_ENTITY;

(*

Attribute definitions:***torsional_resistance_mx***

Declares the first instance of `moment_measure_with_unit` associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist torsional moment about the x axis.

bending_resistance_my

Declares the second instance of `moment_measure_with_unit` associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist flexural bending moment about the y axis.

bending_resistance_mz

Declares the third instance of `moment_measure_with_unit` associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist flexural bending moment about the z axis.

buckling_resistance_mb

Declares the fourth instance of *moment_measure_with_unit* associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist buckling due to flexural bending.

Formal propositions:*WRR28*

At least one of the attributes *torsional_resistance_mx*, *bending_resistance_my*, *bending_resistance_mz*, or *buckling_resistance_mb* must be populated. In other words, a *resistance_bending* cannot exist without a value being specified for the component's resistance to torsion, major axis bending, minor axis bending, or buckling.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

7.3.34 resistance_shear**Entity definition:**

A type of resistance that provides the shear capacity of a structural component through two references to *force_measure_with_unit*, which provide the normal and buckling shear resistance.

EXPRESS specification:

*)

ENTITY *resistance_shear*

SUBTYPE OF (*resistance*);

normal_shear_resistance_pv : OPTIONAL *force_measure_with_unit*;

buckling_shear_resistance_pbv : OPTIONAL *force_measure_with_unit*;

WHERE

 WRR29 : EXISTS(*normal_shear_resistance_pv*) OR
 EXISTS(*buckling_shear_resistance_pbv*);

END_ENTITY;

(*

Attribute definitions:*normal_shear_resistance_pv*

Declares the first instance of *force_measure_with_unit* associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist shear force.

buckling_shear_resistance_pbv

Declares the second instance of *force_measure_with_unit* associated with the resistance, which provides the numerical value with an appropriate unit of the capacity of the structural component to resist shear buckling.

Formal propositions:

WRR29

One or other of the attributes `normal_shear_resistance_pv` or `buckling_shear_resistance_pbv` must be provided with a value. In other words, an `resistance_shear` cannot exist without either the normal or the buckling shear resistance being specified.

Notes

New for CIS/2.

See Diagram 36 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8 LPM/6 PARTS

8.1 Parts concepts and assumptions

The Parts subject area covers the description and specification of generic and specific parts, which are then used within design parts and located parts. That is, parts can be described at the specific level, and then implemented many times at the occurrence level. (This can be thought of as the ‘type’ verses ‘token’ approach seen in many database applications; with ‘types’ residing at the specific level and ‘tokens’ residing at the occurrence level.) Thus, a located part is defined as a located and oriented implementation of the corresponding (specific) part. A part is an ‘atomic’ component of the structure; if all the joint systems were removed from the structure, then each of the pieces that remain would be a located part.

Parts are categorized by their dimensionality and the complexity of their shape definition, e.g. prismatic, sheet or complex. LPM/6 supports a comprehensive range of section profiles and their associated section properties described in the ‘Geometry’ subject area.

SUPERTYPE-SUBTYPE hierarchies in LPM/6 categorize objects by their geometry. In order to denote the use of implicit or explicit geometry, LPM/6 has adopted the convention whereby objects are categorized as either simple or complex. Diagrams 62 and 63 of the EXPRESS-G diagrams in Appendix B show examples of how this mechanism is used to define parts in LPM/6. It can be seen that the `part_prismatic_simple` is defined using instances of `positive_length_measure_with_unit`, while the `part_prismatic_complex` refers to an explicit curve (a SUBTYPE of a `geometric_representation_item`). Similarly, a `part_sheet_bounded_simple` is defined using four instances of `positive_length_measure_with_unit`, while the `part_sheet_bounded_complex` requires an explicit `bounded_surface` to be defined.

CIS/2 incorporates all of the constructs for explicit geometry defined in STEP Part 42^[24]. To define the shape of an complex object in LPM/6 using explicit geometry, the STEP entity `shape_representation` is SUBTYPED as `shape_representation_with_units`. This is then used as the data type of the attribute for that entity.

(See also the Geometry subject area.)

8.2 Parts type definitions

No types have been modified or added to this subject area for the 2nd Edition.

8.2.1 `castellation_type`

Type definition:

Classifies the longitudinal repeating feature by its basic shape; i.e. circular, hexagonal, or octagonal. See `part_prismatic_simple_castellated` for its use.

EXPRESS specification:

```
*)
TYPE castellation_type
= ENUMERATION OF
    (circular, hexagonal, octagonal, undefined);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

8.2.2 fabrication_type**Type definition:**

Classifies a part by the primary method of fabrication used in its manufacturing; i.e. rolled, welded, cold formed, cast, forged, or extruded. The method of fabrication (i.e. how the part was made) may be undefined.

EXPRESS specification:

*)

TYPE fabrication_type

= ENUMERATION OF

(rolled, welded, cold_formed, cast, forged, extruded, undefined);

END_TYPE;

(*

Notes:

Known as fab_type in CIS/1. Unchanged for 2nd Edition.

8.3 Parts entity definitions

The following entity have been modified for the 2nd Edition of CIS/2:

- part
- part_prismatic_simple_campered_absolute
- part_prismatic_simple_campered_relative

8.3.1 part**Entity definition:**

A specific level entity which defines the basic geometry of (one or more) corresponding located parts or design parts. A part has its own 3D coordinate system. It provides a self-contained description of an ‘atomic’ component which, when instanced at the occurrence level, may be modified by the presence of located features.

The part may be categorized (through its SUBTYPES) by its geometry as either being prismatic, sheet, or complex. It may also be categorized as a derived part.

For a part to have a specified material it must be instanced with its sibling SUBTYPE structural_frame_product_with_material. It should be noted that a part can only be associated with (at most) one material. Although a material can have a number of constituents, those constituents are intended to represent the materials that form a steel alloy, rather than a steel-concrete composite. Thus, a composite beam (which has separate steel and concrete components) is **not** modelled in CIS/2 as a single part but as an assembly of (at least) two parts.

As a SUBTYPE of structural_frame_product, an part inherits the three attributes item_number, item_name, and item_description (from structural_frame_item). It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a part may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the part to be described in detail
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A part may also be related to another part (or any other structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

*)

ENTITY part

SUPERTYPE OF (ONEOF

(part_prismatic,
part_sheet,
part_complex) ANDOR
part_derived)

SUBTYPE OF (structural_frame_product);

fabrication_method : fabrication_type;

manufacturers_ref : OPTIONAL text;

DERIVE

part_number : INTEGER := SELF\structural_frame_item.item_number;

part_name : label := SELF\structural_frame_item.item_name;

design_uses : SET [0:?] OF design_part := bag_to_set

(USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
DESIGN_PART.DESIGN_PART_SPEC'));

physical_uses : SET [0:?] OF located_part := bag_to_set

(USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
LOCATED_PART.DESRIPTIVE_PART'));

UNIQUE

URP6 : part_number, part_name;

END_ENTITY;

(*

Attribute definitions:

fabrication_method

Declares the classification of the part's primary method of fabrication; i.e. rolled, welded, cold formed, cast, forged, or extruded. The method of fabrication (i.e. how the part was made) can be undefined.

manufacturers_ref

An optional text reference that may be associated with the part to assist its identification during the manufacturing process. This could be, for example, a reference of where the explicit description of the particular manufactured part can be found (e.g. a

manufacturer's catalogue, file name, library name and reference). Specialist parts may be produced by only one or two manufacturers.

part_number

Derives the number for the part based on the item_number it inherits from the SUPERTYPE structural_frame_item. Used as a component of the unique reference for the part.

part_name

Derives the name for the part based on the item_name it inherits from the SUPERTYPE structural_frame_item. Used as a component of the unique reference for the part.

design_uses

Derives the set of zero, one or more instances of design_part that reference this instance of part in the specification of a design model.

physical_uses

Derives the set of zero, one or more instances of located_part that reference this instance of part in the specification of a physical model.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPEs. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

URP6

The combination of the part_number and the part_name shall be unique to this instance of part.

Informal propositions:

As a SUBTYPE of structural_frame_product, a part inherits the attributes item_number, item_name, and item_description (from structural_frame_item) and life_cycle_stage.

The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity structural_frame_product and bring together the SUBTYPEs under a common categorization.

Notes:

Known as S_PART in CIS/1.

This entity can be thought of as a use of the 'type' verses 'token' approach seen in many database applications; with the part entity representing a 'type' of part and the design_part and located_part entities representing 'tokens' of that part.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – Derive and Unique clauses added

8.3.2 part_complex

Entity definition:

A type of part whose geometry is defined explicitly using the geometric constructs of STEP Part 42. (See also the Geometry subject area.)

The complexity of the part is due to its shape rather than its material composition.

EXPRESS specification:

```
*)
ENTITY part_complex
SUBTYPE OF (part);
    part_shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:

part_shape

Declares the instance of `shape_representation_with_units` associated with the `part_complex`, which defines the shape of the complex part using a number of constructs from STEP Part 42. There must be one (and only one) instance of `shape_representation_with_units` referenced by each instance of `part_complex`. (See also the Geometry subject area.)

Notes:

New for CIS/2.

This entity requires the use of several constructs from STEP Parts 42 and 43 to complete its definition.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.3 part_derived

Entity definition:

A type of part that has been (or is going to be) derived from (or obtained / extracted from) another part. The manufacturing processes that have been (or will be) used to derive the part may be specified using the entity's relationship with `structural_frame_process`. For example, two haunches used in a portal frame may be cut from a single beam section.

It should be noted that the derived part is derived from only one part. This entity is not to be used to describe the fabrication of parts such as welded plate girders that are built up from hot rolled plates; these would be assemblies represented using the `assembly_manufacturing` entity. Further, A composite beam is **not** modelled as a `part_derived`.

EXPRESS specification:

```
*)
ENTITY part_derived
SUBTYPE OF (part);
    made_from : part;
    involved_processes : SET [0:?] OF structural_frame_process;
WHERE
    WRP2 : made_from :<>: (SELF);
END_ENTITY;
(*
```


Attribute definitions:*made_from*

Declares the instance of part associated with the part_derived. There must be one (and only one) instance of part referenced by each instance of part_derived.

involved_processes

Declares the set of instances of structural_frame_process associated with the part_derived, which specify the manufacturing processes used to derive the derived_part from the part. As the cardinality of the relationship is unbounded, there may be zero, one or many instances of structural_frame_process referenced by each instance of part_derived.

Formal propositions:*WRP2*

The made_from attribute must reference a separate and distinct instance of derived part. That is, a derived_part cannot be derived from itself. A derived_part can, however, be derived from another derived_part. This rule prevents instance recursion rather than entity type recursion.

Notes:

New for CIS/2.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.4 part_map**Entity definition:**

This entity allows a part to be associated with (and hence mapped onto) a number of analytical elements. This entity is one of the linking constructs between the 3 different representations of the structure. In this case, the linking is from the design & manufacturing back to the analytical.

(See also assembly_map and element_mapping.)

EXPRESS specification:

*)

```
ENTITY part_map;
    represented_part : part;
    representing_elements : SET [1:?] OF element;
END_ENTITY;
(*
```

Attribute definitions:*represented_part*

Declares the instance of part associated with the part_map. There must be one (and only one) instance of part referenced by each instance of part_map.

representing_elements

Declares the set of instances of element associated with the part_map. There must be at least one instance of part referenced by each instance of part_map.

Informal propositions:

This is effectively an associative entity that partly resolves the many-to-many relationship between a part and an element. That is, a part can be represented for analysis purposes by many elements and an element may represent many parts.

Notes:

New for CIS/2.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.5 part_prismatic**Entity definition:**

A type of part whose geometry is defined implicitly by one or more 2D cross sections and the length required. The 2D section itself can be defined either implicitly or explicitly through the SUBTYPEs of the entity section_profile.

The prismatic part must be categorized (through its SUBTYPEs) by its geometry as either simple or complex. The prismatic part coordinate system is that described by Clause 1.6.7 of EC3^[12] with the x axis running down the length of the part.

EXPRESS specification:

*)

```
ENTITY part_prismatic
ABSTRACT SUPERTYPE OF (ONEOF
    (part_prismatic_complex,
     part_prismatic_simple))
SUBTYPE OF (part);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity part.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Notes:

Known as PRISMATIC_PART in CIS/1.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.6 part_prismatic_complex**Entity definition:**

A type of prismatic part whose geometry is defined as a series section profiles related to a list of points on a curve. (See also element_curve_complex.) The complexity of the

part is due to its shape rather than its material composition. The `part_prismatic_complex` has an axis, which is defined by a series of (at least 2) points on a curve. At each of these points along the axis, a `section_profile` is referenced which defines the shape of the cross section of the element at that point. It is assumed that between each defining point there is a linear transition between each section profile.

EXPRESS specification:

```

*)
ENTITY part_prismatic_complex
SUPERTYPE OF (part_prismatic_complex_tapered)
SUBTYPE OF (part_prismatic);
    cross_sections : LIST [2:?] OF section_profile;
    points_defining_part_axis : LIST [2:?] OF UNIQUE point_on_curve;
    section_orientations : LIST [2:?] OF orientation_select;
DERIVE
    number_of_sections : INTEGER := SIZEOF(cross_sections);
    curve_defining_part : curve := points_defining_part_axis[1]\point_on_curve.basis_curve;
WHERE
    WRP3 : ( (SIZEOF (points_defining_part_axis) = number_of_sections)
            AND (SIZEOF (section_orientations) = number_of_sections) );
    WRP4 : SIZEOF(QUERY(temp <* points_defining_part_axis |
        (temp\point_on_curve.basis_curve) :<>: curve_defining_part)) = 0;
END_ENTITY;
(*

```

Attribute definitions:

cross_sections

Declares the list of instances of `section_profile` associated with the `part_prismatic_complex`. These instances are used to define the cross section of the prismatic part at various points along its length. There must be at least 2 instances of `section_profile` referenced by each instance of `part_prismatic_complex`. It is assumed that between each defining point there is a linear transition between each section profile. The LIST data type allows a reference to a section profiles to be repeated. If the cross section of the element is constant throughout, then this attribute will reference the same instance of `section_profile` (at least) twice. The section profile is located such that its cardinal point sits on the defining curve of the part. The cardinal point may (but need not) be the geometric centroid of the section profile. The cardinal point may (be need not) be constant for each instance of `section_profile` referenced. In the simplest case, the same cardinal point will be used throughout the part.

points_defining_part_axis

Declares the list of instances of `point_on_curve` associated with the `part_prismatic_complex`. These instances are used to define the points along the length of the prismatic part at which the cross section is defined. There must be at least 2 instances of `point_on_curve` referenced by each instance of `part_prismatic_complex`.

It is assumed that all the instances of `point_on_curve` are all defined on the same instance of curve, as referenced by the attribute `point_on_curve` of the entity `basis_curve`. That is, all the points for which the cross-section is defined all lie on the same curve. It is also assumed that this curve defines the locating longitudinal (x) axis of the prismatic part, upon which the cardinal point(s) of the section profile(s) is (are) placed.

section_orientations

Declares the list of two or more instances of `plane_angle_measure_with_unit` or `direction` associated with this instance of `part_prismatic_complex`. The choice of which entity is used is made through the `SELECT` type `orientation_select`. The order of the instances in the list is significant. Each instance gives the numerical value of the orientation of the section profile at the corresponding point on the part's locating longitudinal axis. As the cardinal point of the section profile defines the locating longitudinal axis of the prismatic part, the rotation of the section profile is defined as being about its cardinal point.

If this attribute references an instance of `plane_angle_measure_with_unit` (via the `SELECT` type) then the orientation of the part is equivalent to a beta angle measured (with an appropriate unit) from the part's z-axis to the plane generated by the projection of the part's local x-axis in the global Z-axis direction. In the case where the part's x-axis lies parallel to the global Z-axis, the orientation of the part will be the angle measured from the part's z-axis to the global X-axis.

If this attribute references an instance of `direction` (via the `SELECT` type) then the orientation of the part is defined by the three real number values of the `direction_ratios` attribute (of the entity `direction`). These numbers define the 'orientation vector' for the section profile. That is, the direction of the re-oriented local z-axis relative to the global coordinate system. For example, the normal (unrotated) orientation for a 'beam' is given as (0.0, 0.0, 1.0), while the orientation vector for unrotated 'columns' is given as (1.0, 0.0, 0.0).

The two different forms of orientation are related such that one may be derived from the other. Thus:

- for beams, the orientation vector = $[0.0, \sin\beta, \cos\beta]$
- for columns, the orientation vector = $[\cos\beta, -\sin\beta, 0.0]$

Thus, the values provided for the orientation vector will also indicate whether the element represents a column or not.

It should be noted that if a direction with three values is used to define the orientation, one of the values assigned to the `direction_ratios` attribute should be zero, while the other two values should be non-zero. As the orientation is defined in one plane (perpendicular to the part's x-axis), a direction assigned three non-zero values would be incorrect.

The fact that the orientation can vary along the length of the prismatic part allows the part to be twisted as shown in Figure 8.1.

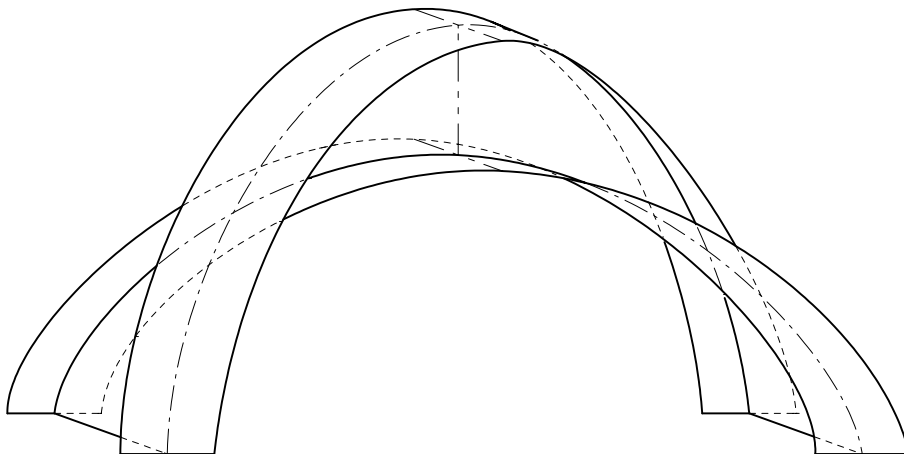


Figure 8.1 *An example of a part_prismatic_complex*

It should be noted that the `section_orientations` attribute does NOT represent the global orientation of the part.

number_of_sections

This attribute derives the total number of cross sections that are used to define the instance of `prismatic_part_complex`. The value is derived by calculating the size of the list of instances of `section_profile` in the populated attribute `cross_sections`.

curve_defining_part

This attribute derives the curve upon which all the defining points of the `prismatic_part_complex` lie. The curve is derived by examining the first instance in the list of instances of `point_on_curve` referenced by the attribute `points_defining_part_axis`.

Formal propositions:

DERIVE

The derived attributes `number_of_sections` and `curve_defining_part` do not appear in the STEP Part 21 file.

WRP3

The size of the list of instances of `point_on_curve` referenced by the attribute `points_defining_part_axis` shall equal the value of the derived attribute `number_of_sections`. Furthermore, the size of the list of instances of `plane_angle_measure_with_unit` referenced by the attribute `section_orientations` shall also equal the value of the derived attribute `number_of_sections`. In other words, for every point defining the axis of the `prismatic_part_complex` there must be a corresponding section profile and angle of orientation.

WRP4

The size of the list of instances of `point_on_curve` having a different `basis_curve` from the first instance of `point_on_curve` shall equal zero. In other words, all the points defining the longitudinal axis of the part must lie on the same curve.

Informal propositions:

The LIST data types used in the attributes `cross_sections` and `points_defining_part_axis` imply that the orders of the instances are important. It is assumed that each instance of `section_profile` referred to by an instance of `part_prismatic_complex` lies on the appropriate instance of `point_on_curve` given in sequence. That is, the third `section_profile` lies on the third `point_on_curve`, and so on.

Notes:

New for CIS/2; partially addressed by `PSEUDO_PRISMATIC_PART` in CIS/1.

See also Figure 5.7 and Figure 5.8 for the relationship between the two different forms of orientation.

This entity does not constrain the defining sections to be all of the same type. Thus, it could be used to define a transition piece (for example between a square and circular hollow section). However, it is recommended that such pieces are defined explicitly using the `part_complex_entity`, and that the `part_prismatic_complex` entity be used with cross-sections of the same type with linear variations in depth or width of the section.

The curve defining the part is initially defined without reference to an external coordinate system; i.e. the curve lies with a local coordinate system. It is only when the part is

referenced by a located_part that the curve is defined relative to a global coordinate system.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.7 part_prismatic_complex_tapered

Entity definition:

A type of complex prismatic part whose geometry varies along its length in a linear fashion from its start to its end. The section can be defined as being tapered in 2 directions; i.e. the depth and width of the section. The increase or decrease in the cross section dimensions (depth and/or width) is defined about the longitudinal axis of the part. Thus, where a prismatic part is used to represent a floor beam with a horizontal top flange, the centroidal axis will be inclined. A positive taper represents an increase in the cross sectional dimensions (i.e. the section is smallest where $x = 0$ and largest where $x = \text{maximum}$). A negative taper represents a decrease in the cross sectional dimensions (i.e. the section is largest where $x = 0$ and smallest where $x = \text{maximum}$).

Where the centroidal axes defines the longitudinal locating axis (i.e. where the cardinal point = 10), skewed end features need to be applied to the ends of the prismatic part if the top of steel is to remain horizontal (see Figure 8.2). In this case, the taper is distributed about the centroidal axis (see Figure 8.3). Alternatively, the part can be defined with the cardinal point at the top of steel (e.g. points 7, 8, or 9) as shown in Figure 8.4.

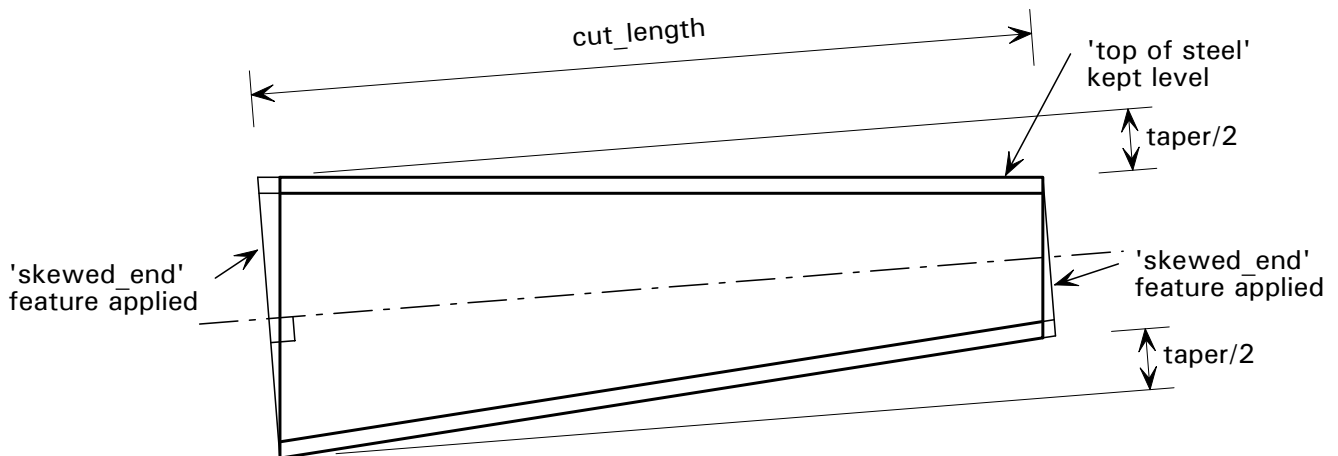


Figure 8.2 Defining a part with a varying depth (Example 1)

EXPRESS specification:

*)

ENTITY part_prismatic_complex_tapered

SUBTYPE OF (part_prismatic_complex);

taper_description : OPTIONAL text;

absolute_taper_1 : OPTIONAL length_measure_with_unit;

absolute_taper_2 : OPTIONAL length_measure_with_unit;

relative_taper_1 : OPTIONAL ratio_measure_with_unit;

relative_taper_2 : OPTIONAL ratio_measure_with_unit;

WHERE

WRP5 : EXISTS (absolute_taper_1) OR EXISTS (absolute_taper_2) OR
EXISTS (relative_taper_1) OR EXISTS (relative_taper_2);

END_ENTITY;

(*

Attribute definitions:

taper_description

An optional text description of the taper represented by the part_prismatic_complex_tapered.

absolute_taper_1

Declares the first instance of length_measure_with_unit that may be associated with the part_prismatic_complex_tapered to provide the numerical value with an appropriate unit of the linear dimension of taper applied in the first direction. The taper is measured perpendicular to the longitudinal axis of the part. When using a local Cartesian coordinate system this dimension will be measured in the local z-direction. A positive value represents an increase in the depth of the cross section (i.e. the section is shallowest where $x = 0$ and deepest where $x = \text{maximum}$). Similarly, a negative value represents a decrease in the depth of the cross section (i.e. the section is deepest where $x = 0$ and shallowest where $x = \text{maximum}$).

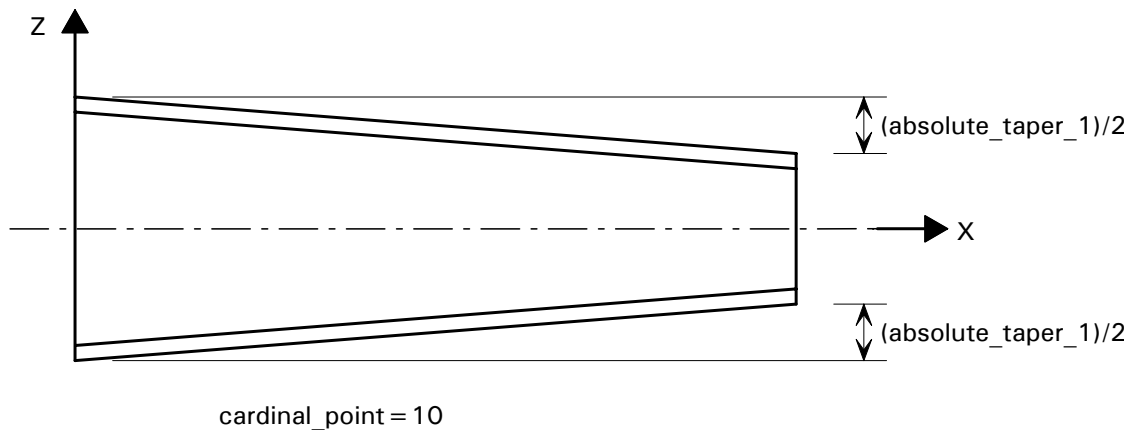


Figure 8.3 *Defining a part with a varying depth (Example 2)*

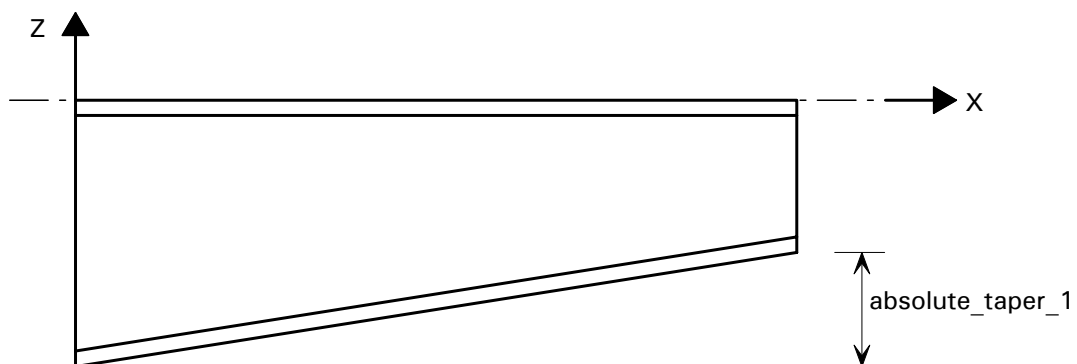


Figure 8.4 *Defining a part with a varying depth (Example 3)*

This attribute is populated when the increase or decrease in the depth of the section is fixed, regardless of the changes in the length of the part.

absolute_taper_2

Declares the second instance of `length_measure_with_unit` that may be associated with the `part_prismatic_complex_tapered` to provide the numerical value with an appropriate unit of the linear dimension of taper applied in the second direction. The taper is measured perpendicular to the longitudinal axis of the part. When using a local Cartesian coordinate system this dimension will be measured in the positive local y-direction. A positive value represents an increase in the width of the cross section (i.e. the section is narrowest where $x = 0$ and widest where $x = \text{maximum}$). Similarly, a negative value represents a decrease in the width of the cross section (i.e. the section is widest where $x = 0$ and where $x = \text{narrowest maximum}$).

This attribute is populated when the increase or decrease in the width of the section is fixed, regardless of the changes in the length of the part.

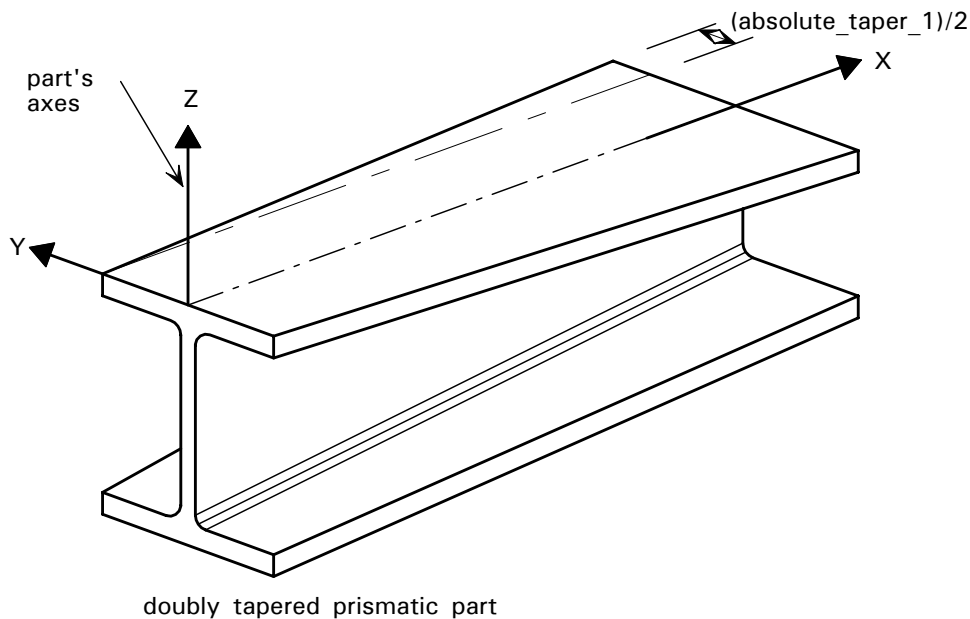


Figure 8.5 *An example of a doubly tapered prismatic part*

relative_taper_1

Declares the first instance of `ratio_measure_with_unit` that may be associated with the `part_prismatic_complex_tapered` to provide the numerical value with an appropriate unit of the linear dimension of taper applied in the first direction. The taper is measured as a proportion of the part's length perpendicular to the longitudinal axis of the part. When using a local Cartesian coordinate system this dimension will be measured in the local z-direction. A positive value represents an increase in the depth of the cross section (i.e. the section is shallowest where $x = 0$ and deepest where $x = \text{maximum}$). Similarly, a negative value represents a decrease in the depth of the cross section (i.e. the section is deepest where $x = 0$ and shallowest where $x = \text{maximum}$).

This attribute is populated when the increase or decrease in depth of the section varies in proportion to the length of the part. In this case, the absolute value of the taper will increase or decrease as the length of the part changes.

relative_taper_2

Declares the second instance of `ratio_measure_with_unit` that may be associated with the `part_prismatic_complex_tapered` to provide the numerical value with an appropriate unit of the linear dimension of taper applied in the second direction. The taper is measured

as a proportion of the part's length perpendicular to the longitudinal axis of the part. When using a local Cartesian coordinate system this dimension will be measured in the local y-direction. A positive value represents an increase in the width of the cross section (i.e. the section is narrowest where $x = 0$ and widest where $x = \text{maximum}$). Similarly, a negative value represents a decrease in the width of the cross section (i.e. the section is widest where $x = 0$ and narrowest where $x = \text{maximum}$).

This attribute is populated when the increase or decrease in width of the section varies in proportion to the length of the part. In this case, the absolute value of the taper will increase or decrease as the length of the part changes.

Formal propositions:

WRP5

At least one of the attributes `absolute_taper_1`, `absolute_taper_2`, `relative_taper_1`, or `relative_taper_2` must be populated. In other words, a tapered part cannot exist without having a taper specified.

Informal propositions:

It should be noted that the taper is defined about the part's locating longitudinal axis, and that the ends of the part are perpendicular to the part's longitudinal axis. The part's locating longitudinal axis is defined by the cardinal point(s) of the section profile(s) referenced by the part. Where the cardinal point is not given at the 'top of steel' (points 7, 8, 9, or 14) but the top of steel is to stay level, features need to be applied to the ends of the part to ensure that they are perpendicular to the 'top of steel' (and thus skew to the part's longitudinal axis).

Where the taper is specified in both relative and absolute terms, this does not mean that the part is tapered twice. The absolute and relative tapers provide different definitions of the same taper. The taper resulting from the relative values should correspond to that provided by the absolute values. (If they do not, the relative values take precedence.)

Notes:

Known as `TAPERED_PP` in CIS/1.

See diagram 64 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.8 `part_prismatic_simple`

Entity definition:

A type of prismatic part whose cross sectional geometry is defined using one instance of `section_profile`. The part's geometry is assumed to be uniform along its length and is defined implicitly by a 2D cross section and the length. The prismatic part coordinate system is that described by Clause 1.6.7 of EC3^[12] with the x axis running down the length of the part.

The prismatic part must be given a cut length, which may be given an offset from the original stock length. The stock material maybe regarded as finite or infinite in length. The dimensions given are those before any features are applied and the part is defined as having square cut ends.

Both 'uniform' and 'regular' prismatic parts may be represented by instances of `part_prismatic_simple`. Uniform prismatic parts refer to those parts possessing an invariable sectional geometry throughout the length of the part (i.e. truly prismatic), while regular prismatic parts refer to those parts possessing a repeating feature (such as a

hole) along its length. The SUBTYPEs of this entity allow for the definition of cambered, castellated, and curved beams. The ANDOR construct in the SUPERTYPE allows a prismatic part to be more than one of its SUBTYPEs. In such a case, the part is only fully defined when all the attributes of each the SUBTYPEs `part_prismatic_simple_cambered`, and `part_prismatic_simple_castellated`, are populated.

EXPRESS specification:

```

*)
ENTITY part_prismatic_simple
SUPERTYPE OF (part_prismatic_simple_cambered ANDOR
               part_prismatic_simple_castellated ANDOR
               part_prismatic_simple_curved)
SUBTYPE OF (part_prismatic);
  profile : section_profile;
  cut_length : positive_length_measure_with_unit;
  stock_length : OPTIONAL positive_length_measure_with_unit;
  x_offset : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
(*

```

Attribute definitions:

profile

Declares the instance of `section_profile` associated with the `part_prismatic_simple`, which is used to define the cross section geometry of the prismatic part. There must be one (and only one) instance of `section_profile` referenced by each instance of `part_prismatic_simple`.

cut_length

Declares the first instance of `positive_length_measure_with_unit` associated with the `part_prismatic_simple`, which specifies the numerical value with an appropriate unit of the original cut length of the prismatic part.

The cut length is the length of the part as cut from the stock piece with square ends before any features are applied.

stock_length

Declares the second instance of `positive_length_measure_with_unit` that may be associated with the `part_prismatic_simple` to specify the numerical value with an appropriate unit of the stock length of the prismatic part. If provided, the value of the stock length must be greater than the value of the `cut_length`.

x_offset

Declares the third instance of `positive_length_measure_with_unit` associated with the `part_prismatic_simple`, which may be used to specify the numerical value with an appropriate unit of the offset dimension measured along the stock length of the part to the position of the first cut.

Formal propositions:**ANDOR SUPERTYPE**

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Informal propositions:

If provided, the value of the stock length must be greater than the value of the cut_length.

Notes:

New for CIS/2; addressed by PRISMATIC_PART in CIS/1.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.9 part_prismatic_simple_cambered**Entity definition:**

A type of simple prismatic part whose longitudinal axis is distorted from the nominal axis during the manufacturing process by being displaced in one or two directions. The nominal axis of the part is defined by the locus of the cardinal point of the section profile that defines the shape of the part between its two ends.

The magnitude of the distortion from the nominal axis is assumed to vary linearly from zero at the ends of the part to the maximum value given by the attributes of the SUBTYPE. If a camber needs to be defined as a curve, then the entity part_prismatic_simple_curved should be used in preference to this entity.

EXPRESS specification:

*)

ENTITY part_prismatic_simple_cambered

SUPERTYPE OF

(part_prismatic_simple_cambered_absolute ANDOR
part_prismatic_simple_cambered_relative)

SUBTYPE OF (part_prismatic_simple);

camber_description : OPTIONAL text;

END_ENTITY;

(*

Attribute definitions:**camber_description**

An optional text description of the type of camber applied to the prismatic part.

Notes:

New for CIS/2.

See diagram 64 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.10 part_prismatic_simple_cambered_absolute

Entity definition:

A type of `part_prismatic_simple_cambered` where the offset is provided by two instances of `length_measure_with_unit`.

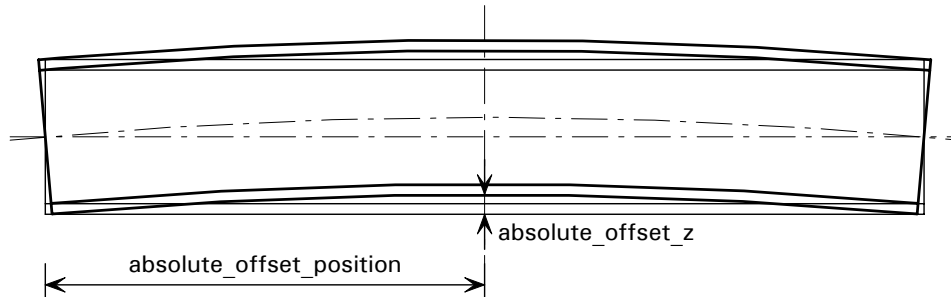


Figure 8.6 *Defining an absolute camber*

EXPRESS specification:

```
*)
ENTITY part_prismatic_simple_cambered_absolute
SUBTYPE OF (part_prismatic_simple_cambered);
    absolute_offset_position : positive_length_measure_with_unit;
    absolute_offset_y : length_measure_with_unit;
    absolute_offset_z : length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

absolute_offset_position

Declares the instance of `positive_length_measure_with_unit` that is associated with the `part_prismatic_simple_cambered` to provide the numerical value with an appropriate unit of the linear dimension to the point where the camber offset is measured. The position is measured along the length of the part from its origin. When using a local Cartesian coordinate system this dimension will be measured in the positive x-direction.

absolute_offset_y

Declares the first instance of `length_measure_with_unit` that is associated with the `part_prismatic_simple_cambered` to provide the numerical value with an appropriate unit of the linear dimension of offset in the first direction. The offset is measured perpendicular to the longitudinal axis, in the positive y-direction.

absolute_offset_z

Declares the second instance of `length_measure_with_unit` that may be associated with the `part_prismatic_simple_cambered` to provide the numerical value with an appropriate unit of the linear dimension of offset in the second direction. The offset is measured perpendicular to the longitudinal axis, in the positive z-direction.

Informal propositions:

Where the camber is specified in both relative and absolute terms, this entity is instanced with its sibling SUBTYPE `part_prismatic_simple_cambered_relative`. Such a complex instance is encoded in a STEP Part 21 file using external mapping. This does not mean

that there is an offset at two positions. The absolute and relative cambers provide different definitions of the same camber. Both the size of the offset and the position of the offset provided by the `part_prismatic_simple_cambered_absolute` should correspond to those provided `part_prismatic_simple_cambered_relative`. (If they do not, the values provided by the `part_prismatic_simple_cambered_relative` take precedence.)

Notes

New for CIS/2.

See Diagram 64 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition - Attribute definitions clarified.

8.3.11 part_prismatic_simple_cambered_relative

Entity definition:

A type of `part_prismatic_simple_cambered` where the offset is provided by two instances of `ratio_measure_with_unit`.

EXPRESS specification:

```
*)
ENTITY part_prismatic_simple_cambered_relative
SUBTYPE OF (part_prismatic_simple_cambered);
    relative_offset_position : ratio_measure_with_unit;
    relative_offset_x : ratio_measure_with_unit;
    relative_offset_y : ratio_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

relative_offset_position

Declares the first instance of `ratio_measure_with_unit` that may be associated with the `part_prismatic_simple_cambered` to provide the numerical value with an appropriate unit of the linear dimension to the point where the camber offset is measured. The position is measured along the length of the part from its origin as a proportion of its length.

relative_offset_y

Declares the second instance of `ratio_measure_with_unit` that may be associated with the `part_prismatic_simple_cambered` to provide the numerical value with an appropriate unit of the linear dimension of offset in the first direction. The offset is measured perpendicular to the longitudinal axis as a proportion of the length of the part, in the positive y-direction.

relative_offset_z

Declares the third instance of `ratio_measure_with_unit` that may be associated with the `part_prismatic_simple_cambered` to provide the numerical value with an appropriate unit of the linear dimension of offset in the second direction. The offset is measured perpendicular to the longitudinal axis as a proportion of the length of the part, in the positive z-direction.

Informal propositions:

Where the camber is specified in both relative and absolute terms, this entity is instanced with its sibling SUBTYPE `part_prismatic_simple_cambered_absolute`. Such a complex

instance is encoded in a STEP Part 21 file using external mapping. This does not mean that there is an offset at two positions. The absolute and relative cambers provide different definitions of the same camber. Both the size of the offset and the position of the offset provided by the `part_prismatic_simple_campered_absolute` should correspond to those provided `part_prismatic_simple_campered_relative`. (If they do not, the values provided by the `part_prismatic_simple_campered_relative` take precedence.)

Notes

New for CIS/2.

See Diagram 64 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition - Attribute definitions clarified.

8.3.12 `part_prismatic_simple_castellated`

Entity definition:

A type of simple `part_prismatic_simple` that possesses a repeating feature (such as a hole) along its length. The repeating feature may be given a classification by its type (circular, hexagonal, or octagonal). It may also be defined in terms of spacing, height and width through references to `positive_length_measure_with_unit`.

EXPRESS specification:

*)

ENTITY `part_prismatic_simple_castellated`

SUBTYPE OF (`part_prismatic_simple`);

`part_castellation_type` : `castellation_type`;

`end_post_width_1` : `positive_length_measure_with_unit`;

`end_post_width_2` : OPTIONAL `positive_length_measure_with_unit`;

`castellation_spacing` : OPTIONAL `positive_length_measure_with_unit`;

`castellation_height` : OPTIONAL `positive_length_measure_with_unit`;

`castellation_width` : OPTIONAL `positive_length_measure_with_unit`;

`castellation_depth` : OPTIONAL `positive_length_measure_with_unit`;

END_ENTITY;

(*

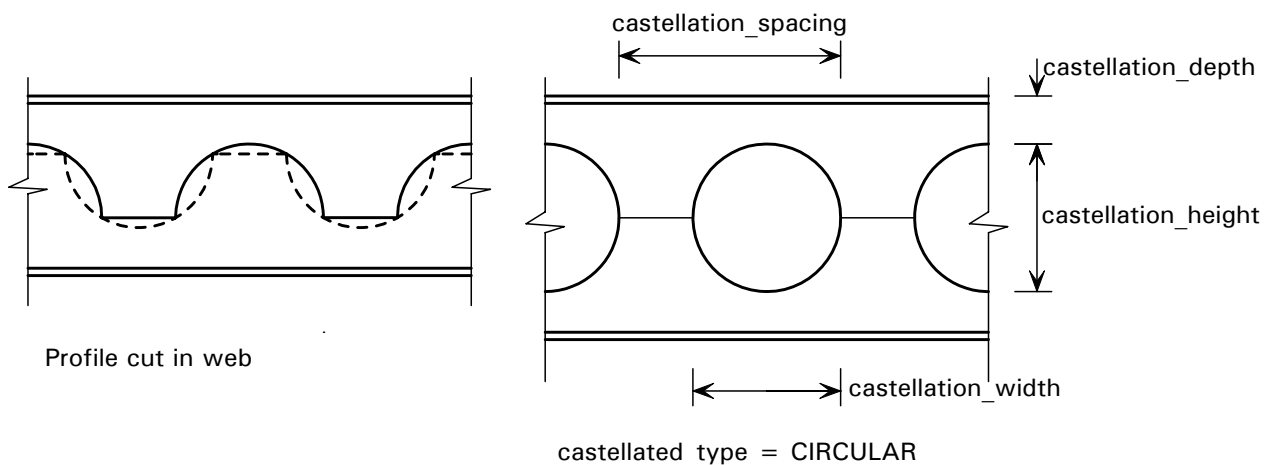


Figure 8.7 Example of `part_prismatic_simple_castellated`

Attribute definitions:***part_castellation_type***

Declares the classification of the type of longitudinal repeating feature by its shape (i.e. circular, hexagonal, or octagonal). The `part_castellation_type` may be declared as being undefined. Traditional castellated beams have a hexagonal castellation type, while the more recent ‘cellform’ beams have a circular castellation type (even where the repeating feature was not a true circle).

end_post_width_1

Declare the first instance of `positive_length_measure_with_unit` associated with the `part_prismatic_simple_castellated`. This provides the numerical value with an appropriate unit of the linear dimension to the first web opening measured from the first cut end of the part ($x=0$). This allows for one or more of the initial castellations to be ‘filled in’.

end_post_width_2

Declare the second instance of `positive_length_measure_with_unit` that may be associated with the `part_prismatic_simple_castellated`. This provides the numerical value with an appropriate unit of the linear dimension measured from the last web opening to the second cut end of the part ($x=\max$). This allows for one or more of the final castellations to be ‘filled in’.

castellation_spacing

Declares the third instance of `positive_length_measure_with_unit` that may be associated with the `part_prismatic_simple_castellated` to specify the numerical value with an appropriate unit of the linear distance between each feature, measured along the length of the part. The length is measured between corresponding points on the repeating feature.

castellation_height

Declares the fourth instance of `positive_length_measure_with_unit` that may be associated with the `part_prismatic_simple_castellated` to specify the numerical value with an appropriate unit of the linear distance between the top and bottom of each feature, measured across the depth of the part.

castellation_width

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `part_prismatic_simple_castellated` to specify the numerical value with an appropriate unit of the linear distance between the extreme edges of each feature, measured along the length of the part.

castellation_depth

Declares the sixth instance of `positive_length_measure_with_unit` that may be associated with the `part_prismatic_simple_castellated` to specify the numerical value with an appropriate unit of the linear distance between the top of the part and the extreme edges of each feature, measured down the depth of the part. (The centre of the castellation need not lie at the mid-depth of the part.)

Informal propositions

The `section_profile` referenced (by the attribute profile inherited from the SUPERTYPE `part_prismatic_simple`) provides the dimensions of the final section. If the dimensions of the original section are required, the attribute profile should refer to an instance of `section_profile_derived`. For an asymmetric section (made from two different section sizes) the attribute profile should refer to an instance of `section_profile_compound`.

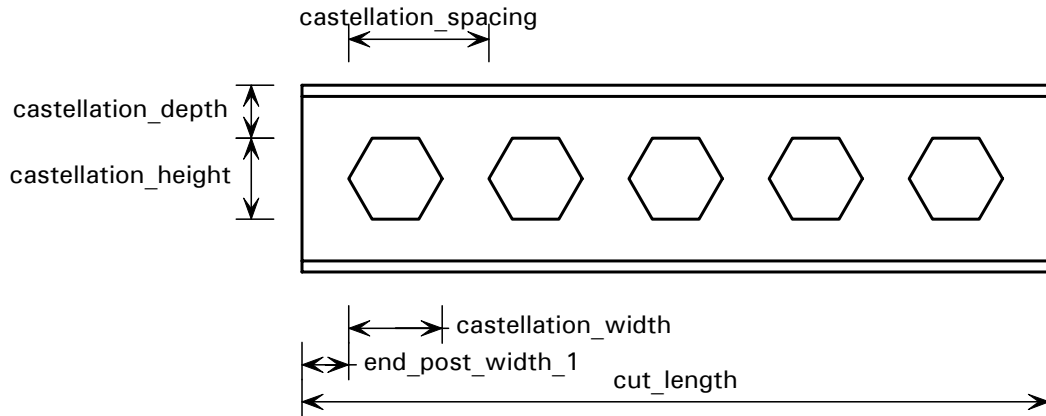


Figure 8.8 *Dimensions for castellated prismatic parts*

Notes:

New for CIS/2.

See diagram 64 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.13 part_prismatic_simple_curved

Entity definition:

A type of simple prismatic part whose longitudinal axis is defined explicitly using a curve from STEP 42.

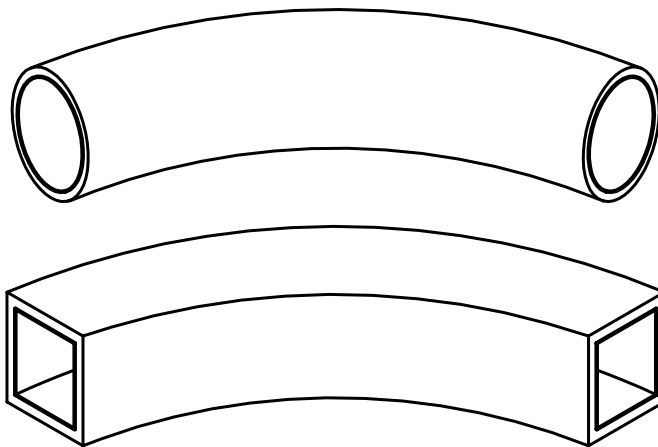


Figure 8.9 *Examples of part_prismatic_simple_curved*

EXPRESS specification:

```

*)
ENTITY part_prismatic_simple_curved
  SUBTYPE OF (part_prismatic_simple);
    axis_definition : curve;
END_ENTITY;
(*

```


Attribute definitions:***axis_definition***

Declares the instance of curve associated with the part_prismatic_simple_curved, which is used to define the longitudinal axis of prismatic part. It is assumed that the defining point of the section is placed on this longitudinal axis. There must be one (and only one) instance of curve referenced by each instance of part_prismatic_simple_curved. The section profile referenced by the SUPERTYPE part_prismatic_simple is positioned such that its cardinal point sits on the axis defined by this attribute.

The curve defining the part is initially defined without reference to an external coordinate system; i.e. the curve lies with a local coordinate system. It is only when the part is referenced by a located_part that the curve is defined relative to a global coordinate system.

Notes:

New for CIS/2.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.14 part_sheet**Entity definition:**

A type of (specific) part whose geometry is either uniform or regular in a plane. When using a Cartesian local coordinate system, a sheet part is defined as lying in the yz-plane. Plane, corrugated and cellular parts are treated in this way. The stock material may be regarded as finite or infinite in length. The sheet part can be bounded or unbounded. It may also be given a profile. The sheet part must be given a thickness, which is taken to be constant. For flat sheets, the material is evenly distributed about the yz-plane; i.e. the yz-plane lies at mid thickness of the sheet. For profiled sheet, the yz-plane lies at the mid-depth of profile. (For non-symmetrical profiles, this will mean that the material is not evenly distributed about the yz-plane, even though the material thickness of the sheet is still constant.)

EXPRESS specification:

*)

ENTITY part_sheet

SUPERTYPE OF (part_sheet_bounded ANDOR part_sheet_profiled)

SUBTYPE OF (part);

sheet_thickness : positive_length_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:***sheet_thickness***

Declares the instance of positive_length_measure_with_unit associated with the part_sheet, which is used to specify the numerical value with an appropriate unit of the out-of-plane linear dimension of the sheet part. When using a Cartesian local coordinate system, this is the dimension is defined in the x-direction. There must be one (and only one) instance of length_measure_with_unit referenced by each instance of part_sheet.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

Known as SHEET_PART in CIS/1.

Where the material thickness is not constant, the part should be represented using the entity part_complex.

See diagram 62 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.15 part_sheet_bounded

Entity definition:

A type of sheet part whose in-plane boundary is defined as either simple or complex.

EXPRESS specification:

```
*)
ENTITY part_sheet_bounded
ABSTRACT SUPERTYPE OF (ONEOF
    (part_sheet_bounded_complex,
    part_sheet_bounded_simple))
SUBTYPE OF (part_sheet);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE part_sheet. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity and to collate the SUBTYPES under a common category.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2.

See diagram 63 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.16 part_sheet_bounded_complex

Entity definition:

A type of bounded sheet part whose in-plane boundary is defined using a bounded_surface - an explicit geometry construct from STEP Part 42.

EXPRESS specification:

```

*)
ENTITY part_sheet_bounded_complex
SUBTYPE OF (part_sheet_bounded);
    sheet_boundary : bounded_surface;
END_ENTITY;
(*)

```

Attribute definitions:*sheet_boundary*

Declares the instance of `bounded_surface` associated with the `part_sheet_bounded_complex`, which is used to define the in-plane boundary of the sheet part. There must be one (and only one) instance of `bounded_surface` referenced by each instance of `part_sheet_bounded_complex`.

Notes:

New for CIS/2.

The entity `bounded_surface` is defined in STEP Part 42^[24].

See diagram 63 of the EXPRESS-G diagrams in Appendix B.

8.3.17 part_sheet_bounded_simple**Entity definition:**

A type of bounded sheet part whose in-plane boundary is defined using the parameters contained as attributes of this entity. It is assumed that a Cartesian coordinate system is used and the dimensions specified are defined in the y and z-directions. The dimensions given are those before any features are applied and the sheet part is defined as rectangular.

EXPRESS specification:

```

*)
ENTITY part_sheet_bounded_simple
SUBTYPE OF (part_sheet_bounded);
    cut_y_dimension : positive_length_measure_with_unit;
    cut_z_dimension : positive_length_measure_with_unit;
    stock_y_dimension : OPTIONAL positive_length_measure_with_unit;
    stock_z_dimension : OPTIONAL positive_length_measure_with_unit;
    y_offset : OPTIONAL positive_length_measure_with_unit;
    z_offset : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
(*)

```

Attribute definitions:*cut_y_dimension*

Declares the first instance of `positive_length_measure_with_unit` associated with the `part_sheet_bounded_simple`, which is used to specify the numerical value with an appropriate unit of the original cut length of the sheet part in the y-direction.

cut_z_dimension

Declares the second instance of `positive_length_measure_with_unit` associated with the `part_sheet_bounded_simple`, which is used to specify the numerical value with an appropriate unit of the original cut length of the sheet part in the z-direction.

stock_y_dimension

Declares the third instance of `positive_length_measure_with_unit` that may be associated with the `part_sheet_bounded_simple` to specify the numerical value with an appropriate unit of the stock length of the sheet part in the y-direction. If provided, this value should be greater than the value of the cut length in the y-direction. Several sheet parts can be cut from the same stock piece.

stock_z_dimension

Declares the fourth instance of `positive_length_measure_with_unit` that may be associated with the `part_sheet_bounded_simple` to specify the numerical value with an appropriate unit of the stock length of the sheet part in the z-direction. If provided, this value should be greater than the value of the cut length in the z-direction. Several sheet parts can be cut from the same stock piece.

y_offset

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `part_sheet_bounded_simple` to specify the numerical value with an appropriate unit of the offset dimension measured along the y-axis of the part applicable from the edge of the stock length to the first cut perpendicular to the y-axis.

z_offset

Declares the sixth instance of `positive_length_measure_with_unit` that may be associated with the `part_sheet_bounded_simple` to specify the numerical value with an appropriate unit of the offset dimension measured along the z-axis of the part applicable from the edge of the stock length to the first cut perpendicular to the z-axis.

Notes:

New for CIS/2; addressed by `SHEET_PART` in CIS/1.

See diagram 63 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

8.3.18 part_sheet_profiled**Entity definition:**

A type of sheet part whose out-of-plane shape is distorted to form a profile. When using a Cartesian local coordinate system, the through thickness has been defined in the x-direction and thus the profile is defined in either the xy-plane or in the xz-plane depending on the definition of the curve. The curve is an explicit geometry construct taken from STEP Part 42.

EXPRESS specification:

*)

ENTITY `part_sheet_profiled`

SUBTYPE OF (`part_sheet`);

`sheet_profile` : curve;

`profile_properties` : OPTIONAL `section_properties`;

END_ENTITY;

(*)

Attribute definitions:***sheet_profile***

Declares the instance of curve associated with the `part_sheet_profiled`, which defines the profile of the sheet part. There must be one (and only one) instance of curve referenced by each instance of `part_sheet_profiled`.

profile_properties

Declares the instance of `section_properties` associated with the `part_sheet_profiled`, which may be used to define the geometric properties of the profile of the sheet part. It should be noted that the section properties apply to a unit width of the sheet part.

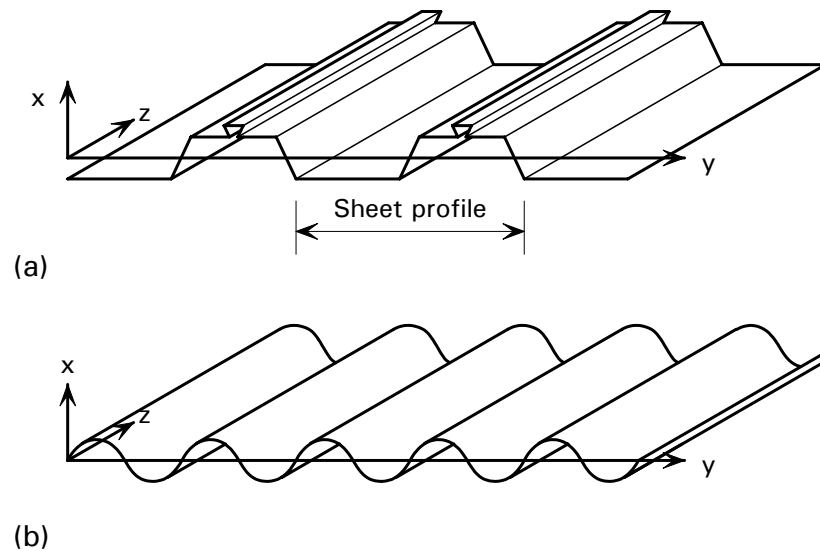


Figure 8.10 *Examples of part_sheet_profiled*

Notes:

New for CIS/2.

The entity curve is defined in STEP Part 42^[24].

See diagram 63 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9 LPM/6 FEATURES

9.1 Features concepts and assumptions

The Features subject area covers the description and specification of specific features, which are then used within located features. That is, features are described at the specific level, and then implemented many times at the occurrence level. (This can be thought of as the ‘type’ verses ‘token’ approach seen in many database applications; with ‘types’ residing at the specific level and ‘tokens’ residing at the occurrence level.) Thus, a located feature is defined as a located and oriented implementation of the corresponding (specific) feature.

Located features are used to modify assemblies, parts and joint systems. Features are categorized by the complexity of their shape definition and by their use, e.g. edge, surface, or volume. Volumetric features include notches, chamfers, and holes. Features also include surface treatment and name tags.

Volumetric features are local variations to the basic geometry of the part (or assembly) to which they apply. All the implemented volume features remove material from the original part; i.e. they are subtractive features. Volumetric features never add material to a part; this must be done using other parts.

Surface features modify the surface of the part to which the feature is applied; for example, the application of paint is regarded in the LPM as a surface feature. If unbounded, surface features apply to the whole surface of the part. To apply a particular surface treatment to one area of the surface of a part, the surface feature must be bounded using the entities `feature_surface_simple` or `feature_surface_complex`.

Features have their own local coordinate system, which is then transformed to fit the part’s coordinate system. In some cases, the coordinate system and transformation is implicit, rather than explicit; for example, prismatic end features and edge chamfers are located with respect to the part to which they are applied.

Features are located with respect to the part they modify. Parts and joint systems are located with respect to the assembly to which they belong. Each assembly has its own coordinate system. Similarly each part, joint system and feature have their own coordinate systems. Each part, feature, and joint system coordinate system must be located within a parent coordinate system.

The use of coordinate systems in the LPM is essential. Everything in the design model and physical model is located by its coordinate system. Coordinate systems are detailed the LPM/6 Location subject area described in Section 12.

9.2 Features type definitions

No types have been added or modified for 2nd Edition in this subject area.

9.2.1 left_or_right

Type definition:

Classifies the prismatic feature as being applied to either the ‘left hand side’ or the ‘right hand side’ of the prismatic part. With reference to the part’s coordinate system, the left hand side is where y is positive and the right hand side is where y is negative. This definition of left and right holds for both ends of the prismatic part. See Figure 9.1.

EXPRESS specification:

```

*)
TYPE left_or_right
= ENUMERATION OF
    (left_hand, right_hand);
END_TYPE;
(*)

```

Notes

New for CIS/2. Unchanged for 2nd Edition.

9.2.2 start_or_end_face**Type definition:**

Classifies the feature is applying to the start or end face of the prismatic part. With reference to the part's coordinate system, the start face is where $x=0$, while the end face is where x has its maximum value. See feature_volume_prismatic and Figure 9.1.

EXPRESS specification:

```

*)
TYPE start_or_end_face
= ENUMERATION OF
    (start_face, end_face);
END_TYPE;
(*)

```

Notes:

Known as st_or_end_face in CIS/1. Unchanged for 2nd Edition.

9.2.3 top_or_bottom**Type definition:**

Classifies the feature is applying to the top or bottom edge of the prismatic part. See feature_volume_prismatic. With reference to the part's coordinate system, the top is where z has its maximum positive value, while the bottom is where z has its maximum negative value. This definition of top and bottom holds for both ends of the prismatic part. See Figure 9.1.

EXPRESS specification:

```

*)
TYPE top_or_bottom
= ENUMERATION OF
    (top_edge, bottom_edge);
END_TYPE;
(*)

```

Notes:

New as top_or_bottom in CIS/1. Unchanged for 2nd Edition.

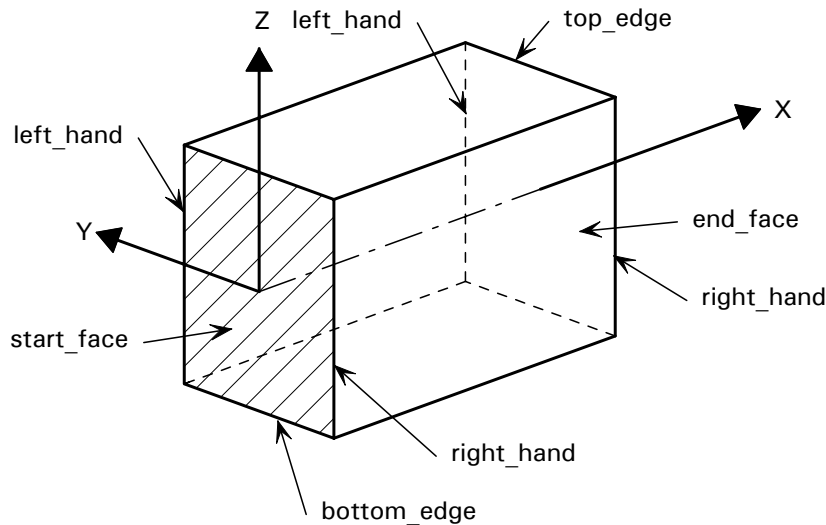


Figure 9.1 Definition of 'left', 'right', 'top', 'bottom', 'start' and 'end'

9.3 Features entity definitions

The following entities have been added to this subject area for the 2nd Edition:

- feature_surface_point
- feature_surface_point_mark
- feature_volume_with_depth
- feature_volume_with_limit

The following entities have been modified in this subject area for the 2nd Edition:

- feature
- feature_surface
- feature_volume

9.3.1 feature

Entity definition:

A specific level entity which defines the type and (where applicable) the shape of (one or more) corresponding located features. Each instance of feature represents one (type) of feature. This may then be referenced by as many instances of located_feature (token) as require. A feature has its own implicit 3D coordinate system.

A feature is a variation to the basic form of a part or assembly; e.g. the removal of material or a treatment applied to the surface of a part. It would normally be instanced through one of its SUBTYPEs, which categorize the feature as either volumetric, surface, or plane. A thread can also be described as a (special) category of feature.

As a SUBTYPE of structural_frame_item, a feature inherits the three attributes item_number, item_name, and item_description. It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a feature may have:

- approvals (via the entity structural_frame_item_approved)

- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the feature to be described in detail
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A feature may also be related to another feature (or any other structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

```

*)
ENTITY feature
  SUPERTYPE OF (ONEOF
    (feature_cutting_plane,
     feature_edge_chamfer,
     feature_surface,
     feature_thread,
     feature_volume))
  SUBTYPE OF (structural_frame_item);
  DERIVE
    uses : SET [0:?] OF located_feature := bag_to_set
      (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
        LOCATED_FEATURE.DESCRPTIVE_FEATURE'));
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit the attributes item_number, item_name, and item_description from its SUPERTYPE structural_frame_item. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity structural_frame_item and bring together the SUBTYPES under a common category.

Notes:

Known as S_FEATURE in CIS/1.

This entity can be thought of as a use of the ‘type’ verses ‘token’ approach seen in many database applications; with the feature entity representing a ‘type’ of feature and the located_feature entity representing ‘tokens’ of that feature.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

9.3.2 feature_cutting_plane

Entity definition:

A type of feature, where the feature’s yz-plane acts as a boundary between the part and the material to be removed. The cutting plane is effectively unbounded, and thus, becomes a semi-infinite half-space. The cutting plane is used by rotating the feature’s

coordinate system to the part's coordinate system to suit the cut required. The cutting plane is defined such that the waste material is removed from the positive x side of the plane.

EXPRESS specification:

```
*)
ENTITY feature_cutting_plane
  SUBTYPE OF (feature);
    plane_definition : plane;
END_ENTITY;
(*
```

Attribute definitions:

plane_definition

Declares the instance of plane associated with the feature_cutting_plane, which defines the 2D cutting plane. There must be one (and only one) instance of plane referenced by each instance of feature_cutting_plane.

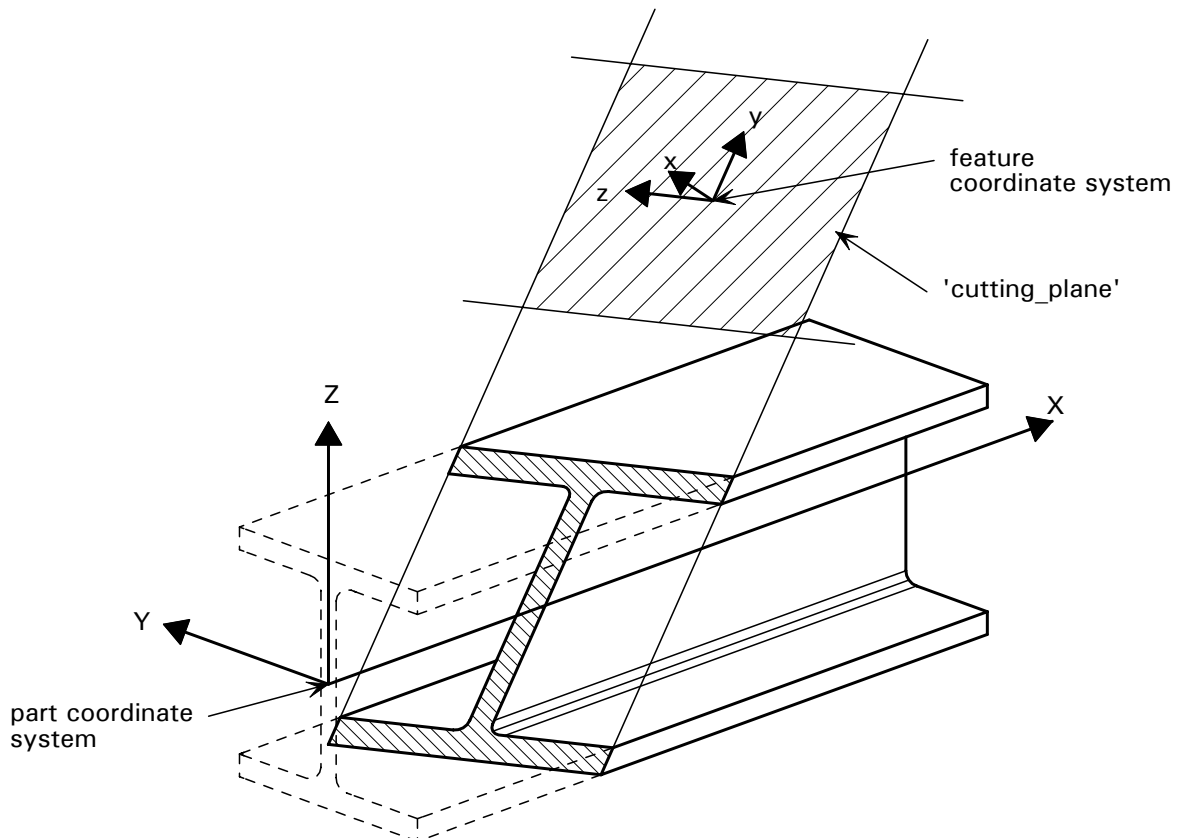


Figure 9.2 An example of a cutting plane applied to a prismatic part

Notes:

Known as CUTTING_PLANE in CIS/1.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.3 feature_edge_chamfer

Entity definition:

A type of feature which describes the geometry and location of a chamfer along the edge of a part; for example, a weld preparation along the edge of a plate. In applying a chamfer to an edge of a part, one surface on that part is identified as a reference surface. The edge chamfer is defined such that, the feature's y-axis aligns with the reference surface of the part to which it is applied (see Figure 9.3). This is particularly important where the edge is not a right angle. Further, if the attribute `follow_round` is assigned as `TRUE`, then the feature's x-axis follows the complete edge of the reference surface, keeping the feature's y-axis pointing inwards (i.e. towards the centre of the reference surface of the part).

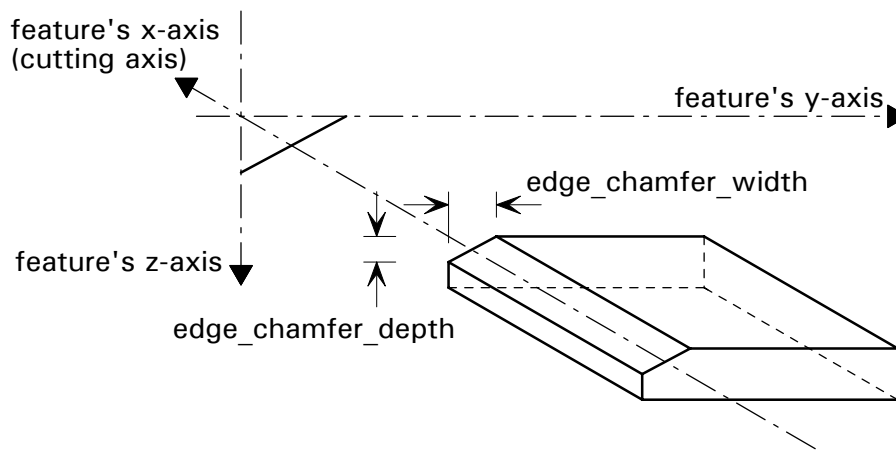


Figure 9.3 Example of a straight edge chamfer applied to a part

This entity must be instantiated as one of its SUBTYPEs. Its SUBTYPEs are used to describe, for example, the rounding of a corner on the edge of a flange of a beam (see Figure 9.4.)

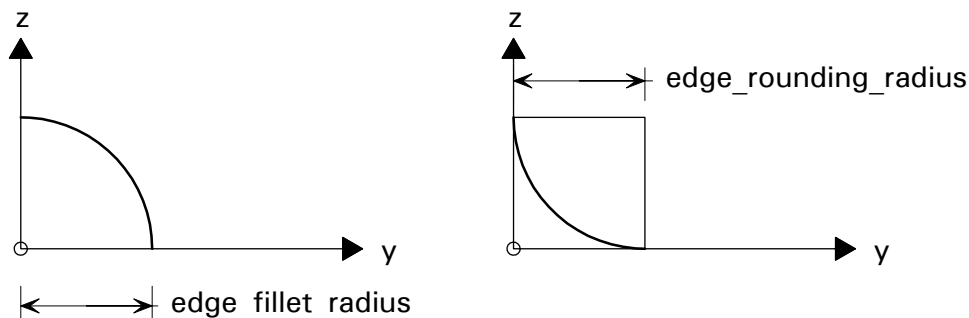


Figure 9.4 Examples of types of edge chamfer

EXPRESS specification:

```

*)
ENTITY feature_edge_chamfer
ABSTRACT SUPERTYPE OF (ONEOF
    (feature_edge_chamfer_straight,
     feature_edge_chamfer_fillet,
     feature_edge_chamfer_rounding) )
SUBTYPE OF (feature);
    follow_round : BOOLEAN;
  
```

END_ENTITY;
(*

Attribute definitions:

follow_round

Declares whether the edge chamfer is to be applied along the complete edge (in the same plane) i.e. to the full profile of the part (TRUE) or simply to one straight edge (FALSE).

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

Known as EDGE_CHAMFER in CIS/1.

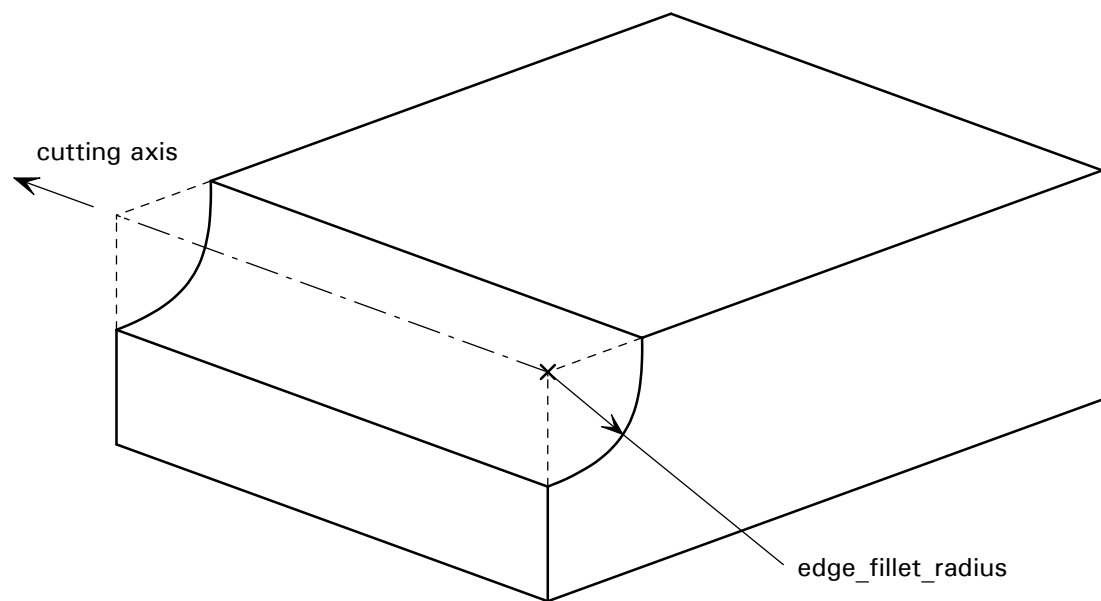
See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.4 feature_edge_chamfer_fillet

Entity definition:

A type of edge chamfer where the cut is defined by the surface revolved from a circle whose cutting axis is placed at the original edge of the part, with the centre of the circle on the cutting axis.



feature_edge_chamfer_fillet

Figure 9.5 Definition of a feature_edge_chamfer_fillet

EXPRESS specification:

*)
ENTITY feature_edge_chamfer_fillet
SUBTYPE OF (feature_edge_chamfer);
 edge_fillet_radius : positive_length_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:

edge_fillet_radius

Declares the instance of `positive_length_measure_with_unit` associated with the `feature_edge_chamfer_fillet`, which is used to define the numerical value with an appropriate unit of the linear dimension from the intersection of the cutting edge and the reference surface to the circle that defines the fillet. There must be one (and only one) instance of `positive_length_measure_with_unit` referenced by each instance of `feature_edge_chamfer_fillet`. (See Figure 9.4.)

Notes:

New for CIS/2; addressed by `EDGE_CHAMFER` in CIS/1.

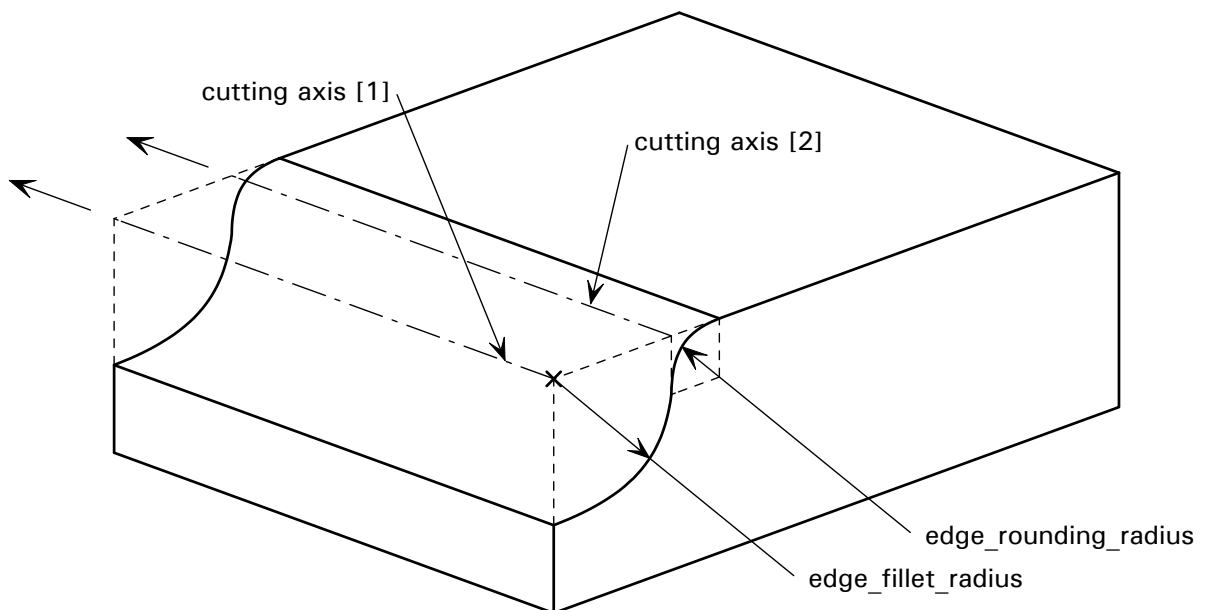
See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.5 feature_edge_chamfer_rounding

Entity definition:

A type of edge chamfer where the cut is defined by the surface revolved from a circle that is placed tangentially to the two faces of the part.



[1] `feature_edge_chamfer_fillet` + [2] `feature_edge_chamfer_rounding`

Figure 9.6 Example of a 'rounding' and a 'fillet' applied to an edge

EXPRESS specification:

*)

ENTITY `feature_edge_chamfer_rounding`

SUBTYPE OF (`feature_edge_chamfer`);

`edge_rounding_radius` : `positive_length_measure_with_unit`;

END_ENTITY;

(*)

Attribute definitions:*edge_rounding_radius*

Declares the instance of `positive_length_measure_with_unit` associated with the `feature_edge_chamfer_rounding`, which is used to define the numerical value with an appropriate unit of the linear dimension from the cutting edge to the centre of the circle used to define the edge chamfer. There must be one (and only one) instance of `length_measure_with_unit` referenced by each instance of `feature_edge_chamfer_rounding`. (See Figure 9.4.)

Notes:

New for CIS/2; addressed by `EDGE_CHAMFER` in CIS/1.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.6 feature_edge_chamfer_straight**Entity definition:**

A type of edge chamfer where the cut is defined by the plane that intersects the two original edges of the part by specified amounts.

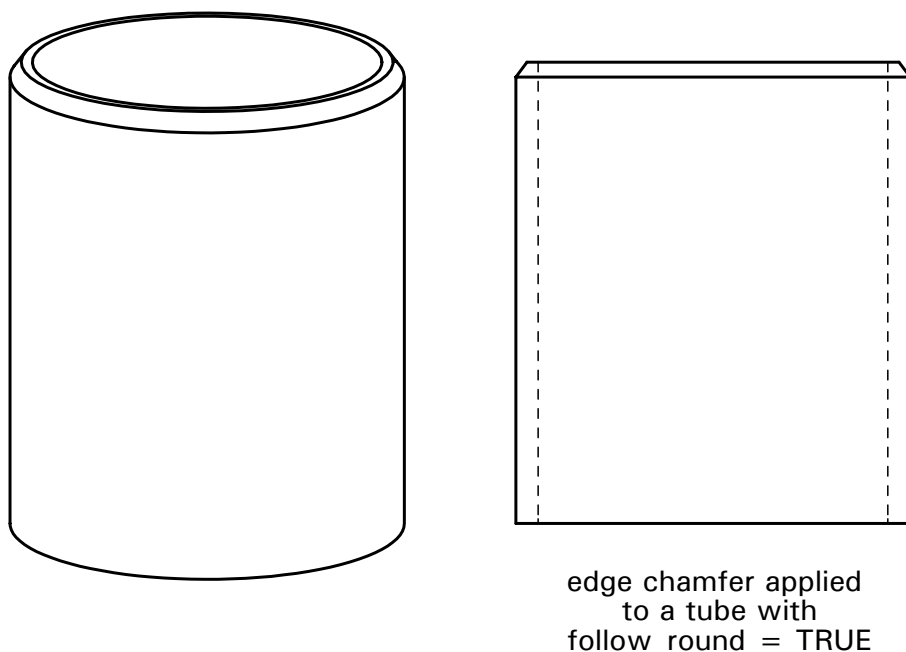


Figure 9.7 Example of a *feature_edge_chamfer_straight*

EXPRESS specification:

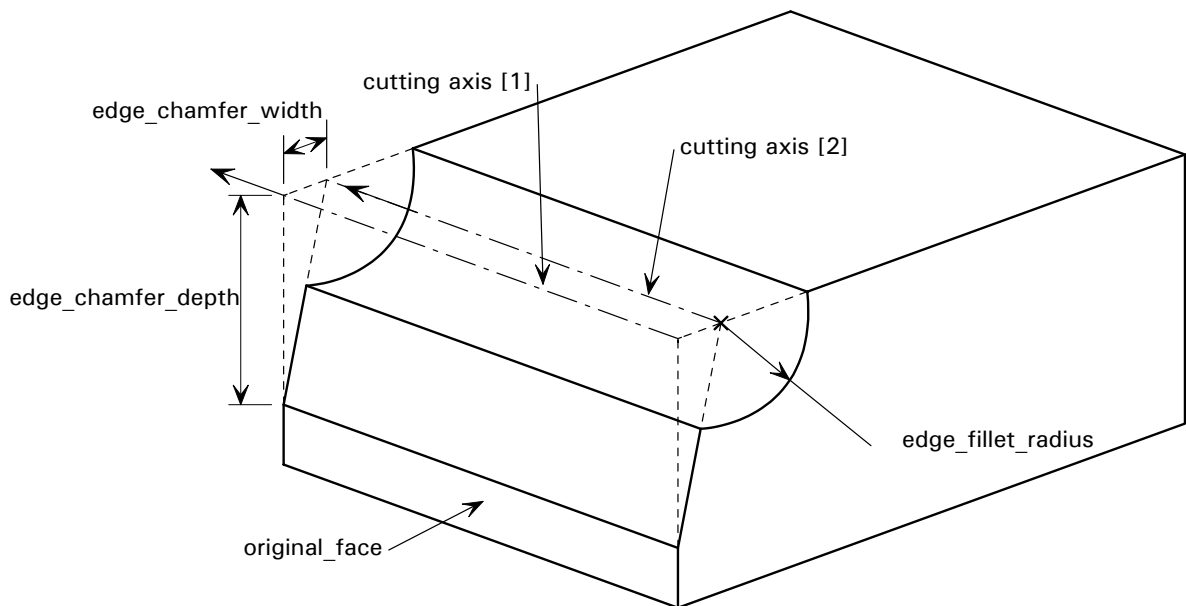
*)

```
ENTITY feature_edge_chamfer_straight
  SUBTYPE OF (feature_edge_chamfer);
    edge_chamfer_width : positive_length_measure_with_unit;
    edge_chamfer_depth : positive_length_measure_with_unit;
END_ENTITY;
```

(*)

Attribute definitions:***edge_chamfer_width***

Declares the instance of `positive_length_measure_with_unit` associated with the `feature_edge_chamfer_straight`, which is used to define the numerical value with an appropriate unit of the linear dimension of edge chamfer measured along the positive y-direction from the feature's origin. In other words, it is the distance between the original edge and the cutting line on the adjacent surface of the plate (see Figure 9.3). Where the chamfer is applied to an edge that is not a right angle, the `edge_chamfer_width` remains the dimension in the y direction, but the cutting plane is extended or reduced (as appropriate) to meet the true face.



[1] feature edge chamfer straight + [2] feature edge chamfer fillet

Figure 9.8 Example of a combination a 'straight chamfer' and a 'fillet'

edge_chamfer_depth

Declares the instance of `positive_length_measure_with_unit` associated with the `feature_edge_chamfer_straight`, which is used to define the numerical value with an appropriate unit of the linear dimension of edge chamfer measured along the positive z-direction from the feature's origin. In other words, it is the distance between the original edge and the cutting line on the adjacent surface of the plate (see Figure 9.3). Where the chamfer is applied to an edge that is not a right angle, the `edge_chamfer_depth` remains the dimension in the feature's z-direction, but the cutting plane is extended or reduced (as appropriate) to meet the true face.

Notes:

New for CIS/2; addressed by `EDGE_CHAMFER` in CIS/1.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.7 feature_surface**Entity definition:**

A type of feature that is applied to the surface of a part or assembly. The feature can be categorized (through its SUBTYPES) by the shape of its defining boundary as either

simple or complex. Other SUBTYPEs allow the feature to be a ‘name tag’ or a surface treatment. The feature can also be given a layout.

This entity would normally be instantiated as one of its SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY feature_surface
ABSTRACT SUPERTYPE OF (ONEOF
    (feature_surface_complex,
    feature_surface_point,
    feature_surface_simple) ANDOR
    feature_surface_name_tag ANDOR
    feature_surface_treatment ANDOR
    feature_surface_with_layout)
SUBTYPE OF (feature);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity feature and to collect together the SUBTYPEs under a common categorization.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instantiated on its own - it must be instantiated with one of its SUBTYPEs.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instantiated with more than one of its SUBTYPEs. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

New for CIS/2.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – SUBTYPEs added.

9.3.8 feature_surface_complex

Entity definition:

A type of surface feature whose defining boundary is itself defined by a bounded_surface – a STEP Part 42 explicit geometry construct.

EXPRESS specification:

```
*)
ENTITY feature_surface_complex
SUBTYPE OF (feature_surface);
    feature_boundary : bounded_surface;
```


END_ENTITY;

(*

Attribute definitions:

feature_boundary

Declares the instance of bounded_surface associated with the feature_surface_complex, which is used to define the boundary of the surface feature. There must be one (and only one) instance of bounded_surface referenced by each instance of feature_surface_complex.

Informal propositions:

Any surface feature that has a defined boundary must be instanced with either feature_surface_simple or feature_surface_complex. If the boundary can be defined by a series of points, then feature_surface_simple should be used. If not, then feature_surface_complex should be used.

Notes:

New for CIS/2.

The entity bounded_surface is defined in STEP Part 42.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition

9.3.9 feature_surface_name_tag

Entity definition:

A type of surface feature that is used to specify a 'name tag', 'piece mark' or 'signature', which is then applied to one or more parts. The name tag has an implicit coordinate system such that all of the text strings are defined within the positive yz-plane. The name tag is only given a position and orientation when referenced by a located_feature.

EXPRESS specification:

*)

ENTITY feature_surface_name_tag

SUBTYPE OF (feature_surface);

name_tag_items : LIST [1:?] OF text;

END_ENTITY;

(*

Attribute definitions:

name_tag_items

Specifies the list of text strings to be used as the name tag. The LIST data type implies that the order of the strings is important. There must be at least one string contained by each instance of feature_surface_name_tag.

The maximum size of the characters of the name tag is dependent upon the boundary defined in either feature_surface_simple or feature_surface_complex. If the list of text items contains only plain characters (a-z, A-Z, 0-9) the text has no mark up (the character style and font is not specified). Mark up tags may be placed within each string (within the limitations of the encoding of the communication mechanism (e.g. a STEP Part 21 file).

Each item in the list defines one line of the name tag; thus if this attribute is populated with a LIST of 3 items, there would be 3 lines in the name tag.

The name tag is defined in the positive y-z plane and projected in the positive x-direction. The lines of the name tag are oriented in the positive y-direction with the positive z direction defining the ‘up’ direction. The last line is located with its base on the z-axis.

If this entity is not instanced with either a `feature_surface_simple` or `feature_surface_complex`, the size of the characters of the name tag is defined by the mark up tags. Plain text would be unbounded, and the name tag would be insufficiently defined.

Informal propositions:

This entity may be instanced with the entity `surface_treatment_hard_stamp` (a SUBTYPE of one of its sibling SUBTYPES) to specify the process that produces the name tag.

Notes:

New for CIS/2.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.10 feature_surface_point

Entity definition:

A type of surface feature whose boundary is a point.

EXPRESS specification:

```
*)
ENTITY feature_surface_point
  SUBTYPE OF (feature_surface);
    feature_point : point;
END_ENTITY;
(*
```

Attribute definitions:

feature_point

The instance of point associated with the `feature_surface_point` used to define the limit of application of the feature.

Notes:

New for CIS/2 2nd Edition.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

9.3.11 feature_surface_point_mark

Entity definition:

A type of surface feature applied at a point and associated with a hard stamping process. This entity is intended to represent a ‘pock mark’ used for setting out

EXPRESS specification:

```
*)
```

```

ENTITY feature_surface_point_mark
SUBTYPE OF (feature_surface);
    marking_process : surface_treatment_hard_stamp;
END_ENTITY;
(*)

```

Attribute definitions:

marking_process

The instance of `surface_treatment_hard_stamp` associated with the `feature_surface_point_mark` used to define the process used to form the feature.

Notes:

New for CIS/2 2nd Edition.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

9.3.12 feature_surface_simple

Entity definition:

A type of surface feature whose defining boundary is itself defined by a set of 3 or more points. The boundary may be open or closed.

EXPRESS specification:

```

*)
ENTITY feature_surface_simple
SUBTYPE OF (feature_surface);
    feature_boundary : LIST [3:?] OF point;
END_ENTITY;
(*)

```

Attribute definitions:

feature_boundary

Declares the set of instances of point associated with the `feature_surface_simple`, which define the boundary of the surface feature. There must be at least 3 instances of point referenced by each instance of `feature_surface_simple`. If the boundary is closed, there should be 4 references in the list, with the last point coincidental with the first point. It is assumed that where the final point is not coincidental with the first point, the feature has an open boundary.

The points defining the boundary, may (but need not) lie in the same plane. If they do not, then the boundary is defined by the polyline connecting the points projected in the x-direction onto the surface of the part to which the feature is applied. In the simplest case, all the points will lie in the same plane and will be defined in the feature's yz plane. This plane is then projected the feature's x-direction onto the surface of the part to which the feature is applied.

Informal propositions:

Any surface feature that has a defined boundary must be instanced with either `feature_surface_simple` or `feature_surface_complex`. If the boundary can be defined by a series of points, then `feature_surface_simple` should be used. If not, then `feature_surface_complex` should be used.

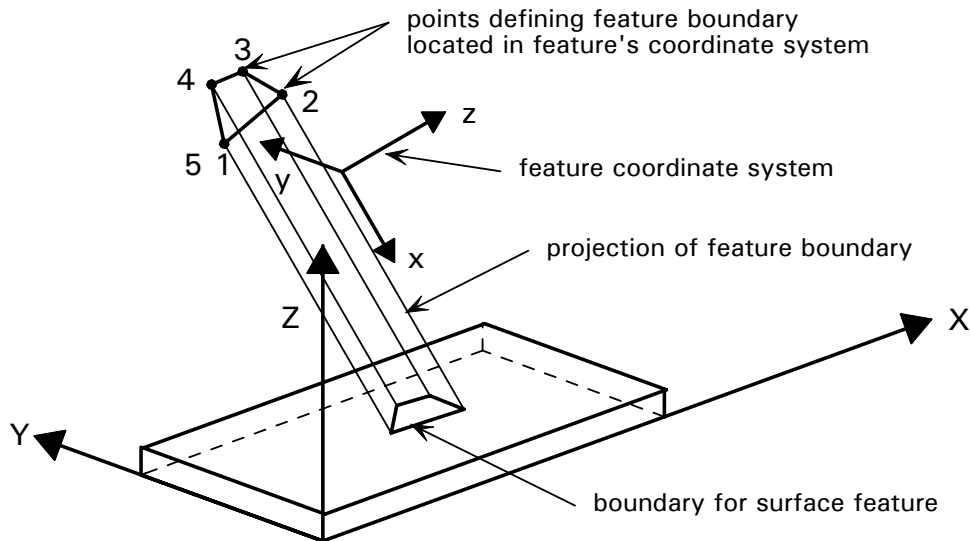


Figure 9.9 Example of a *feature_surface_simple* applied to a sheet part

Notes:

New for CIS/2.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

9.3.13 feature_surface_treatment

Entity definition:

A type of surface feature which applies a surface treatment to a part. If the surface feature has been defined with a boundary (through being instanced with the SUBTYPE *feature_surface_complex* or *feature_surface_simple*), the treatment applies only to the area lying within the boundary.

EXPRESS specification:

```
*)
ENTITY feature_surface_treatment
  SUBTYPE OF (feature_surface);
    treatment_definition : surface_treatment;
END_ENTITY;
(*)
```

Attribute definitions:

treatment_definition

Declares the instance of *surface_treatment* associated with the *feature_surface_treatment*, which define the process used to create the feature (e.g. coating or grinding). There must be one (and only one) instance of *surface_treatment* associated with each *feature_surface_treatment*.

Notes:

New for CIS/2.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.14 feature_surface_with_layout

Entity definition:

A type of surface feature that is repeated at each of the points defined within this entity. This entity will normally be instantiated with one of the other SUBTYPES of feature_surface using the ANDOR construct.

EXPRESS specification:

```
*)
ENTITY feature_surface_with_layout
  SUBTYPE OF (feature_surface);
    layout : SET [2:?] OF point;
END_ENTITY;
(*
```

Attribute definitions:

layout

Declares the set of instances of point associated with the feature_surface_with_layout, which are used to define the layout. It is assumed that all the points are defined from the same origin, and that that origin is the origin of the surface feature's own local coordinate system. There must be at least 2 separate and distinct instances of point referenced by each instance of feature_surface_with_layout.

Notes:

New for CIS/2.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.15 feature_thread

Entity definition:

A special type of feature used to define a thread for a special bolt, screw or threaded connector. The thread may be standard, but if it is not, then the thread profile for the non-standard thread must be defined explicitly. The number of threads (which can be more than one) must be declared. The thread must also be declared as being either right handed or left handed. The pitch and length of the thread must also be specified with appropriate units. Finally, the shape of the thread profile may be defined using explicit geometry constructs from the STEP Generic Resources.

If the 'fit' of the thread can be described (e.g. "close tolerance"), then this is represented by the attribute item_description inherited from structural_frame_item. If the 'fit' of the thread needs to be specified numerically, then this is represented using the constructs for tolerance (adopted from STEP Part 45).

EXPRESS specification:

```
*)
ENTITY feature_thread
  SUBTYPE OF (feature);
    thread_pitch : positive_length_measure_with_unit;
    thread_length : positive_length_measure_with_unit;
    thread_profile : OPTIONAL shape_representation_with_units;
    right_handed : BOOLEAN;
```

```

    number_of_threads : INTEGER;
WHERE
    WRF12 : number_of_threads > 0;
END_ENTITY;
(*)

```

Attribute definitions:

thread_pitch

Declares the first instance of `positive_length_measure_with_unit` associated with the `feature_thread`, which specifies the numerical value with an appropriate unit of the linear distance between sequential points on the thread.

thread_length

Declares the second instance of `positive_length_measure_with_unit` associated with the `feature_thread`, which specifies the numerical value with an appropriate unit of the linear dimension of the threaded portion of the connector.

thread_profile

Declares the instance of `shape_representation_with_units` that may be associated with the `feature_thread` to defines the explicit geometry of the 2 dimensional profile of one the threads. Where there is more than one thread, it is assumed that each of the threads shares the same 2 dimensional profile. There may be at most one instance of `shape_representation_with_units` referenced by each instance of `feature_thread`.

right_handed

Declared whether the thread is right handed (TRUE) or left handed (FALSE). While standard bolts are right handed, some special bolts are left handed.

number_of_threads

Declares the number of threads possessed by the `feature_thread`. While standard bolts are single threaded, some special bolts are double or even triple threaded, in which case, this attribute would be set to 2 or 3, respectively.

Formal propositions:

WRF12

The value assigned to the attribute `number_of_threads` shall be greater than zero. (A positive number of threads is required.)

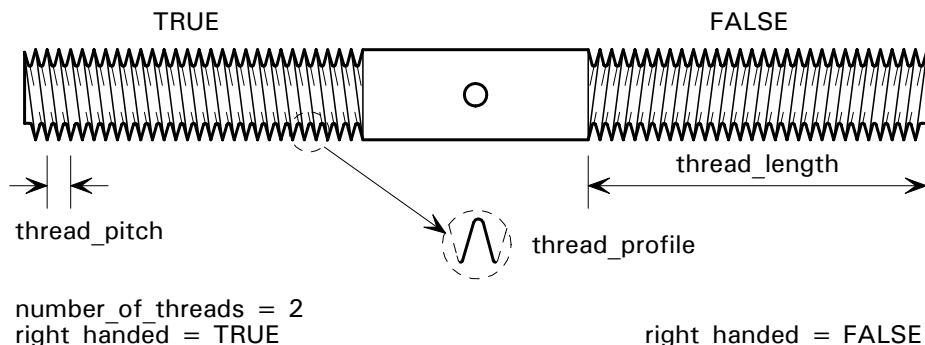


Figure 9.10 Examples of right and left-handed double threads

Notes:

New for CIS/2.

See diagram 43 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.16 feature_volume**Entity definition:**

A type of feature whose geometry is defined in 3 dimensions. Volume features always remove material from the part to which they are applied. This entity must be instanced as one of its SUBTYPES.

EXPRESS specification:

*)

ENTITY feature_volume

ABSTRACT SUPERTYPE OF (ONEOF (feature_volume_complex,
 feature_volume_hole,
 feature_volume_prismatic) ANDOR
 feature_volume_curved ANDOR
 feature_volume_with_depth ANDOR
 feature_volume_with_limit ANDOR
 feature_volume_with_layout ANDOR
 feature_volume_with_process)

SUBTYPE OF (feature);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE feature. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity, and to collect together the SUBTYPES under one category.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Informal propositions:

Volume features remove material. To add material – add a part.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – SUBTYPEs added.

9.3.17 feature_volume_complex

Entity definition:

A type of volumetric feature whose 3 dimensional geometry is defined using the explicit geometry constructs of the STEP Generic Resources.

EXPRESS specification:

```
*)
ENTITY feature_volume_complex
  SUBTYPE OF (feature_volume);
    feature_shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:

feature_shape

Declares the instance of *shape_representation_with_units* associated with the *feature_volume_complex*, which defines the explicit geometry of the 3 dimensional feature. There must be one (and only one) instance of *shape_representation_with_units* referenced by each instance of *feature_volume_complex*. Where the *feature_volume_complex* is used on its own, the dimensionality of the *geometric_representation_items* shall be = 3. The dimensionality of the *geometric_representation_items* may = 2 if the *feature_volume_complex* is used in conjunction with a *feature_volume_curved*. In this case, the *shape_representation* defines a 2 dimensional cutting edge which is projected along the curve. The cutting edge is defined in the feature's yz plane and the cutting axis is defined as the feature's x-axis, which follows the feature trace of the *feature_volume_curved*. Material is then removed between the cutting edge and the cutting axis. If the cutting edge is defined as open, then it is effectively infinite. In that case, the limit on the volume of material removed is determined by the part to which the feature is applied. If, on the other hand, the cutting edge is defined as closed, the volume of material removed by the feature is only partly determined by the part to which the feature is applied. That is, if the cutting edge lies wholly within a part, the volume of material removed is determined only by the feature.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.18 feature_volume_curved

Entity definition:

A type of volumetric feature where the removed volume of material is defined by moving a 2 dimensional cutting edge along a curve. This entity would normally be instanced with one of the other SUBTYPEs of *feature_volume* using the ANDOR SUPERTYPE construct.

EXPRESS specification:

```

*)
ENTITY feature_volume_curved
SUBTYPE OF (feature_volume);
    feature_trace : curve;
END_ENTITY;
(*

```

Attribute definitions:*feature_trace*

Declares the instance of curve associated with the feature_volume_curved, which is used to define the path of the cutting edge of the feature. The curve is defined in the coordinate system of the feature. There must be one (and only one) curve associated with each instance of feature_volume_curved. The cutting axis is defined as the feature's x-axis. Thus, the origin of the 2 dimensional cutting edge sits on the path of this cutting axis.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.19 feature_volume_curved_line**Entity definition:**

A type of feature_volume_curved, where the curve is constrained to be a line. That is, a type of volumetric feature where the removed volume of material is defined by moving a 2 dimensional cutting edge along a straight line. This entity would normally be instanced with one of the other SUBTYPES of feature_volume using the ANDOR SUPERTYPE construct.

It should be noted that a line has a position, direction and magnitude. As a straight line is a specialized curve, the entity line is a SUBTYPE of the entity curve (which is a SUBTYPE of the entity geometric_representation_item). As such, it inherits a geometric_representation_context.

EXPRESS specification:

```

*)
ENTITY feature_volume_curved_line
SUBTYPE OF (feature_volume_curved);
WHERE
    WRF13 : 'STRUCTURAL_FRAME_SCHEMA.LINE' IN TYPE OF
        (SELF\feature_volume_curved.feature_trace);
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE feature_volume_curved. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity such that curve is defined as a straight line.

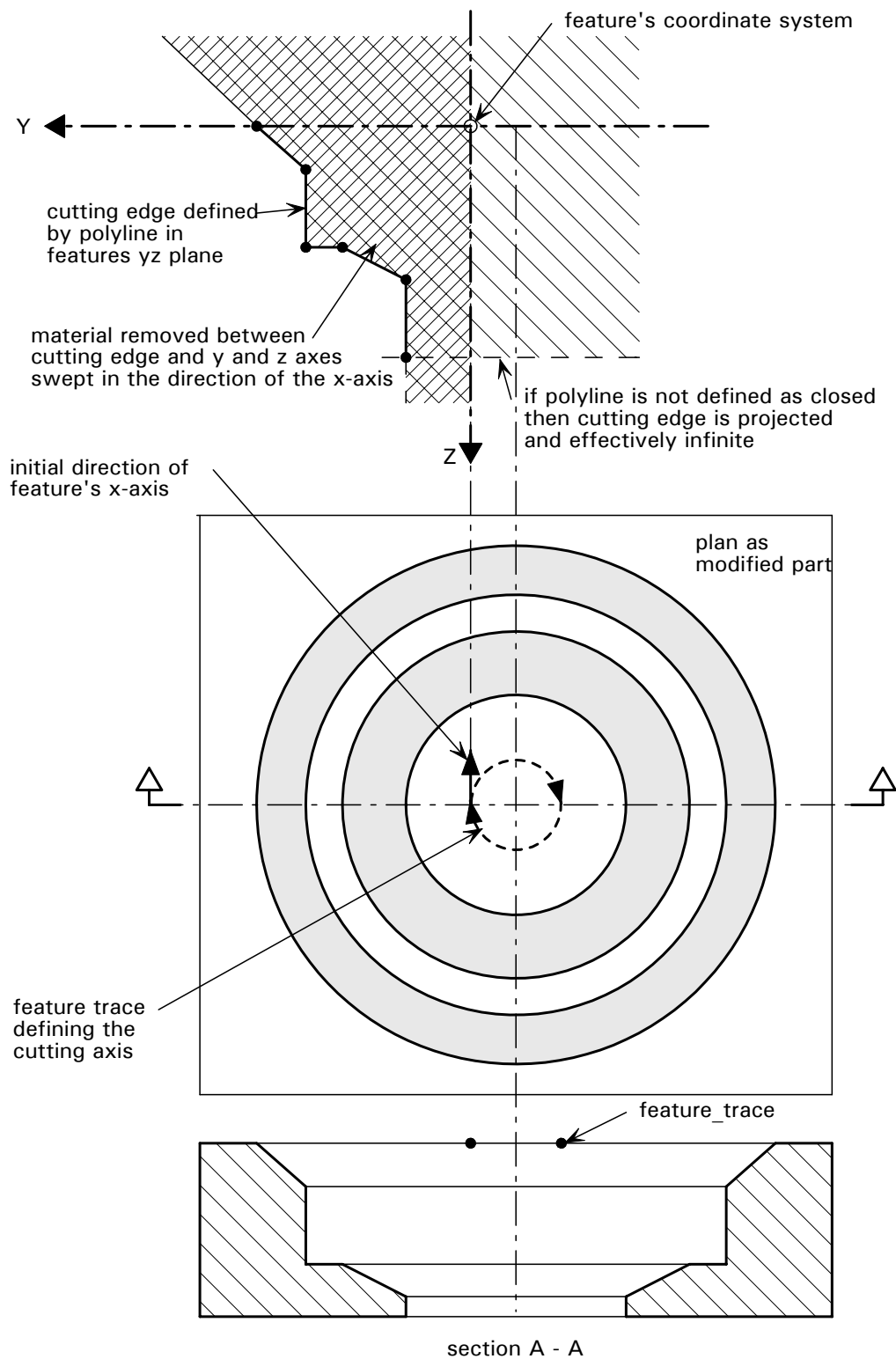


Figure 9.11 *An example of a feature_volume_curved*

Formal propositions:

WRF13

The instance of curve referenced by the attribute feature_trace (inherited from the SUPERTYPE feature_volume_type) shall be of the type line.

Notes

New for CIS/2.

See Diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.20 feature_volume_hole**Entity definition:**

A type of volumetric feature that is defined by a 2D closed cutting edge and a depth. The 2 dimensional cutting edge is defined implicitly by the attributes of the SUBTYPEs. If the feature_volume_hole **is not** associated with a feature_volume_curved, it is assumed that the hole is of unspecified depth in the cutting direction; i.e. it passes right through the part to which it is applied. If the feature_volume_hole **is** associated with a feature_volume_curved, the hole has a specified depth in the cutting direction. The depth of the hole is equal to the length of the cutting axis. In this case, the hole may or may not pass right through the part to which it is applied, depending on how the feature_trace (an attribute of the entity feature_volume_curved) is defined.

This implies that an unconstrained hole applied to the top flange of an I-beam will also pass through the bottom flange. Where the hole only passes through the top flange of the I-beam, it must be instanced with a feature_volume_curved with the curve set to a straight line with its magnitude set to the top flange thickness (or the length of the drill movement).

An unconstrained feature_volume_hole is assumed to have an unconstrained (infinite) cutting length with respect to the part to which it is applied. This applies only to the part being modified, not to adjacent parts. Where adjacent parts are modified by the same type of feature, they must be reapplied to each of the parts. The location of a feature is specified by the entity located_feature and its SUBTYPEs.

This entity must be instanced as one of its SUBTYPEs.

EXPRESS specification:

```
*)
ENTITY feature_volume_hole
ABSTRACT SUPERTYPE OF ( ONEOF
    (feature_volume_hole_circular,
    feature_volume_hole_rectangular,
    feature_volume_hole_slotted) )
SUBTYPE OF (feature_volume);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE feature_volume. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity feature_volume, and to collect together the SUBTYPEs under one common category.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2; partially addressed by HOLE in CIS/1.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.21 feature_volume_hole_circular**Entity definition:**

A type of hole (volumetric feature) whose 2 dimensional cutting edge is implicitly defined as circular with given radius. The feature's cutting axis lies at the centre of the circle. The origin of the feature's coordinate system is taken to be the centre of the circle.

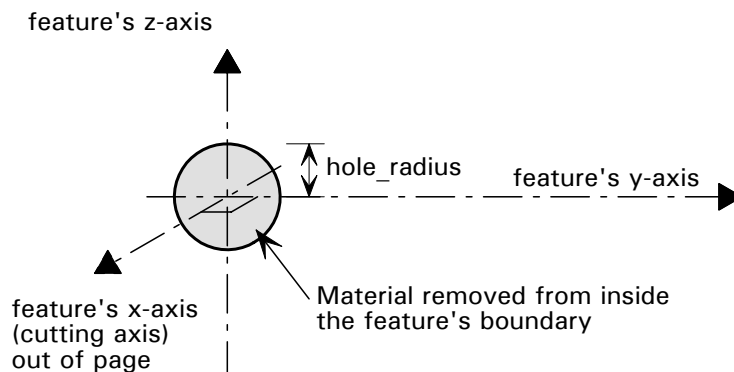


Figure 9.12 *Defining a circular hole*

EXPRESS specification:

*)

```
ENTITY feature_volume_hole_circular
  SUPERTYPE OF (feature_volume_hole_circular_threaded)
  SUBTYPE OF (feature_volume_hole);
    hole_radius : positive_length_measure_with_unit;
END_ENTITY;
```

(*

Attribute definitions:**hole_radius**

Declares the instance of `positive_length_measure_with_unit` associated with the `feature_volume_hole_circular`, which specifies the numerical value with an appropriate unit of the radius of the circular cutting edge. There must be one (and only one) instance of `length_measure_with_unit` referenced by each instance of `feature_volume_hole_circular`.

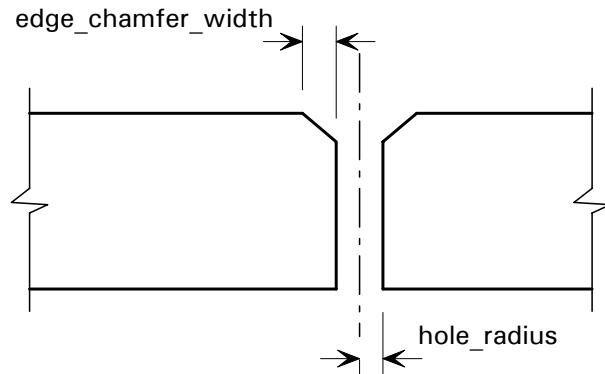


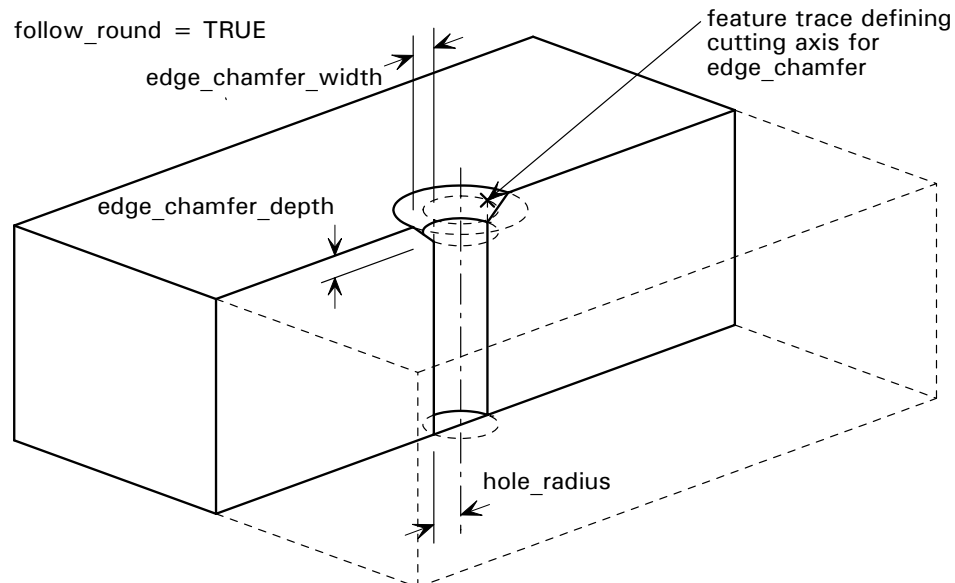
Figure 9.13 *Applying an edge chamfer to a hole*

Notes:

New for CIS/2; partially addressed by HOLE in CIS/1.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.



'countersunk hole' ≡ feature volume hole circular + edge chamfer straight

Figure 9.14 *The resulting 'countersunk hole'*

9.3.22 feature_volume_hole_circular_threaded

Entity definition:

A special type of circular hole that has an additional feature of a thread. The thread may be standard or special (non-standard), depending on the definition of the associated instance of feature_thread.

EXPRESS specification:

```
*)
ENTITY feature_volume_hole_circular_threaded
  SUBTYPE OF (feature_volume_hole_circular);
    thread_definition : feature_thread;
END_ENTITY;
```

(*)

Attribute definitions:***thread_definition***

Declares the instance of `feature_thread` associated with the `feature_volume_hole_circular_threaded`, which specifies the form and type of the thread. There must be one (and only one) instance of `feature_thread` associated with each instance of `feature_volume_hole_circular_threaded`.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.23 feature_volume_hole_rectangular**Entity definition:**

A type of hole (volumetric feature) whose 2 dimensional cutting edge is implicitly defined as rectangular with a given length and height. The cutting edge is defined in the feature's yz plane. The feature's cutting axis is defined as the feature's x-axis, which effectively lies in one corner of the rectangle. The rectangle may also be given a fillet radius. If the hole has square corners, it is given a fillet radius of 0.0. The cutting edge of the feature is defined within the positive y and positive z-axes of the feature's coordinate system.

EXPRESS specification:

*)

ENTITY `feature_volume_hole_rectangular`

SUBTYPE OF (`feature_volume_hole`);

`hole_length` : `positive_length_measure_with_unit`;

`hole_height` : `positive_length_measure_with_unit`;

`fillet_radius` : OPTIONAL `positive_length_measure_with_unit`;

DERIVE

`fillet_radius_value` : REAL := NVL(`fillet_radius.value_component`, 0.0);

`hole_length_value` : REAL := `hole_length.value_component`;

`hole_height_value` : REAL := `hole_height.value_component`;

WHERE

 WRF14 : `fillet_radius_value` < (`hole_length_value`/2);

 WRF15 : `fillet_radius_value` < (`hole_height_value`/2);

END_ENTITY;

(*)

Attribute definitions:***hole_length***

Declares the first instance of `positive_length_measure_with_unit` associated with the `feature_volume_hole_rectangular`, which is used to specify the numerical value with an appropriate unit of the linear dimension of the rectangular hole in the y direction. (See Figure 9.15.)

hole_height

Declares the second instance of `positive_length_measure_with_unit` associated with the `feature_volume_hole_rectangular`, which is used to specify the numerical value with an appropriate unit of the linear dimension of the rectangular hole in the z direction. (See Figure 9.15.)

fillet_radius

Declares the third instance of `positive_length_measure_with_unit` that may be associated with the `feature_volume_hole_rectangular` to specify the numerical value with an appropriate unit of the fillet radius that reduces the corners of the rectangular hole. (See Figure 9.15.)

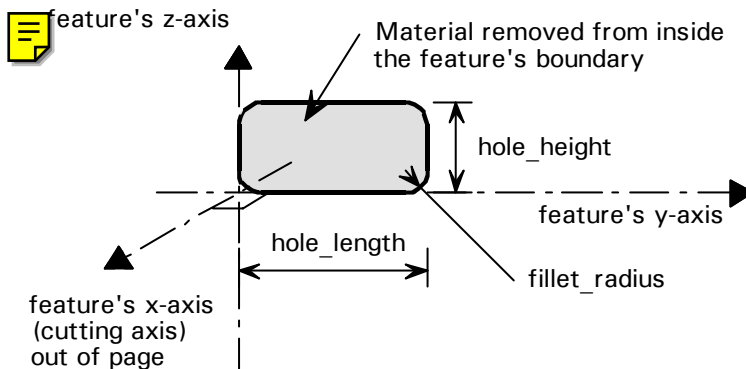


Figure 9.15 *Defining a rectangular hole*

fillet_radius_value

This attribute derives the real number value of the fillet radius from the instance of the `positive_length_measure_with_unit` referenced by the attribute `fillet_radius`. If the attribute `fillet_radius` has been assigned a NULL value, the `fillet_radius_value` is assigned the value 0.0. When this entity is instantiated in a STEP Part 21 file, this attribute does not appear in the encoded exchange structure. This attribute is checked in the WHERE rules.

hole_length_value

This attribute derives the real number value of the hole length from the instance of the `positive_length_measure_with_unit` referenced by the attribute `hole_length`. When this entity is instantiated in a STEP Part 21 file, this attribute does not appear in the encoded exchange structure. This attribute is checked in the WHERE rule WRF14.

hole_height_value

This attribute derives the real number value of the hole height from the instance of the `positive_length_measure_with_unit` referenced by the attribute `hole_height`. When this entity is instantiated in a STEP Part 21 file, this attribute does not appear in the encoded exchange structure. This attribute is checked in the WHERE rule WRF15.

Formal propositions:*DERIVE*

The derived attributes `fillet_radius_value`, `hole_length_value`, and `hole_height_value` do not appear in the STEP Part 21 file.

WRF14

The value of the derived attribute `fillet_radius_value` shall be less than one half of the value derived by the attribute `hole_length_value`. In other words, the fillet radius must be less than half the hole length.

WRF15

The value of the derived attribute `fillet_radius_value` shall be less than one half of the value derived by the attribute `hole_height_value`. In other words, the fillet radius must be less than half the hole height.

Informal propositions:

All of the dimensions defining the feature are measured in the same units.

For square cornered holes, the `fillet_radius` shall be set to zero.

For square holes, the `hole_length` shall be set equal to the `hole_height`.

For square cornered square holes, the `hole_length` shall be set equal to the `hole_height`, and the `fillet_radius` shall be set to zero.

Notes:

New for CIS/2; partially addressed by HOLE in CIS/1.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.24 feature_volume_hole_slotted**Entity definition:**

A type of hole (volumetric feature) whose 2 dimensional cutting edge is implicitly defined as a slot (defined here as an elongated circle) with a given slot length and slot height. The cutting edge is defined in the feature's yz plane. The feature's cutting axis is defined as the feature's x-axis.

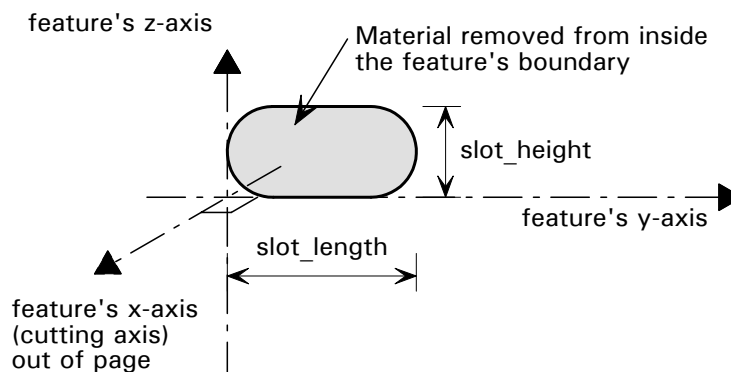


Figure 9.16 *Defining a slotted hole*

EXPRESS specification:

*)

ENTITY `feature_volume_hole_slotted`

SUPERTYPE OF (`feature_volume_hole_slotted_curved`)

SUBTYPE OF (`feature_volume_hole`);

`slot_height` : `positive_length_measure_with_unit`;

`slot_length` : `positive_length_measure_with_unit`;

WHERE

WRF16 : slot_length.value_component > slot_height.value_component;

END_ENTITY;

(*

Attribute definitions:

slot_height

Declares the first instance of positive_length_measure_with_unit associated with the feature_volume_hole_slotted, which is used to specify the numerical value with an appropriate unit of the linear dimension of the rectangular hole in the z direction. (See Figure 9.16.)

slot_length

Declares the second instance of positive_length_measure_with_unit associated with the feature_volume_hole_slotted, which is used to specify the numerical value with an appropriate unit of the linear dimension of the rectangular hole in the y direction. Note that because the slot is defined as having semi-circular ends, the slot_length must be greater than the slot_height. (See Figure 9.16.)

Formal propositions:

WRF16

The numerical value of the instance of positive_length_measure_with_unit referenced by the attribute slot_length shall be greater than the numerical value of the instance of positive_length_measure_with_unit referenced by the attribute slot_height. In other words, the slot length shall be greater than the slot height.

Informal propositions:

All of the dimensions defining the feature are measured in the same units.

Notes:

New for CIS/2; partially addressed by HOLE in CIS/1. The slotted hole dimensions are defined differently in CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.25 feature_volume_hole_slotted_curved

Entity definition:

A type of slotted hole that is defined by rolling a circle on an arc. This is also known as a ‘kidney shaped slot’ (See Figure 9.17). The hole is defined in the feature’s yz-plane, while the x-axis defines the ‘cutting’ axis for the hole.

EXPRESS specification:

*)

ENTITY feature_volume_hole_slotted_curved

SUBTYPE OF (feature_volume_hole_slotted);

curve_radius : positive_length_measure_with_unit;

sector_angle : plane_angle_measure_with_unit;

DERIVE

slot_radius : REAL :=

(SELF.feature_volume_hole_slotted.slot_height.value_component)/2.0;

END_ENTITY;

(*

Attribute definitions:

curve_radius

Declares the instance of `positive_length_measure_with_unit` associated with the `feature_volume_hole_slotted_curved`, which provides the numerical value with an appropriate unit of radius of the curve (measured from the feature's origin). The circular hole that defines the feature is centred on this curve and its centre is swept along the curve to form the kidney shaped slot. (See Figure 9.17.)

sector_angle

Declares the instance of `plane_angle_measure_with_unit` associated with the `feature_volume_hole_slotted_curved` which provides the numerical value with an appropriate unit of angle of the sector subtended by the curve, measured from the feature's z-axis. (See Figure 9.17.)

slot_radius

This attribute derives the numerical value of the slot radius from the instance of `positive_length_measure_with_unit` referenced by the attribute `slot_height` (inherited from the SUPERTYPE `feature_volume_hole_slotted`). The slot radius is set equal to half of the slot height. When this entity is instantiated in a STEP Part 21 file, this attribute does not appear in the encoded exchange structure.

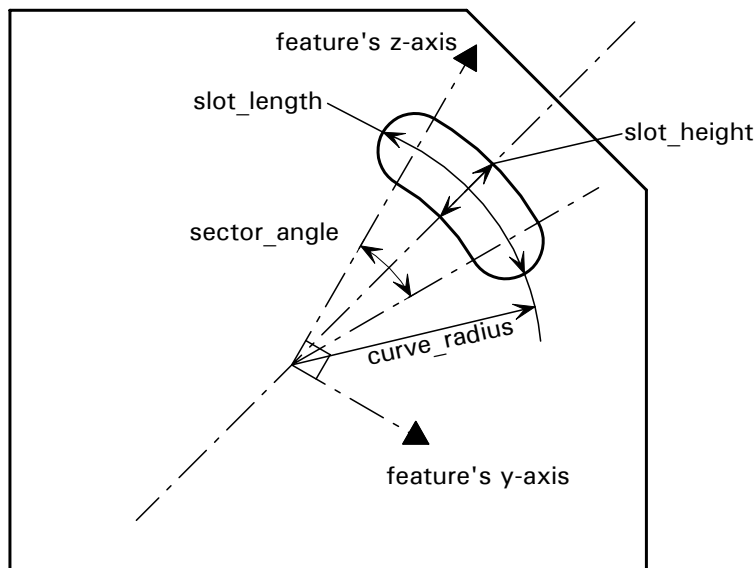


Figure 9.17 *Defining a curved slotted hole*

Formal propositions:

DERIVE

The derived attribute `slot_radius` does not appear in the STEP Part 21 file.

Informal propositions:

The dimensions used to define this feature are measured using the same units.

This entity could be used in conjunction with an instance of `feature_volume_hole_circular` to define the pair of holes required for a plate used in a simple connection for wind bracing.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.26 feature_volume_prismatic**Entity definition:**

A type of volumetric feature specifically defined for use at the ends of prismatic parts using parameters, rather than explicit geometry. This entity must be instantiated as one of its SUBTYPES, which encompass the typical features applied to the ends of prismatic parts. The purpose of these prismatic end feature is to define a basic set of commonly used features.

As prismatic parts will (typically) be defined as having a Cartesian coordinate system, so features applied to the ends of prismatic parts will (typically) be defined using Cartesian coordinate systems. Furthermore, the feature's (child) coordinate system will be located within the part's (parent) coordinate system when referenced by an instance of located_feature_for_located_part.

The feature is defined such that it may be applied at the start face of the part (where $x = 0.0$ in the part's coordinate system), or at the end face of the part (where x has its maximum value in the part's coordinate system). The feature is also defined such that it may be applied at the top edge of the part (where z has its maximum positive value in the part's coordinate system), or at the bottom edge of the part (where z has its maximum negative value in the part's coordinate system).

Prismatic end features may be applied in sequence; for example, the end of a beam may be cut at an angle - using a skewed end - and then a notch may be cut, whose trace (defining the cutting line) coincides with one of the new edges of the skew cut part.

EXPRESS specification:

```

*)
ENTITY feature_volume_prismatic
ABSTRACT SUPERTYPE OF ( ONEOF
    (feature_volume_prismatic_chamfer,
    feature_volume_prismatic_flange_notch,
    feature_volume_prismatic_flange_chamfer,
    feature_volume_prismatic_notch,
    feature_volume_prismatic_skewed_end))
SUBTYPE OF (feature_volume);
    top_or_bottom_edge : top_or_bottom;
    start_or_end : start_or_end_face;
    original_face : BOOLEAN;
END_ENTITY;
(*

```

Attribute definitions:**top_or_bottom_edge**

Declares the classification of the prismatic feature as either placed at the top or bottom edge of the part to which it is applied. The top edge corresponds to the maximum positive value on the part's local z axis, while the bottom edge corresponds to the

maximum negative value on the part's local z axis. The definitions of top and bottom remain valid regardless of which end of the part the feature applies.

start_or_end

Declares the classification of the prismatic feature as either placed at the start face or the end face of the part to which it is applied. The start face corresponds to the minimum value on the part's local x axis ($x = 0.0$), while the end face corresponds to the maximum positive value on the part's local x axis ($x = \text{max positive}$).

original_face

Declaring whether the prismatic feature is applied to the original (i.e. square) end of the prismatic part (TRUE), or applied to the residual edge after the end has been cut skew (FALSE).

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

Known as PRISMATIC_END_FEATURE in CIS/1.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.27 feature_volume_prismatic_chamfer

Entity definition:

A type of feature_volume_prismatic that may be used to define a chamfer on the end of a prismatic part. A Chamfer (for the purposes of the LPM) is a plane cut on the end of the prismatic part. It removes a volume of material from the end of a prismatic part. Here, the location of the chamfer is defined by an edge on the face of the part. The chamfer is initially defined in the yz-plane of the feature coordinate system. The cutting axis – defined as the x-axis of the feature coordinate system - is taken to be one of the edges of either the start face or end face of the prismatic part (i.e. yz-plane of the part's coordinate system). See Figure 9.18.

The Chamfer is defined such that waste material is removed from the area between the cutting edge and the y and z axes.

EXPRESS specification:

*)

ENTITY feature_volume_prismatic_chamfer

SUBTYPE OF (feature_volume_prismatic);

 chamfer_length : positive_length_measure_with_unit;

 chamfer_depth : positive_length_measure_with_unit;

END_ENTITY;

(*)

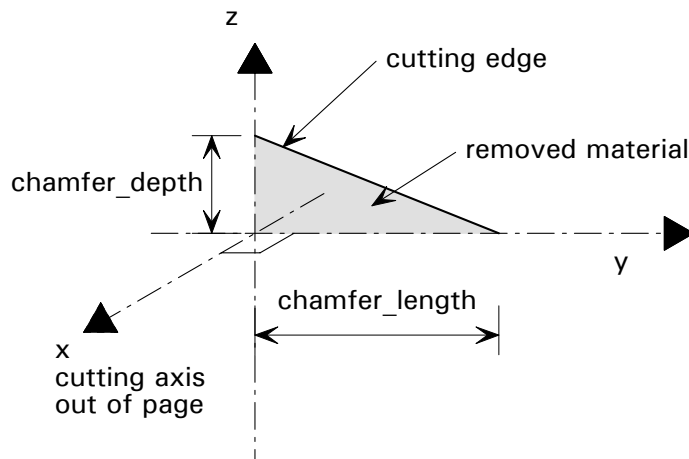


Figure 9.18 *Defining a chamfer (prismatic volume feature)*

Attribute definitions:

chamfer_length

Declares the first instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_chamfer`, which provides the numerical value with an appropriate unit of the length of the chamfer along the feature's y-axis (and thus, the x-axis of the prismatic part), in a positive direction. (See Figure 9.18.)

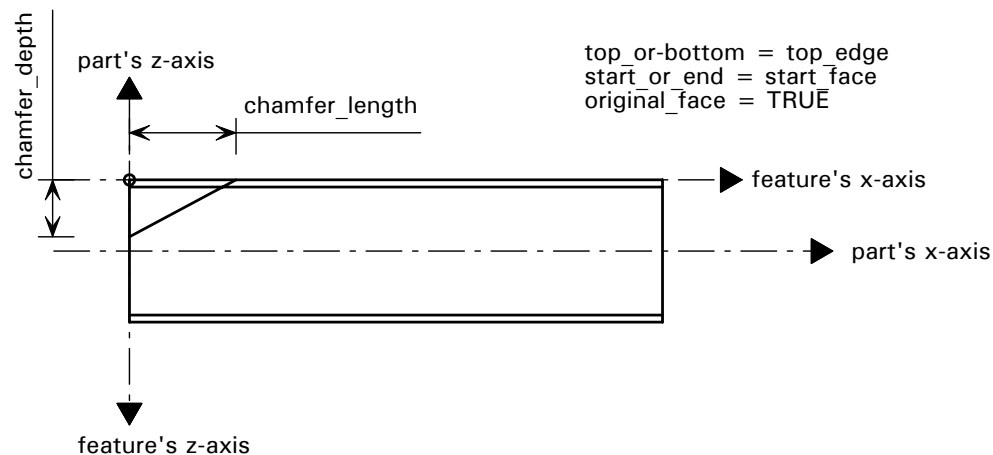


Figure 9.19 *A chamfer applied to the start face of a part*

chamfer_depth

Declares the second instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_chamfer`, which provides the numerical value with an appropriate unit of the depth of the chamfer along the feature's z-axis (and thus the y-axis of the prismatic part), in the positive z-direction. (See Figure 9.18.) If the chamfer is applied to a beam that has a skewed end, the `chamfer_depth` remains the dimension in the z-axis, but the chamfer cut is extended or reduced along the same plane, as appropriate (see Example 2 in Figure 9.20).

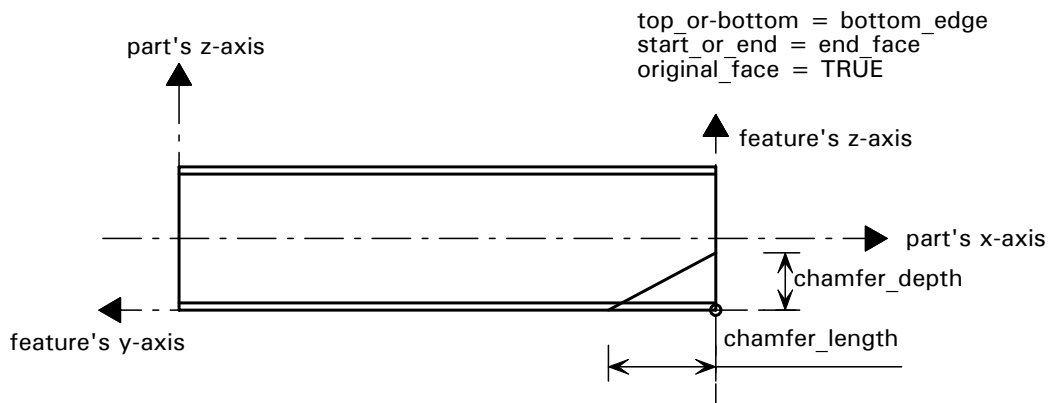


Figure 9.20 A chamfer applied to the end face of a part

Notes:

Known as CHAMFER in CIS/1.

See diagram 45 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.28 feature_volume_prismatic_flange_chamfer

Entity definition:

A type of feature_volume_prismatic that may be used to define a chamfer on the flange of a prismatic part. A chamfer (for the purposes of the LPM) is a plane cut on the flange of the prismatic part. It removes a volume of material from the flange of a prismatic part. Here, the cutting axis of the chamfer is defined by an edge on the flange of the part. The chamfer's cutting edge is initially defined as a 2D cutting edge in the yz-plane of the feature's coordinate system. The cutting axis – defined as the x-axis of the feature's coordinate system – is taken to be either the left or right edge of either the start face or end face of the prismatic part. If the flange chamfer is applied to the original face of the prismatic part, then, by definition, the cutting axis will lie parallel to the z-axis of the prismatic part.

The instance of feature_volume_prismatic_flange_chamfer is applied to one flange at a time. If both top and bottom flanges are chamfered (on one side only), then two instances of feature_volume_prismatic_flange_chamfer are required. If both top and bottom flanges are chamfered on both left and right hand sides, then four instances are required.

EXPRESS specification:

*)

ENTITY feature_volume_prismatic_flange_chamfer

SUBTYPE OF (feature_volume_prismatic);

left_or_right_hand : left_or_right;

flange_chamfer_length : positive_length_measure_with_unit;

flange_chamfer_width : positive_length_measure_with_unit;

END_ENTITY;

(*

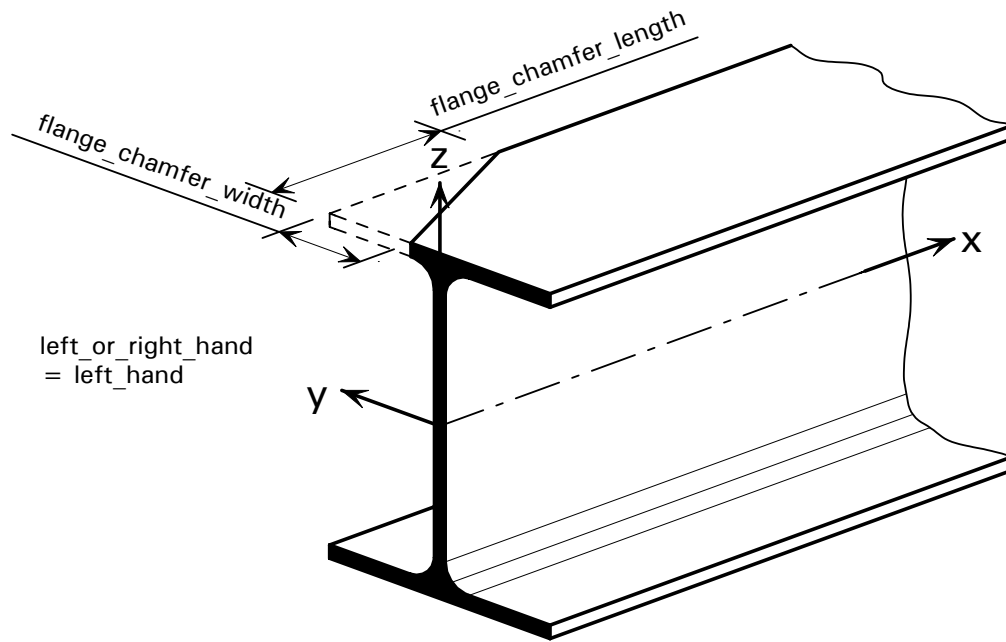


Figure 9.21 Defining a 'flange chamfer'

Attribute definitions:

left_or_right_hand

Declares the classification of the prismatic feature as either placed at the left or right hand side of the part to which it is applied. The left hand side corresponds to the maximum positive value on the part's local y-axis, while the right hand side corresponds to the maximum negative value on the part's local y-axis. The definitions of left and right remain valid regardless of which end of the part the feature applies.

flange_chamfer_length:

Declares the first instance of *positive_length_measure_with_unit* associated with the *feature_volume_prismatic_flange_chamfer*, which provides the numerical value with an appropriate unit of the length of the chamfer along the feature's y-axis (and thus, the x-axis of the prismatic part), in a positive direction. If the chamfer is applied to a beam that has a skewed end, the *flange_chamfer_length* remains the dimension in the feature's y-axis, but the chamfer cut is extended or reduced along the same plane, as appropriate.

flange_chamfer_width:

Declares the second instance of *positive_length_measure_with_unit* associated with the *feature_volume_prismatic_flange_chamfer*, which provides the numerical value with an appropriate unit of the depth of the chamfer along the feature's z-axis (and thus the y-axis of the prismatic part), in the positive z-direction. If the chamfer is applied to a beam that has a skewed end, the *flange_chamfer_width* remains the dimension in the feature's z-axis, but the chamfer cut is extended or reduced along the same plane, as appropriate.

Notes

New for CIS/2.

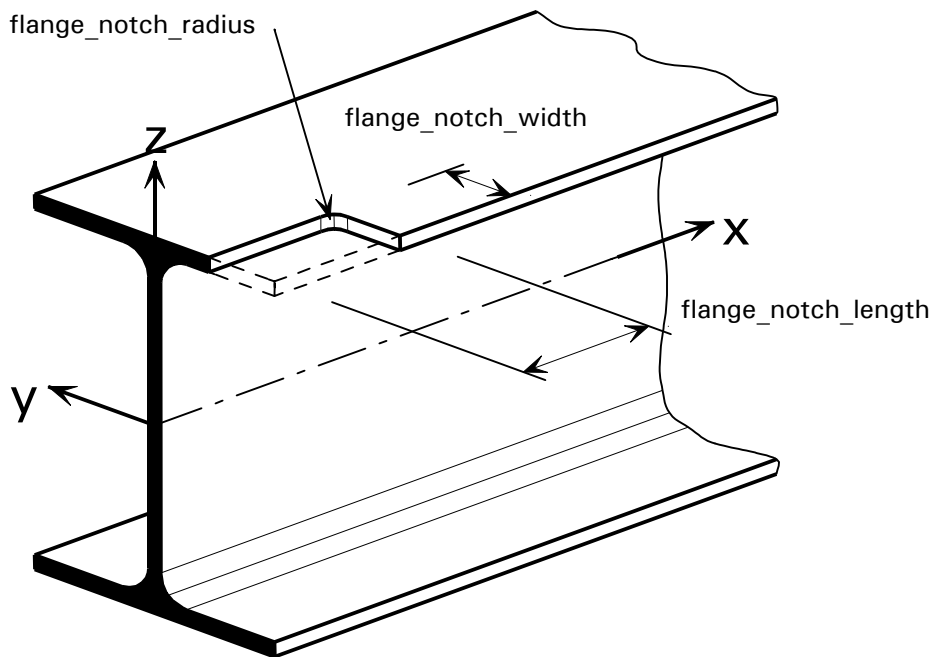
See Diagram 45 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.29 feature_volume_prismatic_flange_notch

Entity definition:

A type of feature_volume_prismatic that may be used to define a notch on the flange of a prismatic part. A notch removes a volume of material from the flange of a prismatic part. Here, the cutting axis of the notch is defined by an edge on the flange of the part. The notch is initially defined as a 2D cutting edge in the yz-plane of the feature coordinate system. The cutting direction – defined as the x-axis of the feature coordinate system – is taken to be either the left or right edges of either the start face or end face of the prismatic part. If the flange notch is applied to the original face of the prismatic part, then, by definition, the cutting axis will lie parallel to the z-axis of the prismatic part.



top_or_bottom = top_edge
 start_or_end = start_end
 original_face = true

Figure 9.22 Defining a 'flange notch'

EXPRESS specification:

```
*)
ENTITY feature_volume_prismatic_flange_notch
SUBTYPE OF (feature_volume_prismatic);
  left_or_right_hand : left_or_right;
  flange_notch_length : positive_length_measure_with_unit;
  flange_notch_width : positive_length_measure_with_unit;
  flange_notch_radius : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

left_or_right_hand

Declares the classification of the prismatic feature as either placed at the left or right hand side of the part to which it is applied. The left hand side corresponds to the

maximum positive value on the part's local y axis, while the right hand side corresponds to the maximum negative value on the part's local y-axis. The definitions of left and right remain valid regardless of which end of the part the feature applies.

flange_notch_length:

Declares the first instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_flange_notch`, which provides the numerical value with an appropriate unit of the length of the notch along the feature's y-axis (and thus, the x-axis of the prismatic part), in a positive direction. (See Figure 9.22.) If the notch is applied to a beam that has a skewed end, the `flange_notch_length` remains the dimension in the feature's y-axis, but the notch cut is extended or reduced along the same plane, as appropriate.

flange_notch_width:

Declares the second instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_flange_notch`, which provides the numerical value with an appropriate unit of the depth of the notch along the feature's z-axis (and thus the y-axis of the prismatic part), in the positive z-direction. (See Figure 9.22.) If the notch is applied to a beam that has a skewed end, the `flange_notch_width` remains the dimension in the feature's z-axis, but the notch cut is extended or reduced along the same plane, as appropriate.

flange_notch_radius:

Declares the third instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_flange_notch`, which provides the numerical value with an appropriate unit of the corner (fillet) radius of the notch.

Notes

New for CIS/2.

See Diagram 45 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.30 feature_volume_prismatic_notch

Entity definition:

A type of `feature_volume_prismatic` that may be used to define a notch on the end of a prismatic part. The notch is defined such that the waste material is removed from the area between the cutting edge and the feature's y and z-axes. The cutting axis – defined as the x-axis of the feature's coordinate system – is a straight line coinciding with either the top or bottom edge of either the start or end face of the prismatic part. If the `original_face` attribute (inherited from the SUPERTYPE `feature_volume_prismatic`) is set to TRUE, then the cutting axis lies parallel to the part's yz plane.

EXPRESS specification:

```
*)
ENTITY feature_volume_prismatic_notch
SUBTYPE OF (feature_volume_prismatic);
    notch_length : positive_length_measure_with_unit;
    notch_depth : positive_length_measure_with_unit;
    notch_radius : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
(*
```

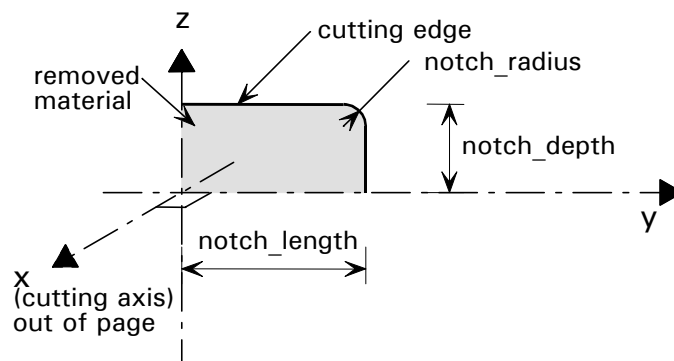


Figure 9.23 *Defining a notch (prismatic volume feature)*

Attribute definitions:

notch_length

Declares the first instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_notch`, which provides the numerical value with an appropriate unit of the length of the notch along the feature's y-axis (and, if the notch is applied to the original face, the x-axis of the prismatic part), in a positive direction. If the notch is applied to a beam that has a skewed end, the `notch_length` remains the dimension in the feature's y-axis, but the notch cut is extended or reduced along the same plane, as appropriate. The `notch_length` and `notch_depth` attribute essentially provide the proportions for the cut. (See Figure 9.23 and Figure 9.25.)

notch_depth

Declares the second instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_notch`, which provides the numerical value with an appropriate unit of the depth of the notch measured along the feature's z-axis (and, if the notch is applied to the original face, the prismatic part's z-axis). If the notch is applied to a beam that has a skewed end, the `notch_depth` remains the dimension in the feature's z-axis, but the notch cut is extended or reduced along the same plane, as appropriate. The `notch_length` and `notch_depth` attribute essentially provide the proportions for the cut. (See Figure 9.23 and Figure 9.25.)

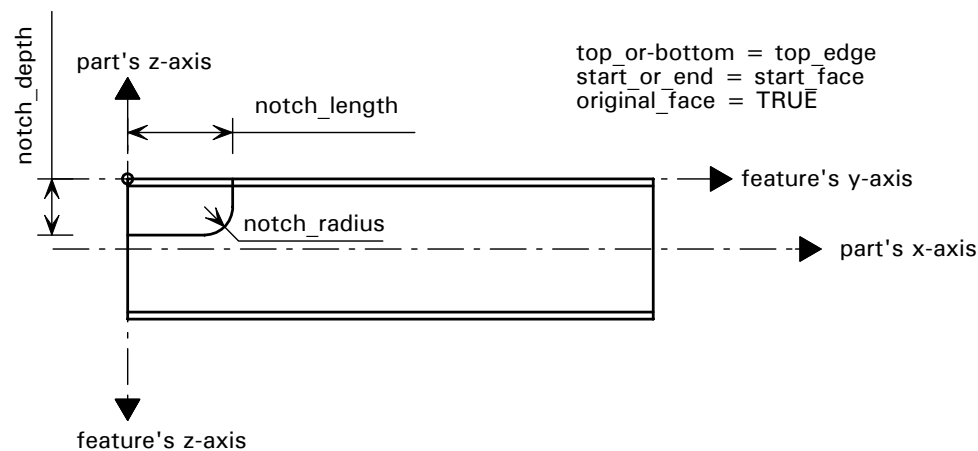


Figure 9.24 *Applying a notch to a part*

notch_radius

Declares the third instance of `positive_length_measure_with_unit` associated with the `feature_volume_prismatic_notch`, which provides the numerical value with an appropriate unit of the corner (fillet) radius of the notch. (See Figure 9.23.)

Notes:

Known as NOTCH in CIS/1.

See diagram 45 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.31 feature_volume_prismatic_skewed_end**Entity definition:**

A type of `feature_volume_prismatic` that may be used to define the skewed end of a prismatic part. This entity describes the geometry and location of a cutting plane that is used to cut the end of a prismatic part at an angle to the part's y and z-axes.

The feature is defined as an infinite plane. Thus, there is no definitive cutting axis or cutting edge; rather, this feature uses a 'cutting plane'. This entity may be viewed as a special case of the entity `feature_cutting_plane`. Here, the feature's coordinate system is implied rather than explicitly provided. The feature's coordinate system may be derived from the definition of the skew angles and the values of the attributes inherited from the SUPERTYPE `feature_volume_prismatic`.

The feature is defined such that the cutting plane lies in the feature's yz plane and material is removed from the positive x side of the plane. The cutting plane that creates the skewed end is located such that at least one vertex of the original face of the prismatic part lies on that plane. Where an end is cut skew both in plan and elevation, only one vertex of the original face will remain as original. Where an end is cut skew in one aspect only, two vertices and the edge connected them will remain as original.

Where a non-zero value is assigned to the attribute `skew_angle_1`, the end is cut skew in elevation only. Where a non-zero value is assigned to the attribute `skew_angle_2`, the end is cut skew in plan only.

Where non-zero values are assigned to the attributes `skew_angle_1` and `skew_angle_2`, the end is cut skew both in plan and elevation. In this case, the values assigned to these attributes do not represent the true angle between the planes, but rather (more conveniently), angles relative to a reference edge. (See Figure 9.27 and Figure 9.28.) These angles may be thought of as the angle through which the original end is rotated to form the skewed end.

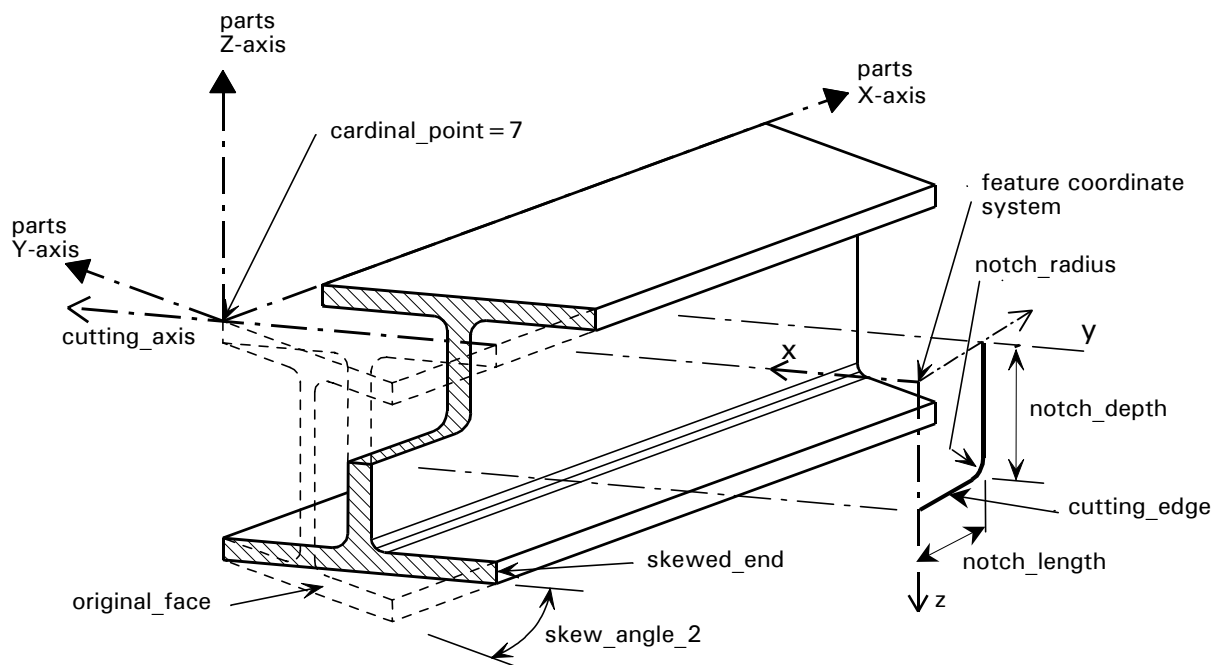
EXPRESS specification:

```
*)
ENTITY feature_volume_prismatic_skewed_end
SUBTYPE OF (feature_volume_prismatic);
    skew_angle_1 : plane_angle_measure_with_unit;
    skew_angle_2 : plane_angle_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:***skew_angle_1***

Declares the first instance of `plane_angle_measure_with_unit` associated with the `feature_volume_prismatic_skewed_end`, which provides the numerical value with an appropriate unit of the angle between the plane of the resulting skewed cut end of the prismatic part, and the part's yz plane. The convention adopted is clockwise rotation about the y axis is taken as positive. Thus, for a skewed end applied to the start face, where the `skew_angle_1` is positive, the bottom edge (where z has its maximum negative value in the part's coordinate system) would be as original. For a skewed end at the end face, a positive value would mean that the top edge would be as original. (See Figure 9.26.)

Where an end is cut skew both in plan and elevation, `skew_angle_1` is not a true angle between the planes, but rather, the angle between the original edge and the skewed edge measured in the vertical plane about the vertex that remains as original. (See Figure 9.27 and Figure 9.28.)

**Skewed End**

original_face = TRUE
 start_or_end = start face
 top_or_bottom = bottom_edge

Notch

original_face = FALSE
 start_or_end =
 top_or_bottom = top_edge

Figure 9.25 *Combining a notch and a skewed end*

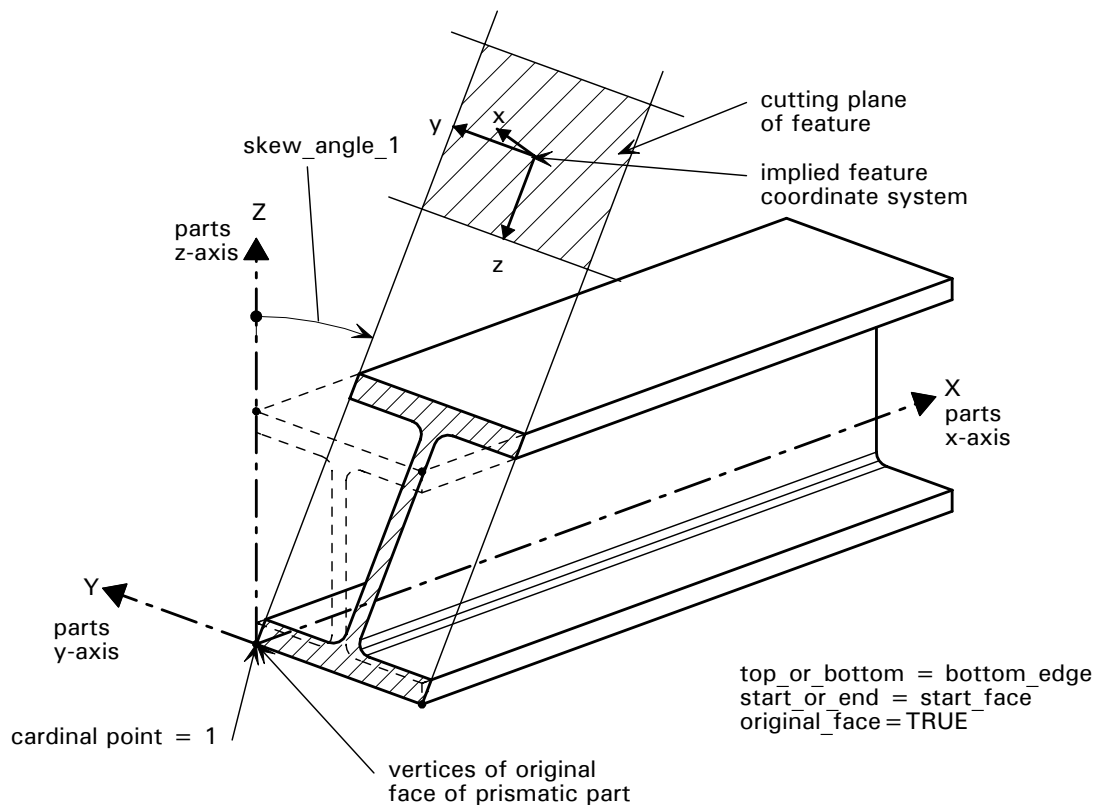


Figure 9.26 *Applying a skewed end to a part (Example 1)*

skew_angle_2

Declares the second instance of `plane_angle_measure_with_unit` associated with the `feature_volume_prismatic_skewed_end`, which provides the numerical value with an appropriate unit of the angle between of the plane of the resulting skewed cut end of the prismatic part, and the part's yz plane. The convention adopted is clockwise rotation about the z axis is taken as positive. Thus, for a skewed end applied to the start face, where the `skew_angle_2` is positive, the left edge (where y has its maximum positive value in the part's coordinate system) would be as original. (See Figure 9.27 and Figure 9.28.)

Where an end is cut skew both in plan and elevation, `skew_angle_2` is not a true angle between the planes, but rather, the angle between the original edge and the skewed edge measured in the horizontal plane about the vertex that remains as original. (See Figure 9.27 and Figure 9.28.)

Notes:

Known as `SKEWED_END` in CIS/1.

See diagram 45 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

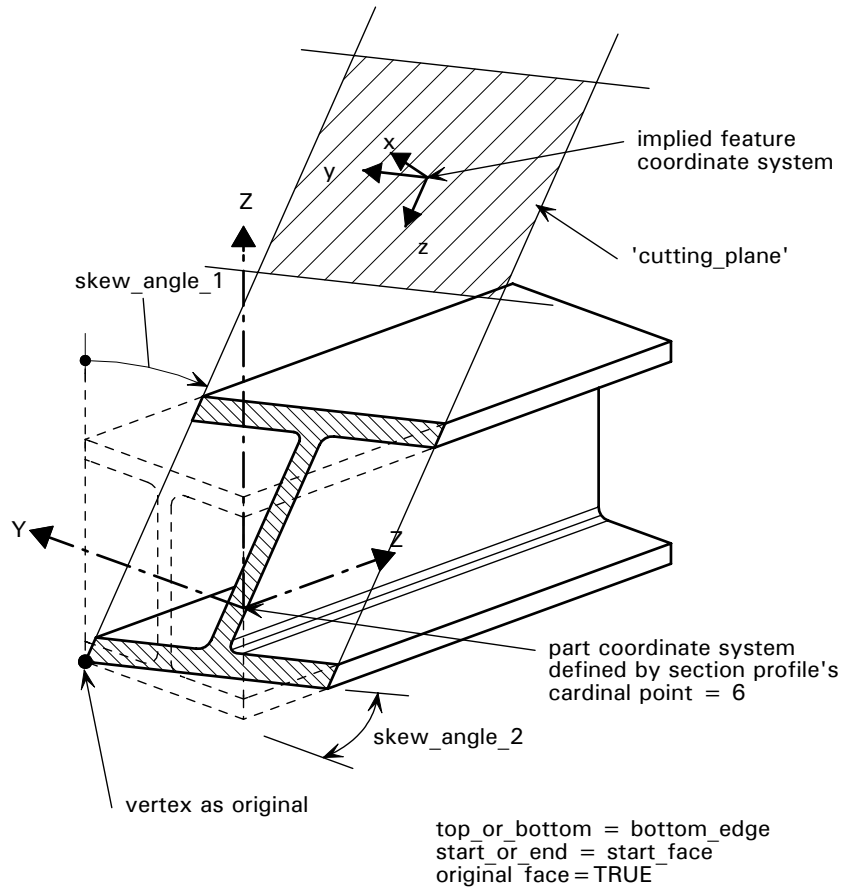


Figure 9.27 Applying a doubly skewed end to a part

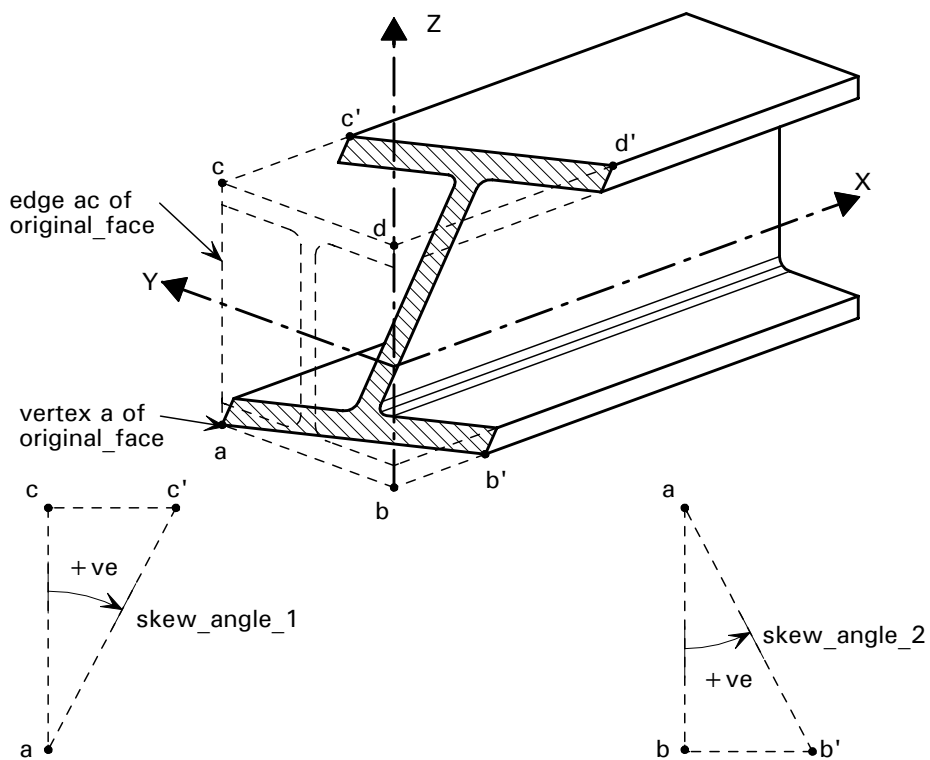


Figure 9.28 Defining the skew angles for a doubly skewed end

9.3.32 feature_volume_with_depth

Entity definition:

A type of volumetric feature that is associated with a positive_length_measure_with_unit. Volumetric features defined using the entity feature_volume are normally assumed to have no limits to extent of their application. Thus, if a hole is applied to the top flange of an I-section beam, it would normally be assumed that the hole passes through the bottom flange of the section if the cutting axis is so arranged. This entity should be instanced with another SUBTYPE of feature_volume to restrict the depth of penetration of that feature. It may also be used to represent a 'blind' hole.

EXPRESS specification:

```
*)
ENTITY feature_volume_with_depth
SUBTYPE OF (feature_volume);
    penetration_depth : positive_length_measure_with_unit;
WHERE
    WRF34 : NOT('STRUCTURAL_FRAME_SCHEMA.
    FEATURE_VOLUME_COMPLEX' IN TYPE OF(SELF));
END_ENTITY;
(*
```

Attribute definitions:

penetration_depth

Declares the instance of positive_length_measure_with_unit used to define the length of the penetration of the feature.

Formal propositions

WRF34

The feature must be simple to be assigned a depth.

Notes:

New for CIS/2 2nd Edition.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

9.3.33 feature_volume_with_layout

Entity definition:

A type of volumetric feature that is repeated at each of the points defined by the layout. This entity would normally be instanced with another SUBTYPE of feature_volume using the ANDOR SUPERTYPE construct.

EXPRESS specification:

```
*)
ENTITY feature_volume_with_layout
SUBTYPE OF (feature_volume);
    layout : SET [2:?] OF point;
END_ENTITY;
(*
```

Attribute definitions:*layout*

Declares the set of instances of point associated with the feature_volume_with_layout, which are used to define the layout. It is assumed that all the points are defined from the same origin, and that that origin is the origin of the volume feature's own local coordinate system. There must be at least 2 separate and distinct instances of point referenced by each instance of feature_volume_with_layout.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

9.3.34 feature_volume_with_limit**Entity definition:**

A type of volumetric feature that is assigned an integer that indicates the extent of the feature's intended penetration. Volumetric features defined using the entity feature_volume are normally assumed to have no limits to extent of their application. Thus, if a hole is applied to the top flange of an I-section beam, it would normally be assumed that the hole passes through the bottom flange of the section if the cutting axis is so arranged. This entity would be instanced with another SUBTYPE of feature_volume to restrict the number of surfaces penetrated by that feature.

EXPRESS specification:

```

*)
ENTITY feature_volume_with_limit
  SUBTYPE OF (feature_volume);
    penetration_limit : INTEGER;
  WHERE
    WRF35 : NOT('STRUCTURAL_FRAME_SCHEMA.
    FEATURE_VOLUME_COMPLEX' IN TYPE OF(SELF));
    WRF36 : penetration_limit > 0;
END_ENTITY;
(*

```

Attribute definitions:*penetration_limit*

Declares the number of surfaces penetrated by the feature.

Formal propositions**WRF35**

The feature must be simple to be assigned a penetration limit.

WRF36

The feature must penetrate at least one surface

Notes:

New for CIS/2 2nd Edition.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

9.3.35 feature_volume_with_process

Entity definition:

A type of volumetric feature that is associated with a cutting process. This entity would normally be instanced with another SUBTYPE of feature_volume using the ANDOR SUPERTYPE construct. (See definitions of cut and structural_frame_process.)

EXPRESS specification:

```
*)
ENTITY feature_volume_with_process
  SUBTYPE OF (feature_volume);
    process_definition : cut;
END_ENTITY;
(*
```

Attribute definitions:

process_definition

Declares the instance of cut associated with the feature_volume_with_process, which specifies the nature and place of the cutting process that is used to form this feature. There must be one (and only one) instance of cut referenced by each instance of feature_volume_with_process.

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10 LPM/6 JOINT SYSTEMS

10.1 Joint Systems concepts and assumptions

Joint systems may be mechanical (e.g. a bolt group), chemical, welded, complex (e.g. welded fasteners such as shear studs) or amorphous (e.g. grout). LPM/6 supports a comprehensive range of fasteners such as self-drilling and self-tapping screws.

A located joint system is defined as a located and oriented implementation of the corresponding (specific) joint system. A (specific) joint system is the combination of the means by which a structural connection is realized, together with the geometric description of that connection. It could be, for example, a bolted mechanism arranged in accordance with a given layout (or bolt pattern) or a welded mechanism arranged in accordance with a given weld path.

10.2 Joint Systems type definitions

The following types have been added to the subject area for the 2nd Edition of CIS/2:

- weld_alignment
- weld_backing_type
- weld_configuration
- weld_intermittent_rule
- weld_shape_bevel
- weld_shape_butt
- weld_sidedness
- weld_surface_shape
- weld_taper_type

The following types have been modified in the subject area for the 2nd Edition of CIS/2:

- weld_type

10.2.1 chemical_mechanism_type

Type definition:

Classifies the chemical mechanism by its primary component; i.e. adhesive, grout, filler, or sealant. The chemical_mechanism_type may be declared as being undefined.

EXPRESS specification:

```
*)
TYPE chemical_mechanism_type
= ENUMERATION OF
    (adhesive, grout, filler, sealant, undefined);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

10.2.2 weld_alignment

Type definition:

An indicator as to whether or not each one of paired single fillet welds making up the double fillet weld are aligned with each other or offset from each other. This only has meaning in the case of intermittent fillet welds. The enumeration must be populated as one of the following values:

- 1) staggered -- the intermittent fillet welds are offset from each other
- 2) chained – the intermittent fillet welds are aligned with each other.

weld_alignment

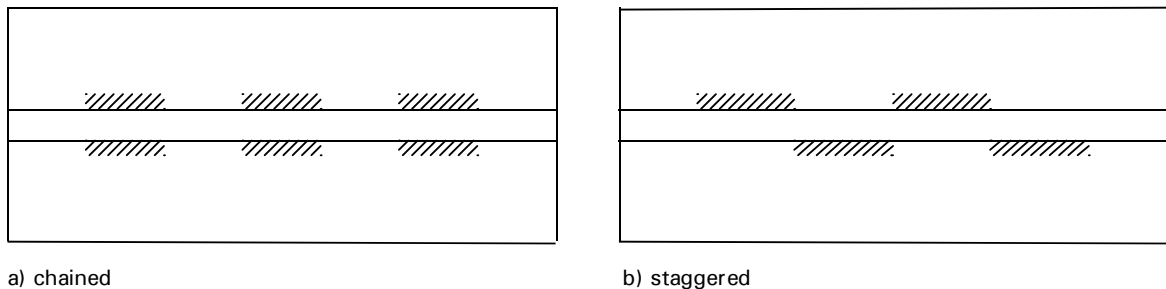


Figure 10.1: Alignment for intermittent welds

EXPRESS specification:

```
*)
TYPE weld_alignment
= ENUMERATION OF(staggered, chained);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

10.2.3 weld_backing_type

Type definition

An indicator as to the type of required underside (back) of the weld in order to prevent “blow through” and/or eliminate the need for back gouging. The enumeration must be populated as one of the following values:

- 1) none - no backing is defined
- 2) permanent - a backing attached at the welded joint permanently
- 3) copper_backing_bar - a backing bar made of copper
- 4) ceramic_tape - a backing tape made of ceramic
- 5) flare_backing_ring - a backing ring of flare form
- 6) permanent_backing_ring - a backing ring attached at the welded joint permanently.
- 7) removable_backing_ring - a removable backing ring
- 8) user_defined - a backing defined by the user.

EXPRESS specification:

```
*)
TYPE weld_backing_type
= ENUMERATION OF
  (none,
   permanent,
   copper_backing_bar,
   ceramic_tape,
   flare_backing_ring,
   permanent_backing_ring,
   removable_backing_ring,
   user_defined);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

10.2.4 weld_configuration

Type definition:

An indicator as to the configuration of the welded joint. The enumeration must be populated as one of the following values:

- 1) butt_joint – where 2 parts meet at the welded joint and where the components meet in-line and lie (approximately) parallel without overlapping
- 2) tee_joint – where 2 parts meet at the welded joint and the components meet at (approximately) right angles in a T shape formed by putting the end of one part on the surface of the other
- 3) corner_joint – where the ends of 2 parts meet at the welded joint and the components meet at (approximately) right angles in an L shape
- 4) lap_joint – where the common ends of 2 parts meet at the welded joint and the components lie (approximately) parallel with an overlap
- 5) edge_joint – where the edges of 2 parts meet at the welded joint and the components meet face to face
- 6) cruciform_joint – where the ends of 2 parts meet a 3rd part at the welded joint and the components meet at (approximately) right angles forming a cross
- 7) undefined.

EXPRESS specification:

```
*)
TYPE weld_configuration
= ENUMERATION OF
  (butt_joint,
   tee_joint,
   corner_joint,
   lap_joint,
   edge_joint,
   cruciform_joint,
```

```

        undefined);
END_TYPE;
(*)

```

Notes:

New for CIS/2 2nd Edition.

10.2.5 weld_intermittent_rule**Type definition:**

An indicator as to describe the rules of an intermittent weld. The enumeration must be populated as one of the following values:

- 1) none - no intermittent weld rule is defined
- 2) fixed_rule - fixed intermittent weld rule is defined
- 3) member_depth - intermittent weld rule depends on the depth of the parts being welded
- 4) percent_length - intermittent weld rule is expressed as a percent of the joint length of the parts being welded.

EXPRESS specification:

```

*)
TYPE weld_intermittent_rule
= ENUMERATION OF
    (none,
     fixed_rule,
     member_depth,
     percent_length);
END_TYPE;
(*)

```

Notes:

New for CIS/2 2nd Edition.

10.2.6 weld_penetration**Type definition:**

An indicator used to denote whether a special welded joint has been determined to result in “full” or “partial” penetration of the weld metal with the base metal of the parts being joined. (See Figure 10.14.) The penetration may be classed as undefined. The enumeration must be populated as one of the following values:

- 1) full_penetration - the weld penetrates fully through the base metal of the parts being joined.
- 2) deep_penetration - the weld penetrates deep inside the base metal of the parts being joined (applicable only to fillet welds).
- 3) partial_penetration - the weld penetrates only partially through the base metal of the parts being joined.
- 4) undefined – the class of penetration is undefined.

EXPRESS specification:

*)

TYPE weld_penetration

= ENUMERATION OF

(full_penetration, deep_penetration, partial_penetration, undefined);

END_TYPE;

(*)

Notes:

New for CIS/2 – replacing the type weld_penetrn in CIS/1.

Unchanged for the 2nd Edition.**10.2.7 weld_shape_bevel****Type definition:**

An indicator used to denote the particular shape of the cut form of the plate parts being welded by the bevel groove weld where the parts are of same or different thickness. The shape includes the flare-V form, flare-bevel, bevel-form, V-form, U-form, J-form, etc. Generally, bevel, flare-bevel and V-form are used for thickness of plates between 6mm - 18mm, while the U-form and J-form are used for heavy plate thicker than 18mm. The enumeration must be populated as one of the following values:


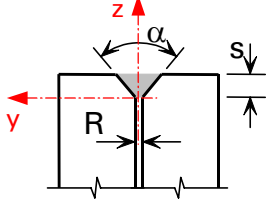
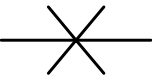

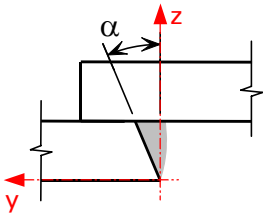
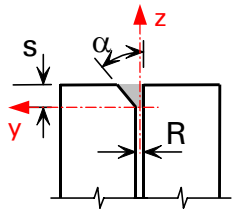
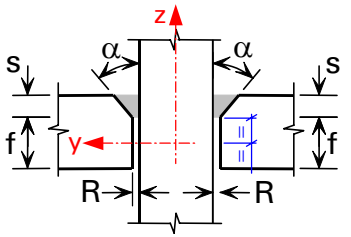
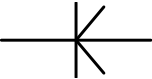
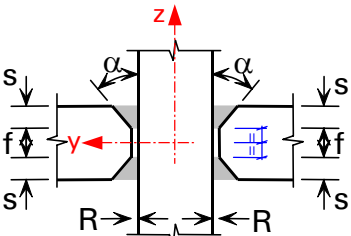

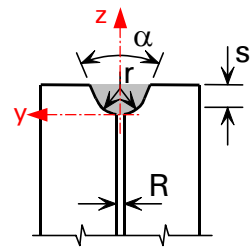
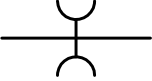
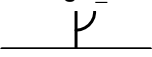
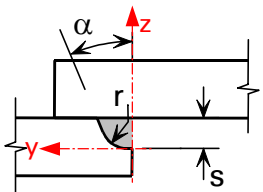
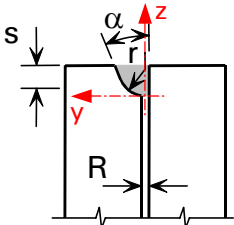
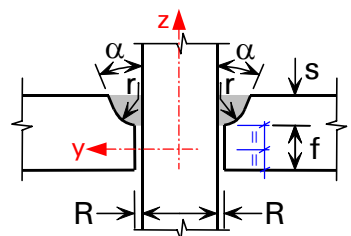
- 1) flare_single_V - the bevel form which is symmetrically open and spread outwards, has cuts on both plate parts, at one side of the welded edge.
- 2) flare_double_V - the bevel form which is symmetrically open and spread outwards, has cuts on both plate parts, at both sides of the welded edge.
- 3) flare_single_bevel - the bevel form which is non-symmetrically open and spread outwards, has cut on one plate part, at one side of the welded edge.
- 4) flare_double_bevel - the bevel form which is non-symmetrically open and spread outwards, has cut on one plate part, at both sides of the welded edge.
- 5) single_bevel - the bevel form which is non-symmetrically open and has a flat cut on one plate part, at one side of the welded edge.
- 6) double_bevel - the bevel form which is non-symmetrically open and has a flat cut on one plate part, at both sides of the welded edge (called also K-form).
- 7) single_V - the bevel form which is symmetrically open and has a flat cut on both plate parts, at one side of the welded edge.
- 8) double_V - the bevel form which is symmetrically open and has a flat cut on both plate parts, at both sides of the welded edge (called also X-form).
- 9) single_J - the bevel form which is non-symmetrically open and has a curved cut on one plate part, at one side of the welded edge.
- 10) double_J - the bevel form which is non-symmetrically open and has a curved cut on one plate part, at both sides of the welded edge.
- 11) single_U - the bevel form which is symmetrically open and has curved cut on both plate parts, at one side of the welded edge.
- 12) double_U - the bevel form which is symmetrically open and has curved cut on both plate parts, at both sides of the welded edge.
- 13) user_defined - other possible bevel cut form of edge defined by the user.

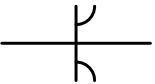
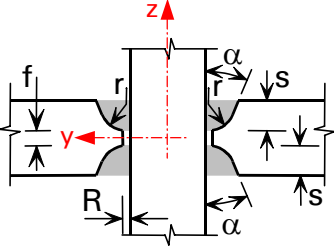
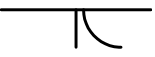
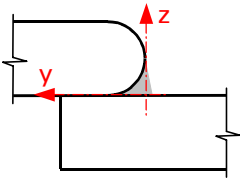
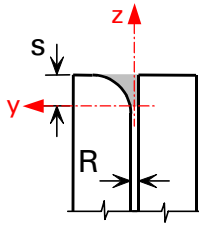
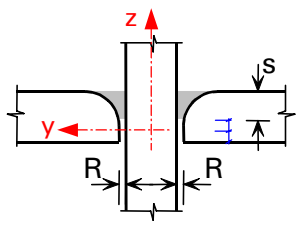
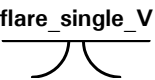
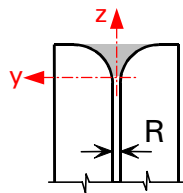
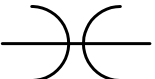
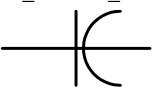
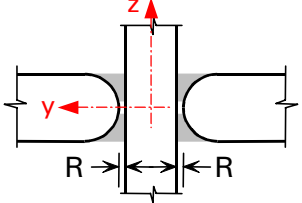
Table 10.1: *Examples of weld_shape_bevel for butt, corner and tee joints*

TYPE weld_shape_bevel	TYPE weld_configuration		
	butt_joint	corner_joint	tee_joint
single_V			-
double_V		-	-
single_bevel			
double_bevel			
single_U			-
double_U		-	-

TYPE weld_shape_bevel	TYPE weld_configuration		
	butt_joint	corner_joint	tee_joint
single_J			
double_J			
flare_single_bevel			
flare_single_V			-
flare_double_V		-	-
flare_double_bevel			
<p>Legend: R = root_gap, f = root_face, α = groove_angle, r = root_radius, S = groove_depth</p> <p>Local coordinate systems are shown with an x-axis running into the page and an origin that lies at mid-depth of root_face and mid width of root_gap. Weld path is based on this origin.</p>			

Table 10.2: *Examples of weld_shape_bevel for lap, edge and cruciform joints*

TYPE weld_shape_bevel	TYPE weld_configuration		
	lap_joint	edge_joint	cruciform_joint
single_V 	-		-
double_V 	-	-	-
single_bevel 			
double_bevel 	-	-	
single_U 	-		-
double_U 	-	-	-
single_J 			

TYPE weld_shape_bevel	TYPE weld_configuration		
	lap_joint	edge_joint	cruciform_joint
double_J 	-	-	
flare_single_bevel 			
flare_single_V 	-		-
flare_double_V 	-	-	-
flare_double_bevel 	-	-	
<p>Legend: R = root_gap, f = root_face, α = groove_angle, r = root_radius, S = groove_depth</p> <p>Local coordinate systems are shown with an x-axis running into the page and an origin that lies at mid-depth of root_face and mid width of root_gap. Weld path is based on this origin. Where $f = 0$, typically the case for edge joints, the origin lies at the base of the groove.</p>			

EXPRESS specification:

*)

TYPE weld_shape_bevel

= ENUMERATION OF

(flare_single_V,

flare_double_V,

flare_single_bevel,

flare_double_bevel,

single_bevel,

```

    double_bevel,
    single_V,
    double_V,
    single_J,
    double_J,
    single_U,
    double_U,
    user_defined);
END_TYPE;
(*)

```

Notes:

New for CIS/2 2nd Edition.

10.2.8 weld_shape_but**Type definition:**

An indicator used to denote the particular shape of the cut form of plate parts being welded by the butt groove weld where the parts are of same approximate thickness. The enumeration must be populated as one of the following values:

- 1) square - butt shape with no edge preparation (i.e., bevel/chamfering) is required, used for relatively thin plate parts with same approximate thickness
- 2) scarf - particular butt shape of parts whose welding surface is broadened by chamfering or cutting away the base metal
- 3) user_defined - other butt cut form defined by the user

EXPRESS specification:

```

*)
TYPE weld_shape_but
= ENUMERATION OF (square, scarf, user_defined);
END_TYPE;
(*)

```

Notes:

New for CIS/2 2nd Edition.

10.2.9 weld_sidedness**Type definition:**

An indicator denoting whether one side or both sides of a welded joint are welded. The enumeration must be populated as one of the following values:

- 1) one_side - one side of a weld joint is welded
- 2) both_sides - both sides of a weld joint are welded.

EXPRESS specification:

```

*)
TYPE weld_sidedness
= ENUMERATION OF (one_side, both_sides);
END_TYPE;
(*)

```

Notes:

New for CIS/2 2nd Edition.

10.2.10 weld_surface_shape**Type definition:**

An indicator used to denote the particular shape of the final exposed surface of the weld. The enumeration must be populated as one of the following values:

- 1) flush – the weld surface forms is flat
- 2) convex - the weld surface forms a surface that curves away from the weld root
- 3) concave - the weld surface forms a surface that curves towards from the weld root
- 4) user_defined - other surface shape defined by the user.

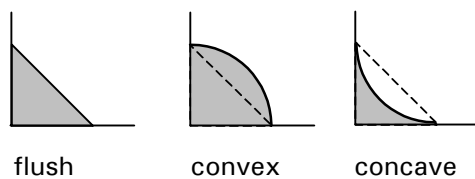


Figure 10.2: Example of weld_surface_shape

EXPRESS specification:

```
*)
TYPE weld_surface_shape
= ENUMERATION OF
  (flush,
   convex,
   concave,
   undefined);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

10.2.11 weld_taper_type**Type definition:**

An indicator used to denote the particular taper of the welded surface of a bevel weld. The enumeration must be populated as one of the following values:

- 1) non_taper - the welded surface is not sloped.
- 2) one_side_taper - the welded surface is sloped on one side of a bevel weld.
- 3) both_sides_taper - the welded surfaces are sloped on both sides of the bevel weld.

EXPRESS specification:

```
*)
TYPE weld_taper_type
= ENUMERATION OF (non_taper, one_side_taper, both_sides_taper);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

10.2.12 weld_type**Type definition:**

An indicator as to the actual arrangement of the weld metal in relation to the parts being joined. The required configuration should agree with this physical implementation. The weld type may be classed as undefined. The enumeration must be populated as one of the following values:

- 1) butt_weld – weld in which the weld metal lies within the (projected) boundaries of the parts
- 2) fillet_weld – weld in which the weld bead is laid down along the outside of the part
- 3) spot_weld – weld performed by locally heating and simultaneously pressurizing through the electrode, with current and welding force concentrated in comparatively small area (spot)
- 4) plug_weld – weld filling a approximately circular void
- 5) seam_weld – weld performed by locally heating and simultaneously pressurizing through the electrode, with current and welding force concentrated in a series of comparatively small areas
- 6) slot_weld – weld filling an elongated void
- 7) stud_weld – weld whose welding is performed by generating arc between the tip of a bolt, bar, etc. and base metal, and pressing it into the resulting molten pool.
- 8) surfacing_weld – weld in which metal is deposited on a base metal surface, such as cladding by welding, thermal spraying, etc.
- 9) undefined – the type of weld is not defined.

EXPRESS specification:

*)

TYPE weld_type

= ENUMERATION OF

(butt_weld,
fillet_weld,
spot_weld,
plug_weld,
seam_weld,
slot_weld,
stud_weld,
surfacing_weld,
undefined);

END_TYPE;

(*

Notes

New for CIS/2 - replacing the type weld_typ in CIS/1.

Modified for 2nd Edition.

10.3 Joint Systems entity definitions

The following entities have been added to this subject area for the 2nd Edition:

- fastener_mechanism_with_position
- joint_system_welded_with_shape
- weld_mechanism_complex
- weld_mechanism_fillet
- weld_mechanism_fillet_continuous
- weld_mechanism_fillet_intermittent
- weld_mechanism_groove
- weld_mechanism_groove_beveled
- weld_mechanism_groove_butt
- weld_mechanism_prismatic
- weld_mechanism_spot_seam

The following entities have been modified in this subject area for the 2nd Edition:

- fastener_simple_bolt
- fastener_simple_nut
- fastener_simple_stud
- fastener_simple_washer
- joint_system
- joint_system_welded_linear
- joint_system_welded_point
- joint_system_welded_surface
- weld_mechanism

10.3.1 chemical_mechanism

Entity definition:

A type of structural frame product that is used to define a component of a joint system in which the action is neither mechanical nor welded but chemical. This entity is used to represent the specification of one of the layers of adhesive, grout, filler, or sealant filling a joint. It is, therefore, normally used with the entity joint_system_chemical.

As a SUBTYPE of structural_frame_product, a chemical_mechanism inherits the three attributes item_number, item_name, and item_description (from structural_frame_item). It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a chemical_mechanism may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)

- document references (via the entity `structural_frame_item_documented`) – this allows the appropriate national standard or code of practice used in the definition of the `chemical_mechanism` to be described in detail
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A `chemical_mechanism` may also be related to another `chemical_mechanism` (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

As a SUBTYPE of `structural_frame_product`, an `chemical_mechanism` may also be assigned a material (via the entity `structural_frame_product_with_material`).

EXPRESS specification:

*)

ENTITY `chemical_mechanism`

SUBTYPE OF (`structural_frame_product`);

`layer_thickness` : `positive_length_measure_with_unit`;

`layer_design_strength` : OPTIONAL `pressure_measure_with_unit`;

`layer_type` : `chemical_mechanism_type`;

END_ENTITY;

(*

Attribute definitions:

layer_thickness

Declares the instance of `positive_length_measure_with_unit` associated with the `chemical_mechanism`, which specifies the numerical value with an appropriate unit of the thickness of the layer of the substance used (adhesive, grout, filler, or sealant) in the chemical mechanism.

layer_design_strength

Declares the instance of `pressure_measure_with_unit` associated with the `chemical_mechanism`, which specifies the numerical value with an appropriate unit of the design strength of the substance used (adhesive, grout, filler, or sealant) in the chemical mechanism.

layer_type

Declares the classification of the type of the chemical mechanism used as either adhesive, grout, filler, or sealant. The type must be declared, but it may be declared as being undefined.

Notes:

New for CIS/2.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.2 fastener

Entity definition:

A generic structural frame product that acts as a mechanical fastener in a joint system. The SUBTYPEs of this entity provide the details of bolts, nuts, washers, screws, pins,

etc. The fastener can be categorized through its SUBTYPEs by the complexity of its geometry as being either a simple fastener or a complex fastener.

As a SUBTYPE of `structural_frame_product`, a fastener inherits the three attributes `item_number`, `item_name`, and `item_description` (from `structural_frame_item`). It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a fastener may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`) – this allows the appropriate national standard or code of practice used in the definition of the fastener to be described in detail
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A fastener may also be related to another fastener (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

As a SUBTYPE of `structural_frame_product`, a fastener may also be assigned a material (via the entity `structural_frame_product_with_material`).

EXPRESS specification:

*)

ENTITY fastener

SUPERTYPE OF (ONEOF(fastener_simple, fastener_complex))

SUBTYPE OF (structural_frame_product);

fastener_grade : OPTIONAL label;

END_ENTITY;

(*

Attribute definitions:

fastener_grade

A short text description of the grade of the material of the fastener. This could be, for example, a codification of the type of bolt (such as 8.8 etc.) in accordance with the applicable design standard / code of practice.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.3 fastener_complex

Entity definition:

A type of fastener whose shape is defined using the explicit geometry constructs of the STEP Generic Resources through a reference to an instance of `shape_representation_with_units`.

EXPRESS specification:

```

*)
ENTITY fastener_complex
SUBTYPE OF (fastener);
    fastener_shape : shape_representation_with_units;
END_ENTITY;
(*)

```

Attribute definitions:*fastener_shape*

Declares the instance of *shape_representation_with_units* associated with the *fastener_complex*, which defines the shape of the fastener. There must be one (and only one) instance of *shape_representation_with_units* associated with each instance of *fastener_complex*.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.4 fastener_mechanism**Entity definition:**

A type of *structural_frame_product* used to bring together a particular set of fasteners. For example, a fastener mechanism could contain a bolt, nut and washer. A particular fastener mechanism is defined once, and then used several times in many different mechanical joint systems (*joint_system_mechanical*). The fastener mechanism itself would be defined as the bolt, nut and washer. It is assumed that each of the fasteners in the fastener mechanism share a common longitudinal axis defined by the longitudinal axis of the first fastener in the mechanism; e.g. the bolt. The origin of this axis is defined as the extreme end of the first fastener (e.g. the top of the head of bolt). To reposition this origin (e.g. to set the origin of the *fastener_mechanism* as the underside of the bolt head), use the SUBTYPE *fastener_mechanism_with_position*.

As a SUBTYPE of *structural_frame_product*, a *fastener_mechanism* inherits the three attributes *item_number*, *item_name*, and *item_description* (from *structural_frame_item*). It also inherits the many implicit relationships that exist because other entities use *structural_frame_item* as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a *fastener_mechanism* may have:

- approvals (via the entity *structural_frame_item_approved*)
- prices (via the entity *structural_frame_item_priced*)
- certificates (via the entity *structural_frame_item_certified*)
- document references (via the entity *structural_frame_item_documented*) – this allows the appropriate national standard or code of practice used in the definition of the *fastener_mechanism* to be described in detail
- properties (via the entity *item_property_assigned*)
- item_references (via the entity *item_reference_assigned*).

A `fastener_mechanism` may also be related to another `fastener_mechanism` (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

As a SUBTYPE of `structural_frame_product`, a `fastener_mechanism` may also be assigned a material (via the entity `structural_frame_product_with_material`).

EXPRESS specification:

```
*)
ENTITY fastener_mechanism
SUBTYPE OF (structural_frame_product);
    sequence : OPTIONAL text;
    fasteners : LIST [1:?] OF fastener;
END_ENTITY;
(*
```

Attribute definitions:

sequence

An optional text description of the sequence in which the components of the `fastener_mechanism` (i.e. the fasteners) are put together.

fasteners

Declares the list of instances of `fastener` associated with the `fastener_mechanism`. There must be at least one `fastener` associated with each instance of `fastener_mechanism`. The LIST data type implies that the order of the instances is important. The sequence of references in the list should, therefore, correspond to the descriptive sequence given above.

Notes:

New for CIS/2; partially addressed by BOLT_MECHANISM in CIS/1.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.5 fastener_mechanism_with_position

Entity definition:

A type of `fastener_mechanism` that includes a list of dimensions used to define the positions of the components making up the fastener mechanism.

The SUPERTYPE entity `fastener_mechanism` typically refers to bolts, nuts, and washers that are all located together in a `located_joint_system`, with no way to specify where the nuts and washers are located relative to the head of the bolt. It is assumed that each of the fasteners in the fastener mechanism share a common longitudinal axis defined by the longitudinal axis of the first fastener in the mechanism; e.g. the bolt. The origin of this axis is defined as the extreme end of the first fastener (e.g. the top of the head of bolt).

The `fastener_mechanism_with_position` is used to reposition this origin (e.g. to set the origin of the `fastener_mechanism` as the underside of the bolt head) and to specify the positions of its constituent fasteners. It may also be used to specify, if there are two washers, where to locate the washers relative to the side of the material the bolt goes through.

This entity may be used even if the first fastener is set at the default position, in which case the first value in the list of positions would be set to zero.

EXPRESS specification:

```

*)
ENTITY fastener_mechanism_with_position
SUBTYPE OF (fastener_mechanism);
    fastener_positions : LIST [1:?] OF length_measure_with_unit;
WHERE
    WRF32 : SIZEOF(fastener_positions) = SIZEOF(SELF\fastener_mechanism.fasteners);
END_ENTITY;
(*

```

Attribute definitions:**fastener_positions**

Declares the list of one or more instances of `length_measure_with_unit` associated with the `fastener_mechanism_with_position`. These instances provide the dimensions from the origin of the `fastener_mechanism` along its longitudinal axis to the start of each fastener. To set the origin of the `fastener_mechanism` as the underside of the bolt head, the first instance in the LIST should provide a negative value equivalent to the height of the bolt head. Furthermore, if the next fastener is a washer fitting tightly under the bolt head, the second instance in the LIST should provide a value equal to or close to zero. Alternatively, to set the origin of the `fastener_mechanism` as the surface of the joined parts, the first instance in the LIST should provide a negative value equivalent to the height of the bolt head plus the thickness of the washer. (See Figure 10.4.)

Formal propositions:**WRF32**

The number of fasteners must equal the number of positions (even if the first fastener is set at the default position).

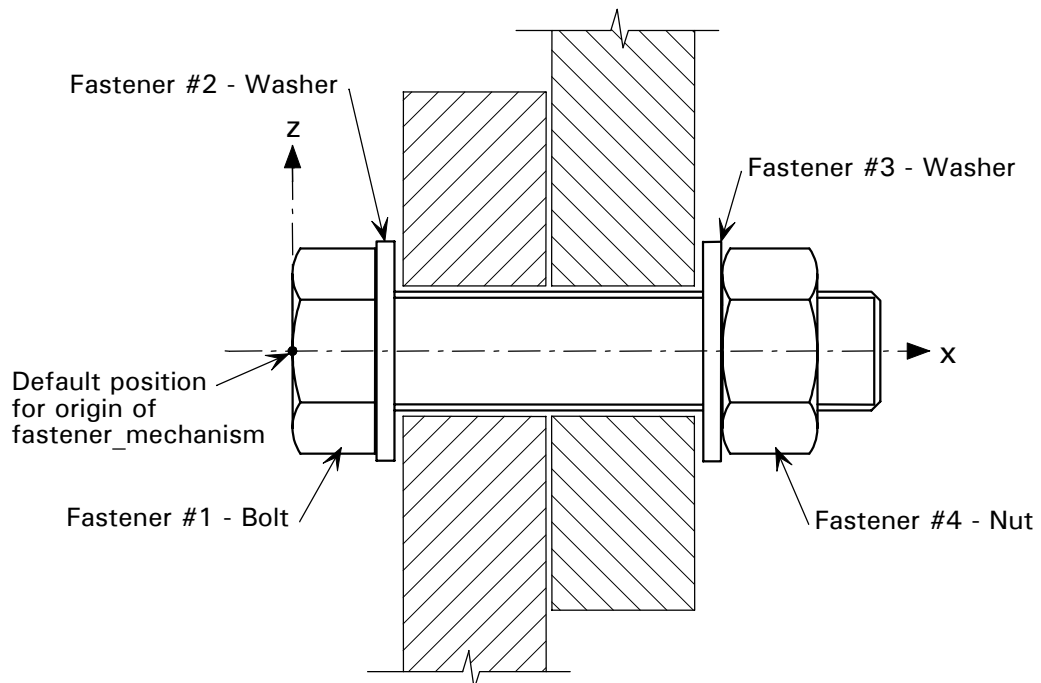


Figure 10.3: *Example of a fastener_mechanism showing origin*

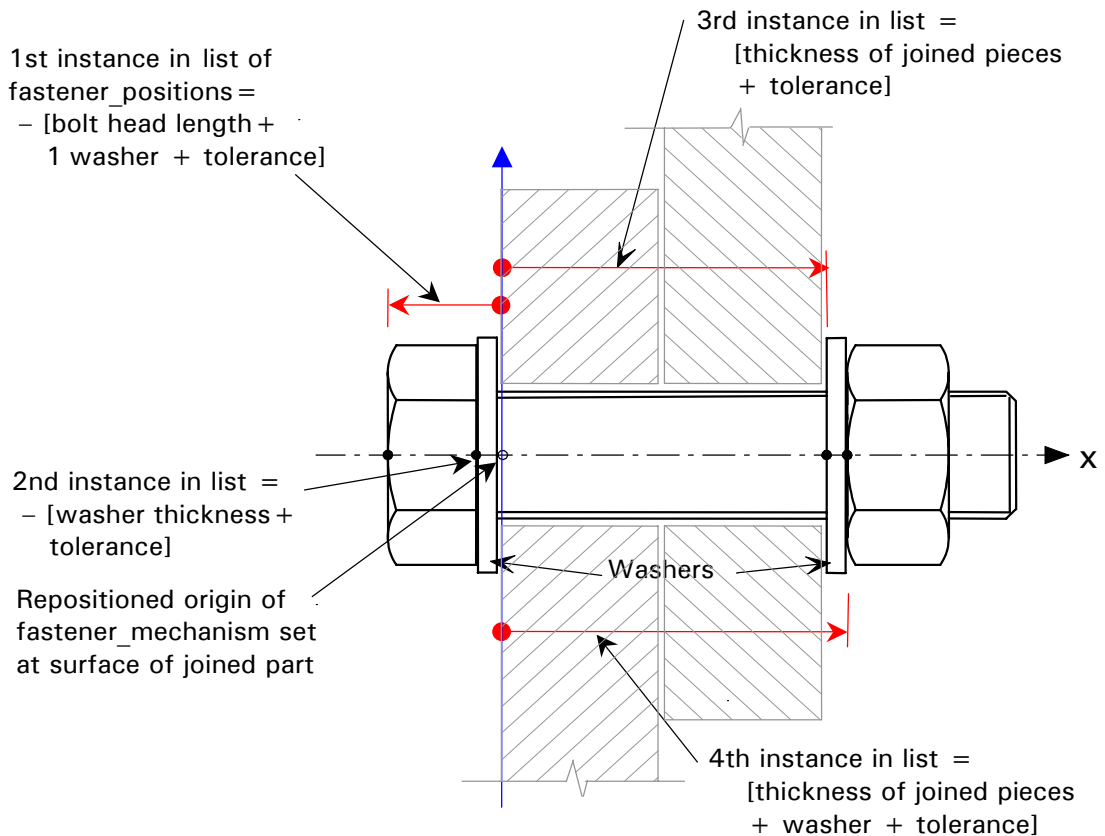


Figure 10.4: *Fastener_mechanism with repositioned origin*

Notes:

New for CIS/2 2nd Edition.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

10.3.6 fastener_simple

Entity definition:

A type of fastener whose geometry is defined implicitly using a set of parameters. In most cases, the longitudinal axis of the fastener will be a straight line. However, this entity is an ANDOR SUPERTYPE, which allowed a simple bolt to have a curved longitudinal axis when the two SUBTYPES fastener_simple_bolt and fastener_simple_curved are instanced together.

EXPRESS specification:

*)

```
ENTITY fastener_simple
SUPERTYPE OF (ONEOF
    (fastener_simple_bolt,
    fastener_simple_nut,
    fastener_simple_washer,
    fastener_simple_stud,
    fastener_simple_nail,
    fastener_simple_pin,
    fastener_simple_screw,
    fastener_simple_shear_connector) ANDOR
```

```

        fastener_simple_countersunk ANDOR
        fastener_simple_curved)
SUBTYPE OF (fastener);
    nominal_diameter : positive_length_measure_with_unit;
    nominal_length : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
(*)

```

Attribute definitions:

nominal_diameter

Declares the first instance of `positive_length_measure_with_unit` associated with the `fastener_simple`, which specifies the numerical value with an appropriate unit of the nominal diameter of the fastener. The nominal diameter is measured perpendicular to the fastener's longitudinal axis in accordance with the appropriate standard or code of practice. (The details of the appropriate standard or code of practice may be supplied via the entity `structural_frame_item_documented`.)

nominal_length

Declares the second instance of `positive_length_measure_with_unit` associated with the `fastener_simple`, which specifies the numerical value with an appropriate unit of the nominal length of the fastener. The nominal length is measured along the fastener's longitudinal axis in accordance with the appropriate standard or code of practice.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.7 fastener_simple_bolt

Entity definition:

A type of simple fastener that provides the description of a bolt. A bolt can be specified once, and then used in many `fastener_mechanisms`. The bolt may be categorized (through the SUBTYPES) by the shape of its head; i.e. circular, hexagonal, or square.

EXPRESS specification:

```

*)
ENTITY fastener_simple_bolt
SUPERTYPE OF (ONEOF
    (fastener_simple_bolt_circular_head,
    fastener_simple_bolt_hexagonal_head,
    fastener_simple_bolt_square_head))
SUBTYPE OF (fastener_simple);
    length_of_shank : OPTIONAL positive_length_measure_with_unit;

```

```

    bolt_preload : OPTIONAL force_measure_with_unit;
    full_section_area : OPTIONAL area_measure_with_unit;
    reduced_section_area : OPTIONAL area_measure_with_unit;
DERIVE
    bolt_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;
END_ENTITY;
(*)

```

Attribute definitions:

length_of_shank

Declares the instance of `positive_length_measure_with_unit` that may be associated with the `fastener_simple_bolt` to specify the numerical value with an appropriate unit of the measurement of the length of the shank (as defined by the appropriate standard) of the bolt.

bolt_preload

Declares the instance of `force_measure_with_unit` that may be associated with the `fastener_simple_bolt` to specify the numerical value with an appropriate unit of the load applied to the bolt (as defined by the appropriate standard or code of practice) when the bolt mechanism is brought together.

full_section_area

Declares the first instance of `area_measure_with_unit` that may be associated with the `fastener_simple_bolt` to specify the numerical value with an appropriate unit of the full sectional area of the bolt shaft (as defined by the appropriate standard or code of practice).

reduced_section_area

Declares the second instance of `area_measure_with_unit` that may be associated with the `fastener_simple_bolt` to specify the numerical value with an appropriate unit of the reduced sectional area of the bolt shaft (as defined by the appropriate standard or code of practice). This value should take into account the bolt thread and any other modifications to the bolt that removes material from its shaft.

bolt_ref

A bag of short alphanumerical identifying references for the bolt derived from the attribute `item_ref` from the SUPERTYPE `structural_frame_item`. If a standard reference is specified for a bolt it should be represented using the attribute `ref` in the entity `item_reference`. A relationship is then made using the entity `item_assignment`. If the bolt is not assigned an item reference, an empty bag will be returned here.

In most cases, a bolt will be assigned only one reference and that would be taken from a list of standard references from a flavour file. However, there may be times when users wish to add their own references for the bolt. These could be taken from an industry agreed library, or from a proprietary list.

Notes:

Known as BOLT in CIS/1.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE clause added.

10.3.8 fastener_simple_bolt_circular_head

Entity definition:

A type of simple bolt whose head is described as circular and is specified using two dimensions.

EXPRESS specification:

```
*)
ENTITY fastener_simple_bolt_circular_head
SUBTYPE OF (fastener_simple_bolt);
    bolt_head_height : positive_length_measure_with_unit;
    head_diameter : positive_length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

bolt_head_height

Declares the first instance of `positive_length_measure_with_unit` associated with the `fastener_simple_bolt_circular_head`, which specifies the numerical value with an appropriate unit of the measurement of the height of the head (as defined by the appropriate standard) of the bolt.

head_diameter

Declares the second instance of `positive_length_measure_with_unit` associated with the `fastener_simple_bolt_circular_head`, which specifies the numerical value with an appropriate unit of the diameter of the head of the bolt. The dimension is measured perpendicular to the longitudinal axis of the bolt (as defined by the appropriate standard or code of practice).

Informal propositions

This entity is instanced when the dimensions of the bolt head need to be specified.

Notes:

New for CIS/2.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.9 fastener_simple_bolt_hexagonal_head

Entity definition:

A type of simple bolt whose head is described as hexagonal and is specified using three dimensions.

EXPRESS specification:

```
*)
ENTITY fastener_simple_bolt_hexagonal_head
SUBTYPE OF (fastener_simple_bolt);
    bolt_head_height : positive_length_measure_with_unit;
    distance_across_vertices : OPTIONAL positive_length_measure_with_unit;
    distance_across_flats : OPTIONAL positive_length_measure_with_unit;
WHERE
```

```

WRF3 : EXISTS (distance_across_vertices) OR
        EXISTS (distance_across_flats);
WRF4 : NOT( (distance_across_flats.value_component >
            distance_across_vertices.value_component) AND (EXISTS
            (distance_across_vertices) AND EXISTS (distance_across_flats)) );
END_ENTITY;
(*)

```

Attribute definitions:

bolt_head_height

Declares the first instance of `positive_length_measure_with_unit` associated with the `fastener_simple_bolt_hexagonal_head`, which specifies the numerical value with an appropriate unit of the measurement of the height of the head (as defined by the appropriate standard) of the bolt.

distance_across_vertices

Declares the second instance of `positive_length_measure_with_unit` that may be associated with the `fastener_simple_bolt_hexagonal_head` to specify the numerical value with an appropriate unit of the distance across the vertices of the hexagonal head of the bolt. The dimension is measured perpendicular to the longitudinal axis of the bolt (as defined by the appropriate standard or code of practice).

distance_across_flats

Declares the third instance of `length_measure_with_unit` that may be associated with the `fastener_simple_bolt_hexagonal_head` to specify the numerical value with an appropriate unit of the distance across the flats of the hexagonal head of the bolt. The dimension is measured perpendicular to the longitudinal axis of the bolt (as defined by the appropriate standard or code of practice).

Formal propositions:

WRF3

Either one of the attributes `distance_across_vertices` or `distance_across_flats` must be populated. (Both attributes may be populated, although one can be derived from the other.)

WRF4

If both of the attributes `distance_across_vertices` and `distance_across_flats` are populated, then the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `distance_across_flats` must not be greater than the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `distance_across_vertices`. In other words, for a hexagonal bolt, the dimensions of the head should be such that the distance across vertices is greater than the distance across flats. It is assumed that both dimensions are measured in the same units.

Informal propositions:

The entity is instanced when the dimensions of the bolt head need to be specified.

Both of the attributes `distance_across_vertices` and `distance_across_flats` are measured in the same units.

Notes:

New for CIS/2; addressed by BOLT in CIS/1.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.10 fastener_simple_bolt_square_head**Entity definition:**

A type of simple bolt whose head is described as square and is specified using two dimensions.

EXPRESS specification:

*)

ENTITY fastener_simple_bolt_square_head

SUBTYPE OF (fastener_simple_bolt);

 bolt_head_height : positive_length_measure_with_unit;

 distance_across_flats : positive_length_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:*bolt_head_height*

Declares the first instance of positive_length_measure_with_unit associated with the fastener_simple_bolt_square_head, which specifies the numerical value with an appropriate unit of the measurement of the height of the head (as defined by the appropriate standard) of the bolt.

distance_across_flats

Declares the second instance of positive_length_measure_with_unit associated with the fastener_simple_bolt_square_head, which specifies the numerical value with an appropriate unit of the distance across the flats of the square head of the bolt. The dimension is measured perpendicular to the longitudinal axis of the bolt (as defined by the appropriate standard or code of practice).

Informal propositions:

The entity is instanced when the dimensions of the bolt head need to be specified.

Notes:

New for CIS/2.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.11 fastener_simple_countersunk**Entity definition:**

A type of simple fastener whose head is defined as being countersunk. (See Figure 10.5.)

EXPRESS specification:

*)

ENTITY fastener_simple_countersunk

SUBTYPE OF (fastener_simple);

countersink_angle : plane_angle_measure_with_unit;

countersink_depth : OPTIONAL positive_length_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:**countersink_angle**

Declares the instance of plane_angle_measure_with_unit associated with the fastener_simple_countersunk, which specifies the numerical value with an appropriate unit of the angle from one face of the countersink to the other. It is implied that the countersink is symmetrical about the longitudinal axis of the fastener.

countersink_depth

Declares the instance of positive_length_measure_with_unit that may be associated with the fastener_simple_countersunk to specify the numerical value with an appropriate unit of the linear dimension from the bottom of the fastener's head to the top of the countersink. Where the head has no 'bottom', this dimension is measured from the intersection of the fastener's shaft and the countersunk face. Where the entire head is countersunk (i.e. the fastener's head has no 'vertical' faces) the countersink_depth is equal to the head height (if given).

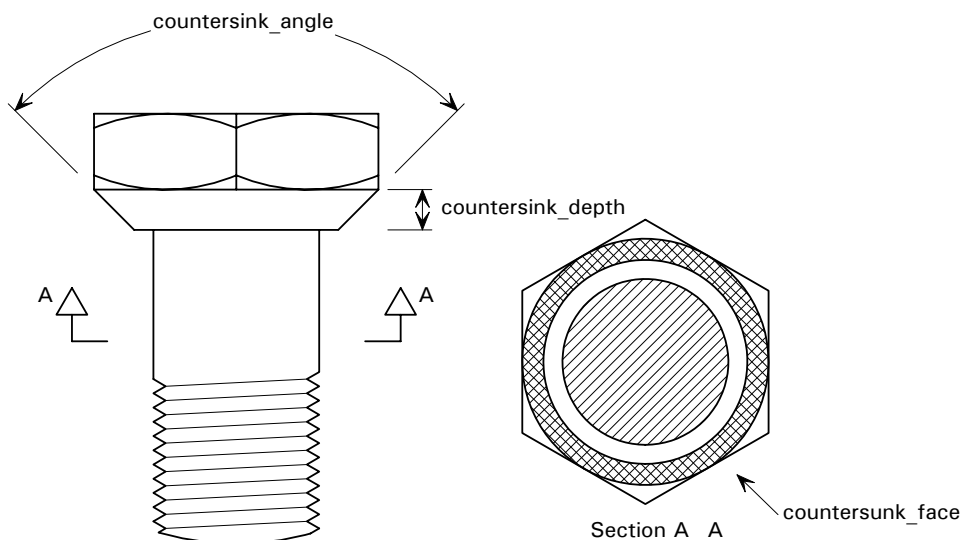


Figure 10.5 An example of a 'countersunk fastener'

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.12 fastener_simple_curved

Entity definition:

A (special) type of simple fastener, whose longitudinal axis is not a straight line, but is defined by an explicit curve; e.g. a 'U-bolt'. The curve entity being taken from STEP Part 42.

EXPRESS specification:

```
*)
ENTITY fastener_simple_curved
SUBTYPE OF (fastener_simple);
    curve_definition : curve;
END_ENTITY;
(*
```

Attribute definitions:

curve_definition

Declares the instance of curve associated with the fastener_simple_curved, which specifies the geometry of the longitudinal axis of the simple fastener. There must be one (and only one) instance of curve referenced by each instance of fastener_simple_curved.

Informal propositions:

This entity is instanced only when the simple fastener is **not** straight.

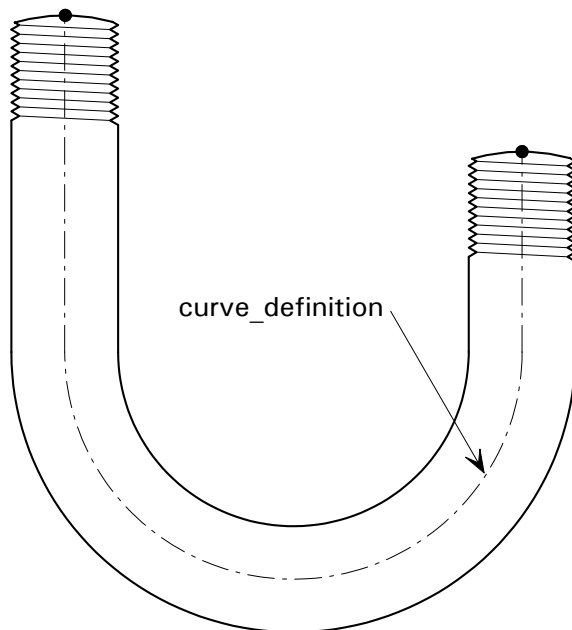


Figure 10.6 An example of a 'curved fastener'

Notes:

New for CIS/2.

The entity curve is defined in STEP Part 42.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.13 fastener_simple_nail

Entity definition:

A type of simple unthreaded fastener that is driven in rather than screwed in.

EXPRESS specification:

```
*)
ENTITY fastener_simple_nail
SUBTYPE OF (fastener_simple);
    nail_type : OPTIONAL text;
    nail_drive_type : OPTIONAL text;
    nail_head_shape : OPTIONAL text;
    nail_point_type : OPTIONAL text;
END_ENTITY;
(*
```

Attribute definitions:

nail_type

An optional text description of the type of the nail.

nail_drive_type

An optional text description of the system used to drive in the nail to its final position; e.g. hand or power driven.

nail_head_shape

An optional text description of the shape of the head of the nail; e.g. circular, oval, etc.

nail_point_type

An optional text description of the shape of the point (of tip) of the nail; e.g. conical, chisel edged, etc.

Informal propositions:

Although all of the attributes are optional, it is expected that at least one of the attributes will be populated.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.14 fastener_simple_nut

Entity definition:

A type of simple fastener that provides the description of a nut. A nut can be specified once, and then used in many fastener mechanisms. A fastener mechanisms may include more than one nut (e.g. a nut and a locking nut). A nut may be categorized (through the SUBTYPEs) by its shape; i.e. circular, hexagonal, or square.

A nut may also be categorized as being 'closed' (through the SUBTYPE fastener_simple_nut_closed). If this SUBTYPE is not instanced then the nut is 'open'. A nut is defined as 'open' if the bolt is able to pass the whole way through the nut, and

‘closed’ if the bolt cannot. For open nuts, it is assumed that the thread extends the full depth of the nut.

EXPRESS specification:

```

*)
ENTITY fastener_simple_nut
SUPERTYPE OF (ONEOF
    (fastener_simple_nut_circular,
    fastener_simple_nut_hexagonal,
    fastener_simple_nut_square) ANDOR
    fastener_simple_nut_closed)
SUBTYPE OF (fastener_simple);
DERIVE
    nut_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;
WHERE
    WRF28 : NOT('STRUCTURAL_FRAME_SCHEMA.FASTENER_SIMPLE_CURVED' IN
        TYPE OF(SELF));
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own but derives one from its SUPERTYPE. It does inherit several attributes from its SUPERTYPE fastener_simple. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity fastener_simple, and to bring together the SUBTYPEs under a common category.

nut_ref

A bag of short alphanumeric identifying references for the nut derived from the attribute item_ref from the SUPERTYPE structural_frame_item. If a standard reference is specified for a nut it should be represented using the attribute ref in the entity item_reference. A relationship is then made using the entity item_assignment. If the nut is not assigned an item reference, an empty bag will be returned here.

In most cases, a nut will be assigned only one reference and that would be taken from a list of standard references from a flavour file. However, there may be times when users wish to add their own references for the nut. These could be taken from an industry agreed library, or from a proprietary list.

Formal propositions:

WRF28

A fastener_simple_nut shall not be of the type fastener_simple_curved. In other words, a nut cannot be curved.

Notes:

Known as NUT in CIS/1.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE clause added.

10.3.15 fastener_simple_nut_circular

Entity definition:

A type of simple nut whose shape is described as circular and is specified using a single dimension.

EXPRESS specification:

```
*)
ENTITY fastener_simple_nut_circular
SUBTYPE OF (fastener_simple_nut);
    outside_diameter : positive_length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

outside_diameter

Declares the instance of `positive_length_measure_with_unit` associated with the `fastener_simple_nut_circular`, which specifies the numerical value with an appropriate unit of the external diameter of the nut. The dimension is measured perpendicular to the longitudinal axis of the nut (as defined by the appropriate standard or code of practice).

Informal propositions:

This entity is instanced when the dimensions of the nut in the yz-plane need to be specified.

Notes:

New for CIS/2.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.16 fastener_simple_nut_closed

Entity definition:

A type of `fastener_simple_nut` described as 'closed'. For a closed nut, a bolt is unable to pass the whole way through the nut. The thread of the nut is assumed to extend to the full clear depth of the hole – as far as the hole extends at the nominal diameter (inherited from the SUPERTYPE entity `fastener_simple`).

EXPRESS specification:

```
*)
ENTITY fastener_simple_nut_closed
SUBTYPE OF (fastener_simple_nut);
    nut_depth : positive_length_measure_with_unit;
WHERE
    WRF5 : SELF.fastener_simple.nominal_length.value_component >
        nut_depth.value_component;
END_ENTITY;
(*
```

Attribute definitions:***nut_depth***

Declares the instance of `positive_length_measure_with_unit` associated with the `fastener_simple_nut_closed`, which specifies the numerical value with an appropriate unit of the linear dimension of the depth of the nut. The depth of the nut is measured in the direction of the longitudinal x-axis of the nut from the top of the of open end to the bottom of the threaded portion. (The actual hole may extend beyond the given nut depth, but at a diameter that is less than the given nominal diameter. This allows for the formation of the 'blind' hole.)

Formal propositions:***WRF5***

The numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `nominal_length` (inherited from the SUPERTYPE `fastener_simple`) must be greater than the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `nut_depth`. In other words, for a closed nut, the dimensions should be such that the nominal (overall) length must be greater than depth of the threaded portion of the nut. It is assumed that both dimensions are measured in the same units.

Informal propositions:

Both of the attributes `nominal_length` and `nut_depth` are measured in the same units.

Notes

New for CIS/2.

See Diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.17 fastener_simple_nut_hexagonal**Entity definition:**

A type of simple nut whose shape is described as hexagonal and is specified using two dimensions.

EXPRESS specification:

*)

ENTITY `fastener_simple_nut_hexagonal`

SUBTYPE OF (`fastener_simple_nut`);

`distance_across_vertices` : OPTIONAL `positive_length_measure_with_unit`;

`distance_across_flats` : OPTIONAL `positive_length_measure_with_unit`;

WHERE

 WRF6 : EXISTS (`distance_across_vertices`) OR
 EXISTS (`distance_across_flats`);

 WRF7 : NOT((`distance_across_flats.value_component` >
 `distance_across_vertices.value_component`) AND (EXISTS
 (`distance_across_vertices`) AND EXISTS (`distance_across_flats`)));

END_ENTITY;

(*

Attribute definitions:*distance_across_vertices*

Declares the first instance of `positive_length_measure_with_unit` associated with the `fastener_simple_nut_hexagonal`, which specifies the numerical value with an appropriate unit of the distance across the vertices of the hexagonal nut. The dimension is measured perpendicular to the longitudinal axis of the nut (as defined by the appropriate standard or code of practice).

distance_across_flats

Declares the second instance of `positive_length_measure_with_unit` associated with the `fastener_simple_nut_hexagonal`, which specifies the numerical value with an appropriate unit of the distance across the flats of the hexagonal nut. The dimension is measured perpendicular to the longitudinal axis of the bolt (as defined by the appropriate standard or code of practice).

Formal propositions:*WRF6*

Either one of the attributes `distance_across_vertices` or `distance_across_flats` must be populated. (Both attributes may be populated, although one can be derived from the other.)

WRF7

If both of the attributes `distance_across_vertices` and `distance_across_flats` are populated, then the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `distance_across_flats` must not be greater than the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `distance_across_vertices`. In other words, for a hexagonal nut, the dimensions should be such that the distance across vertices is greater than the distance across flats. It is assumed that both dimensions are measured in the same units.

Informal propositions:

This entity is instanced when the dimensions of the nut in the yz-plane need to be specified.

Both of the attributes `distance_across_vertices` and `distance_across_flats` are measured in the same units.

Notes:

New for CIS/2; addressed by NUT in CIS/1.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.18 fastener_simple_nut_square**Entity definition:**

A type of simple nut whose shape is described as square and is specified using a single dimension.

EXPRESS specification:

```

*)
ENTITY fastener_simple_nut_square
SUBTYPE OF (fastener_simple_nut);
    distance_across_flats : positive_length_measure_with_unit;
END_ENTITY;
(*

```

Attribute definitions:*distance_across_flats*

Declares the instance of *positive_length_measure_with_unit* associated with the *fastener_simple_nut_square*, which specifies the numerical value with an appropriate unit of the distance across the flats of the square nut. The dimension is measured perpendicular to the longitudinal axis of the nut (as defined by the appropriate standard or code of practice).

Informal propositions:

This entity is instanced when the dimensions of the nut in the yz-plane need to be specified.

Notes:

New for CIS/2.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.19 fastener_simple_pin**Entity definition:**

A type of simple fastener that can be described by its length and diameter (inherited from its SUPERTYPE). It is assumed to be untapered, unthreaded with neither a point or head; i.e. it is a smooth cylinder. The pin may also be given a description of its type.

EXPRESS specification:

```

*)
ENTITY fastener_simple_pin
SUBTYPE OF (fastener_simple);
    pin_type : OPTIONAL text;
END_ENTITY;
(*

```

Attribute definitions:*pin_type*

An optional text description of the type of the pin.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.20 fastener_simple_screw

Entity definition:

A type of simple fastener. The SUBTYPEs of this entity allow the screw to be described as either a machine screw (fully threaded and untapered), or a tapered screw. Other SUBTYPEs allow it to be described as countersunk, self-drilling and self-tapping.

EXPRESS specification:

```

*)
ENTITY fastener_simple_screw
SUPERTYPE OF (ONEOF
    (fastener_simple_screw_machine,
     fastener_simple_screw_tapered) ANDOR
    fastener_simple_screw_self_drilling ANDOR
    fastener_simple_screw_self_tapping)
SUBTYPE OF (fastener_simple);
    screw_type : OPTIONAL text;
    screw_drive_type : OPTIONAL text;
    screw_point_type : OPTIONAL text;
    screw_head_height : OPTIONAL positive_length_measure_with_unit;
    full_section_area : OPTIONAL area_measure_with_unit;
    reduced_section_area : OPTIONAL area_measure_with_unit;
END_ENTITY;
(*

```

Attribute definitions:

screw_type

An optional text description of the type of the screw; e.g. “right handed”, or “single threaded”.

screw_drive_type

An optional text description of the system used to drive in the screw to its final position; e.g. “hand driven” or “power driven”, “slotted” or “cross-headed”.

screw_point_type

An optional text description of the shape of the tip of the screw.

screw_head_height

Declares the instance of *positive_length_measure_with_unit* that may be associated with the *fastener_simple_screw*, which specifies the numerical value with an appropriate unit of the head height of the screw (as defined by the appropriate standard or code of practice). This is measured along the longitudinal axis of the screw.

full_section_area

Declares the first instance of *area_measure_with_unit* that may be associated with the *fastener_simple_screw*, which specifies the numerical value with an appropriate unit of the full sectional area of the screw’s shaft (as defined by the appropriate standard or code of practice).

reduced_section_area

Declares the second instance of *area_measure_with_unit* that may be associated with the *fastener_simple_screw*, which specifies the numerical value with an appropriate unit of the reduced sectional area of the screw's shaft (as defined by the appropriate standard or code of practice). This value should take into account the screw's thread and any other modifications to the screw that removes material from its shaft.

Formal propositions:**ANDOR SUPERTYPE**

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Informal propositions:

Although all the attributes are optional, it is expected that at least one attribute shall be given a value when this entity is instanced.

Notes:

New for CIS/2.

See diagram 42 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.21 fastener_simple_screw_machine**Entity definition:**

A type of simple fastener that can be described as a 'machine screw'. It is assumed to be untapered and fully threaded. (This is not the same as a "fully threaded bolt", whose dimensions are specified using a different code of practice.)

EXPRESS specification:

```
*)
ENTITY fastener_simple_screw_machine
  SUBTYPE OF (fastener_simple_screw);
  WHERE
    WRF29 : NOT ('STRUCTURAL_FRAME_SCHEMA.
      FASTENER_SIMPLE_CURVED' IN TYPE OF(SELF));
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE *fastener_simple_screw*. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity *fastener_simple_screw*.

Formal propositions:**WRF29**

A *fastener_simple_screw_machine* shall not be of the type *fastener_simple_curved*. In other words, a machine screw cannot be curved.

Notes:

New for CIS/2.

See diagram 42 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.22 fastener_simple_screw_self_drilling

Entity definition:

A type of simple fastener that is able to form the hole into which it is placed; i.e. a ‘self-drilling screw’ of the type shown in Figure 10.7. For a self-drilling, self-tapping screw, this entity would be instantiated with a fastener_simple_screw_self_tapping.

EXPRESS specification:

*)

ENTITY fastener_simple_screw_self_drilling

SUBTYPE OF (fastener_simple_screw);

hole_cutting_method : cutting_type;

pilot_hole_diameter : OPTIONAL positive_length_measure_with_unit;

drill_diameter : OPTIONAL positive_length_measure_with_unit;

WHERE

WRF8 : NOT((pilot_hole_diameter.value_component > drill_diameter.value_component)
AND (EXISTS (pilot_hole_diameter) AND EXISTS (drill_diameter)));

END_ENTITY;

(*

Attribute definitions:

hole_cutting_method

Declares the class of the method of cutting that the screw uses to form the hole. It would normally be classed as ‘drilled’ although the hole could also be formed by ‘abrasion’ if no material is removed from the surfaces of the hole.

pilot_hole_diameter

Declares the first instance of positive_length_measure_with_unit that may be associated with the fastener_simple_screw_self_drilling, which provides the numerical value with an appropriate unit of the diameter of the pilot hole that needs to be drilled before the screw is inserted to drill the final hole. If the screw is placed directly onto the material to self-drill, a pilot hole is not required and this attribute is left as a NULL (\$) value. This is separate from the nominal_diameter inherited from the SUPERTYPE fastener_simple.

drill_diameter

Declares the second instance of positive_length_measure_with_unit that may be associated with the fastener_simple_screw_self_drilling, which provides the numerical value with an appropriate unit of the diameter of the hole that is drilled by the screw. This is separate from the nominal_diameter inherited from the SUPERTYPE fastener_simple.

Formal propositions:

WRF8

If both attributes pilot_hole_diameter and drill_diameter are populated, then the numerical value associated with the instance of positive_length_measure_with_unit referenced by the attribute pilot_hole_diameter must not be greater than the numerical value associated with the instance of positive_length_measure_with_unit referenced by the attribute drill_diameter. In other words, if provided, the pilot hole should be smaller than the drill diameter. It is assumed that the both attributes are measured in the same units.

Informal propositions:

Both of the attributes `pilot_hole_diameter` and `drill_diameter` are measured in the same units.

For a self-drilling, self-tapping screw the attributes `cutting_method` and `pilot_hole_diameter` will be populated twice (once for each of the two SUBTYPEs) as the attributes have different meanings in the two `fastener_simple_screw_self_drilling` and `fastener_simple_screw_self_tapping`. In such a case, the `pilot_hole_diameter` for the self-tapping screw is set equal to the `drill_diameter` for the self-drilling screw.

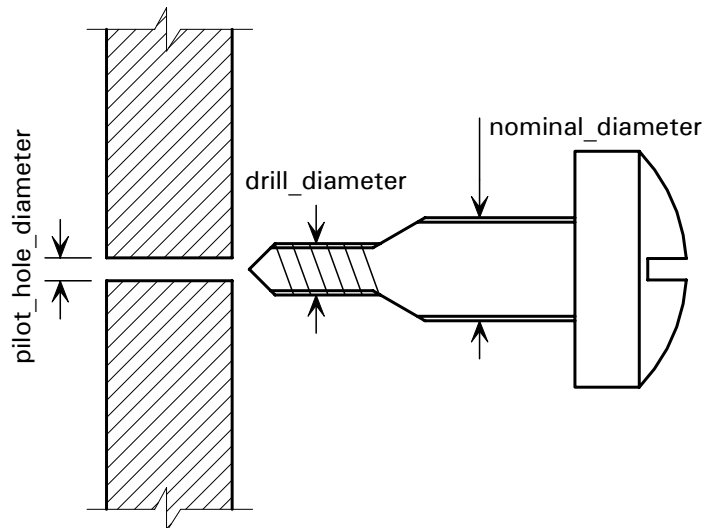


Figure 10.7 An example of a 'self-drilling screw'

Notes:

New for CIS/2.

See diagram 42 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.23 fastener_simple_screw_self_tapping**Entity definition:**

A type of simple fastener that is able to form the thread in the hole into which it is placed. For a self-drilling, self-tapping screw this entity would be instantiated with a `fastener_simple_screw_self_drilling`.

EXPRESS specification:

*)

ENTITY `fastener_simple_screw_self_tapping`

SUBTYPE OF (`fastener_simple_screw`);

`thread_cutting_method` : `cutting_type`;

`pilot_hole_diameter` : OPTIONAL `positive_length_measure_with_unit`;

END_ENTITY;

(*)

Attribute definitions:***thread_cutting_method***

Declares the class of the method of cutting that the screw uses to form the thread within the hole. It would normally be classed as ‘drilled’ if material is removed from the sides of the hole, although the hole could also be classed as ‘sheared’ if material is not removed from the hole sides.

pilot_hole_diameter

Declares the instance of `positive_length_measure_with_unit` that may be associated with the `fastener_simple_screw_self_tapping`, which provides the numerical value with an appropriate unit of the diameter of the pilot hole that needs to be drilled before the screw is inserted to cut the thread of the final hole. This is separate from the `nominal_diameter` inherited from the SUPERTYPE `fastener_simple`.

Informal propositions:

For a self-drilling, self-tapping screw the attribute `pilot_hole_diameter` may be populated twice (once for each of the two SUBTYPES) as the attribute has one meaning in `fastener_simple_screw_self_drilling` and a different meaning in `fastener_simple_screw_self_tapping`. If the `pilot_hole_diameter` is populated for both SUBTYPES then the `pilot_hole_diameter` for the self-tapping screw is set equal to the `drill_diameter` for the self-drilling screw.

Notes:

New for CIS/2.

See diagram 42 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.24 fastener_simple_screw_tapered**Entity definition:**

A type of simple screw that has a tapered shaft; i.e. a tapered screw of the type shown in Figure 10.8. The taper may be specified in absolute or relative terms.

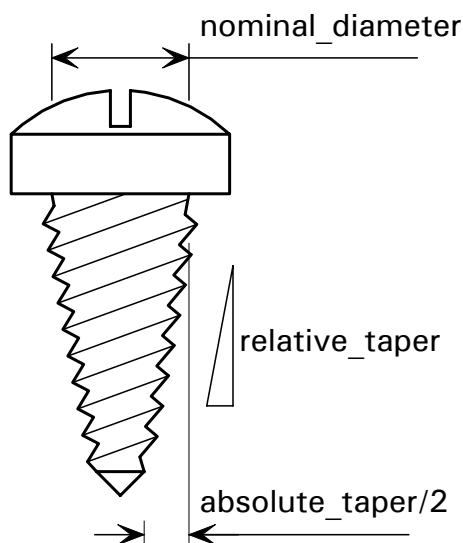


Figure 10.8 *An example of a ‘tapered screw’*

EXPRESS specification:

```

*)
ENTITY fastener_simple_screw_tapered
SUBTYPE OF (fastener_simple_screw);
    absolute_taper : OPTIONAL length_measure_with_unit;
    relative_taper : OPTIONAL ratio_measure_with_unit;
WHERE
    WRF9 : EXISTS (absolute_taper) OR EXISTS (relative_taper);
END_ENTITY;
(*

```

Attribute definitions:*absolute_taper*

Declares the instance of `length_measure_with_unit` that may be associated with the `fastener_simple_screw_tapered`, which provides the numerical value with an appropriate unit of the absolute taper of the shaft of the screw. For example, if the absolute taper for a screw with a nominal diameter of 18mm is specified as being 4mm, the diameter of the screw at its end is 14mm. This excludes the point of the screw. (See Figure 10.8.)

relative_taper

Declares the instance of `ratio_measure_with_unit` that may be associated with the `fastener_simple_screw_tapered`, which provides the numerical value with an appropriate unit of the relative taper of the shaft of the screw. For example, if the absolute taper for a screw with a nominal length of 150mm is specified as being 10%, the diameter of the screw's shaft at its end is 15mm smaller than its diameter at its head. This excludes the point of the screw. (See Figure 10.8.)

Formal propositions:*WRF9*

Either one of the attributes `absolute_taper` or `relative_taper` must be populated.

Notes:

New for CIS/2.

See diagram 42 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.25 fastener_simple_shear_connector**Entity definition:**

A type of simple fastener that acts as a shear connector. This type of fastener would normally be used with composite construction where, for example, the steel beam acts with the concrete slab it carries.

EXPRESS specification:

```

*)
ENTITY fastener_simple_shear_connector
SUBTYPE OF (fastener_simple);
    head_shape : OPTIONAL text;
    connector_type : OPTIONAL text;
    connection_method : OPTIONAL text;

```

END_ENTITY;

(*

Attribute definitions:

head_shape

An optional text description of the shape of the head of the shear connector.

connector_type

An optional text description of the shape of the type of the shear connector.

connection_method

An optional text description of the method used with the shear connector; e.g. “through-deck welded” or “shot-fired”.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.26 fastener_simple_stud

Entity definition:

A type of simple fastener with a threaded portion at both ends of its shaft and a plain (unthreaded) portion. The stud is assumed to be headless and untapered. This entity may also be used to represent a threaded bar, in which case, the threaded portion may extend the full length of the stud.

(The shear stud used in composite construction should be represented by the entity *fastener_simple_shear_connector* and not the *fastener_simple_stud*.)

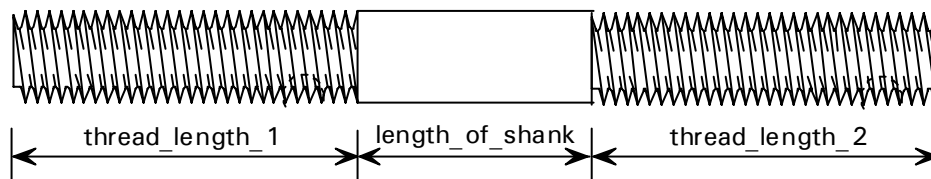


Figure 10.9 *Example fastener_simple_stud*

EXPRESS specification:

*)

ENTITY fastener_simple_stud

SUBTYPE OF (fastener_simple);

thread_length_1 : positive_length_measure_with_unit;

thread_length_2 : OPTIONAL positive_length_measure_with_unit;

length_of_shank : OPTIONAL positive_length_measure_with_unit;

full_section_area : OPTIONAL area_measure_with_unit;

reduced_section_area : OPTIONAL area_measure_with_unit;

DERIVE

thread_length_value_1 : REAL := thread_length_1.value_component;


```

    thread_length_value_2 : REAL := NVL(thread_length_2.value_component, 0.0);
    length_of_shank_value : REAL := NVL(length_of_shank.value_component, 0.0);
WHERE
    WRF33 : NOT (EXISTS(SELF\fastener_simple.nominal_length) AND (
        (thread_length_value_1 + thread_length_value_2 + length_of_shank_value) >
        (SELF\fastener_simple.nominal_length.value_component) ) );
END_ENTITY;
(*)

```

Attribute definitions:

thread_length_1

Declares the first instance of *positive_length_measure_with_unit* associated with the *fastener_simple_stud*, which specifies the numerical value with an appropriate unit of the linear dimension of the length of the threaded portion of the stud at the first end. The dimension is measured along the longitudinal axis of the stud. The first threaded portion corresponds to the position on the shaft where *x* has its minimum value.

When representing a typical stud, the value associated with this attribute will be less than the *nominal_length* of the fastener. When representing a fully threaded bar, the value associated with this attribute should be equal to the *nominal_length* of the fastener. (The attribute *nominal_length* is inherited from the SUPERTYPE entity *fastener_simple*.)

thread_length_2

Declares the second instance of *positive_length_measure_with_unit* that may be associated with the *fastener_simple_stud* to specify the numerical value with an appropriate unit of the linear dimension of the length of the threaded portion of the stud at the second end. The dimension is measured along the longitudinal axis of the stud. The second threaded portion corresponds to the position on the shaft where *x* has its maximum value. If this portion of the stud is unthreaded, this attribute shall be assigned a NULL value.

When representing a typical stud, the value associated with this attribute will be less than the *nominal_length* of the fastener.

length_of_shank

Declares the third instance of *positive_length_measure_with_unit* that may be associated with the *fastener_simple_stud* to specify the numerical value with an appropriate unit of the linear dimension of the length of the unthreaded portion of the stud. The dimension is measured along the longitudinal axis of the stud.

When representing a typical stud, the value associated with this attribute will be less than the *nominal_length* of the fastener. When representing a fully threaded bar, this attribute shall be assigned a NULL value.

full_section_area

Declares the first instance of *area_measure_with_unit* that may be associated with the *fastener_simple_stud*, which specifies the numerical value with an appropriate unit of the full sectional area of the stud's shaft (as defined by the appropriate standard or code of practice).

reduced_section_area

Declares the second instance of *area_measure_with_unit* that may be associated with the *fastener_simple_stud*, which specifies the numerical value with an appropriate unit of the reduced sectional area of the stud (as defined by the appropriate standard or code of

practice). This value should take into account the stud's thread and any other modifications to the stud that removes material from its shaft.

thread_length_value_1

Derives the real number value of the length of the threaded portion of the stud at the first end from the attribute thread_length_1.

thread_length_value_2

Derives the real number value of the length of the threaded portion of the stud at the second end from the attribute thread_length_2. If the attribute thread_length_2 has not been populated, the thread_length_value_2 is assigned a value of zero.

length_of_shank_value

Derives the real number value of the length of the unthreaded portion of the stud from the attribute length_of_shank. If the attribute length_of_shank has not been populated, the length_of_shank_value is assigned a value of zero.

Formal propositions:

WRF33

If a nominal length is specified, its value shall be greater than the sum of the 2 thread lengths and the shank.

Notes:

New for CIS/2.

See diagram 40 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

10.3.27 fastener_simple_washer

Entity definition:

A type of simple fastener that provides the description of a washer. A washer can be specified once, and then used in many fastener mechanisms. A fastener mechanism may include more than one washer. This entity provides the description of a plain flat washer. Tapered and load-indicating washers are represented by the SUBTYPES.

EXPRESS specification:

*)

ENTITY fastener_simple_washer

SUPERTYPE OF (ONEOF

(fastener_simple_washer_tapered, fastener_simple_washer_load_indicating))

SUBTYPE OF (fastener_simple);

washer_shape : OPTIONAL text;

inside_diameter : OPTIONAL positive_length_measure_with_unit;

external_dimension : OPTIONAL positive_length_measure_with_unit;

DERIVE

washer_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;

WHERE

WRF10 : NOT((EXISTS (inside_diameter) AND EXISTS (external_dimension)) AND
(inside_diameter.value_component > external_dimension.value_component));

WRF11 : NOT((EXISTS (inside_diameter)) AND

```

        (inside_diameter.value_component <
        (SELF\fastener_simple.nominal_diameter.value_component));
END_ENTITY;
(*)

```

Attribute definitions:

washer_shape

An optional text description of the shape of the washer; e.g. “circular, flat”.

inside_diameter

Declares the first instance of `positive_length_measure_with_unit` that may be associated with the `fastener_simple_washer`, which specifies the numerical value with an appropriate unit of the diameter of the washer’s hole. The dimension is measured perpendicular to the longitudinal axis of the washer in accordance with the appropriate standard or code of practice. This is **not** the nominal diameter of the washer.

external_dimension

Declares the second instance of `positive_length_measure_with_unit` associated with the `fastener_simple_washer`, which specifies the numerical value with an appropriate unit of the external dimension of the washer. The dimension is measured perpendicular to the longitudinal axis of the washer in accordance with the appropriate standard or code of practice. This is **not** the nominal diameter of the washer. For circular washers, this attribute represents the external diameter of the washer. For square washers, this attribute represents the length of the side of that square.

washer_ref

A bag of short alphanumerical identifying references for the washer derived from the attribute `item_ref` from the SUPERTYPE structural `frame_item`. If a standard reference is specified for a washer it should be represented using the attribute `ref` in the entity `item_reference`. A relationship is then made using the entity `item_assignment`. If the washer is not assigned an item reference, an empty bag will be returned here.

In most cases, a washer will be assigned only one reference and that would be taken from a list of standard references from a flavour file. However, there may be times when users wish to add their own references for the washer. These could be taken from an industry agreed library, or from a proprietary list.

Formal propositions:

WRF10

If both of the attributes `inside_diameter` and `external_diameter` are populated, then the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `inside_diameter` must not be greater than the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `external_diameter`. In other words, the external diameter of a washer should be greater than the inside diameter. It is assumed that both attributes are measured using the same units.

WRF11

If the attribute `inside_diameter` is populated, then the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `inside_diameter` must not be less than the numerical value associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `nominal_diameter`.

(inherited from the SUPERTYPE `fastener_simple`). In other words, the nominal diameter of a washer should be smaller than the inside diameter. It is assumed that both attributes are measured using the same units.

Informal propositions:

The attributes are `inside_diameter`, `external_diameter`, and `nominal_diameter` (inherited from the SUPERTYPE `fastener_simple`) are all measured using the same units. This implies that the instances of `positive_length_measure_with_unit` referenced by these attributes should all reference the same instance of `length_unit`.

Notes:

Known as WASHER in CIS/1.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE clause added.

10.3.28 fastener_simple_washer_load_indicating

Entity definition:

A type of washer whose otherwise flat surfaces have been given raised portions which deform at a known rate under load. The load applied to the washer is indicated by the closing of the gap between the washer and the mating surface. The type of surface deformations on the washer and how the washers are used are often particular to a manufacturer.

EXPRESS specification:

```
*)
ENTITY fastener_simple_washer_load_indicating
SUBTYPE OF (fastener_simple_washer);
    final_gap : positive_length_measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:

final_gap

Declares the instance of `positive_length_measure_with_unit` associated with the `fastener_simple_washer_load_indicating`, which provides the numerical value with the appropriate unit of the linear dimension between the underside of the washer and the mating surface. This final gap dimension is measured along the longitudinal axis of the washer in accordance with the appropriate standard or code of practice.

Notes:

New for CIS/2.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.29 fastener_simple_washer_tapered

Entity definition:

A type of washer that does not have a constant thickness; i.e. a ‘tapered washer’ of the type shown in Figure 10.10.

EXPRESS specification:

```

*)
ENTITY fastener_simple_washer_tapered
SUBTYPE OF (fastener_simple_washer);
    taper : ratio_measure_with_unit;
END_ENTITY;
(*)

```

Attribute definitions:**taper**

Declares the instance of `ratio_measure_with_unit` associated with the `fastener_simple_washer_tapered` which provides the numerical value with the appropriate unit of the slope of the taper. This value is independent of the diameter and length of the washer. For example, a washer with an external diameter of 50mm and a 8% taper would have a 4mm difference between its maximum and minimum thicknesses. If its nominal length was given as 7mm, the washer would be 9mm thick at its maximum edge and 5mm thick on its minimum edge. The maximum and minimum thicknesses are measured in the direction of the longitudinal axis of the washer.

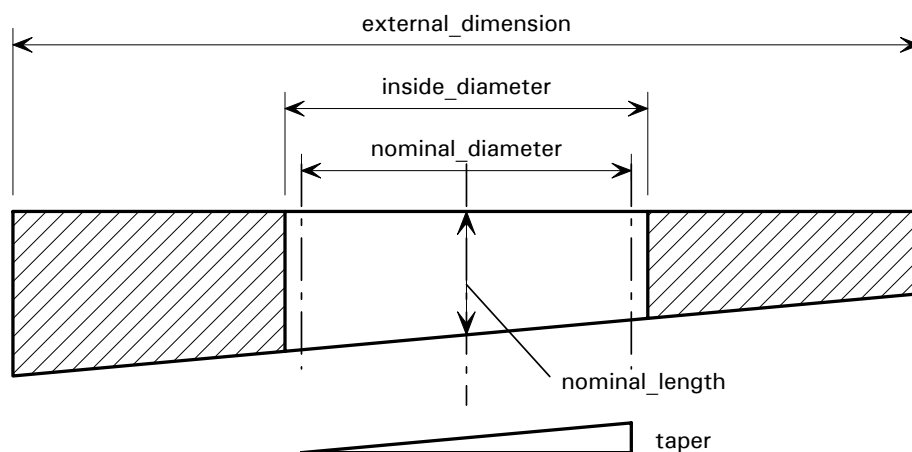


Figure 10.10 An example of a 'tapered washer'

Notes:

New for CIS/2.

See diagram 41 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.30 joint_system**Entity definition:**

A joint system is an abstract specific level entity that represents the combination of the means by which a structural connection is realized, together with the geometric description of that connection. The joint system is categorized (through its SUBTYPES) as either welded, mechanical, chemical, or amorphous. Where joint systems involve the use of one or more joint systems (e.g. welded studs) these are categorized as complex and are represented by the SUBTYPE `joint_system_complex`. (See SUBTYPES for further definitions.)

As a SUBTYPE of `structural_frame_item`, a `joint_system` inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a `joint_system` may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`) – this allows the appropriate national standard or code of practice used in the definition of the joint system to be described in detail
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A `joint_system` may also be related to another `joint_system` (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

*)

ENTITY `joint_system`

SUPERTYPE OF (ONEOF

(`joint_system_amorphous`,
`joint_system_chemical`,
`joint_system_complex`,
`joint_system_mechanical`,
`joint_system_welded`))

SUBTYPE OF (`structural_frame_item`);

`place_of_assembly` : OPTIONAL `shop_or_site`;

DERIVE

`joint_system_number` : INTEGER := SELF\`structural_frame_item`.`item_number`;

`joint_system_name` : LABEL := SELF\`structural_frame_item`.`item_name`;

`design_uses` : SET [0:?] OF `design_joint_system` := bag_to_set
(USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
DESIGN_JOINT_SYSTEM.DESIGN_JOINT_SYSTEM_SPEC'));

`physical_uses` : SET [0:?] OF `located_joint_system` := bag_to_set
(USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
LOCATED_JOINT_SYSTEM.DESCRPTIVE_JOINT_SYSTEM'));

UNIQUE

URJ1 : `joint_system_number`,`joint_system_name`;

END_ENTITY;

(*

Attribute definitions:

`place_of_assembly`

Declares the class of the joint system as either being assembled in the fabrication shop, or on site.

joint_system_number

Derives the reference number for the joint system from the attribute *item_number* in the SUPERTYPE *structural_frame_item*.

joint_system_name

Derives the reference name for the joint system from the attribute *item_name* in the SUPERTYPE *structural_frame_item*.

design_uses

Derives the set of zero, one or more instances of *design_joint_system* that reference the *joint_system* via the attribute *design_joint_system_spec*.

physical_uses

Derives the set of zero, one or more instances of *located_joint_system* that reference the *joint_system* via the attribute *descriptive_joint_system*.

Formal propositions:*URJ1*

The combination of the values of the attributes *joint_system_number* and *joint_system_name* shall be unique to this instance of *joint_system*; i.e. the *joint_system_number* and *joint_system_name* form a unique reference for the *joint_system*.

Notes:

Known as S_JOINT_SYSTEM in CIS/1.

This entity can be thought of as a use of the ‘type’ verses ‘token’ approach seen in many database applications; with the *joint_system* entity representing a ‘type’ of joint system and the *design_joint_system* and *located_joint_system* entities representing ‘tokens’ of that joint system. This is clarified by the derived attributes *design_uses* and *physical_uses*, which derive the use of the *joint_system* in the design model and physical model, respectively.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE and UNIQUE clauses added.

10.3.31 joint_system_amorphous**Entity definition:**

A type of joint system that has no defined form. This is used where the chemical mechanism (adhesive, grout, filler, or sealant) is poured or pumped into the space between two parts. The exact shape of the space is not of concern.

EXPRESS specification:

*)

ENTITY *joint_system_amorphous*

SUBTYPE OF (*joint_system*);

specification : *chemical_mechanism*;

END_ENTITY;

(*

Attribute definitions:*specification*

Declares the instance of `chemical_mechanism` associated with the `joint_system_amorphous`, which is used to specify, for example, the ‘gap filling material’ (adhesive, grout, filler, or sealant). There must be one (and only one) instance of `chemical_mechanism` referenced by each instance of `joint_system_amorphous`.

Notes:

New for CIS/2.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.32 joint_system_chemical**Entity definition:**

A type of joint system where two surfaces are specified as being chemically joined together using a number of layers of adhesive, grout, filler, or sealant.

EXPRESS specification:

```

*)
ENTITY joint_system_chemical
SUBTYPE OF (joint_system);
    joining_surface : surface;
    joined_surface : surface;
    specification : LIST [1:?] OF chemical_mechanism;
DERIVE
    number_of_layers : INTEGER := SIZEOF(specification);
WHERE
    WRJ1 : joining_surface :<>: joined_surface;
END_ENTITY;
(*

```

Attribute definitions:*joining_surface*

Declares the first instance of surface associated with the `joint_system_chemical`, which specifies the bounding shape of the first face involved in the joint system.

joined_surface

Declares the second instance of surface associated with the `joint_system_chemical`, which specifies the bounding shape of the first face involved in the joint system.

specification

Declares the list of one or more instances of `chemical_mechanism` associated with the `joint_system_chemical`, which provide the specification for each layer of adhesive, grout, filler or sealant applied between the two joined surfaces. The size of the list of instances associated with each instance of `joint_system_chemical` shall be equal to the value of the `number_of_layers` attribute. The LIST data type implies that order is important. It is implied that each instance in the list provides the specification for the corresponding layer in the sequence.

If more than one chemical_mechanism is specified, each layer is applied in the order of the LIST starting from the joining_surface to the joined_surface.

number_of_layers

This attribute derives the numerical value of the number of layers of adhesive, grout, filler, or sealant applied between the two joined surfaces from the size of the aggregation associated with the attribute specification.

Formal propositions:

DERIVE

The derived attribute number_of_layers does not appear in the STEP Part 21 file.

WRJ1

The instance of surface referenced by the attribute joined_surface shall not be the same instance of surface referenced by the attribute joining_surface. In other words, the two surfaces joined by this joint_system_chemical shall be distinct and separate.

Informal propositions:

Each instance in the list given by the attribute specification provides the appropriate specification for the layer of adhesive, grout, filler, or sealant in the sequence in which the layers are applied.

Where the instance of joint_system_chemical is used to join the surfaces of two parts, the instances of surface associated with the attributes joining_surface and joined_surface should be items referenced by the shape_representation associated the two complex parts. Similarly, where two assemblies are glued together, the instances of surface should be items referenced by the shape_representation associated with the two instances of assembly_with_shape.

Notes:

New for CIS/2.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.33 joint_system_complex

Entity definition:

A type of joint system that is defined using an ordered list of other joint systems. There must be at least one instance of joint_system referenced by each instance of joint_system_complex. This entity would be used to represent, for example, welded studs or welded shear connectors.

EXPRESS specification:

*)

ENTITY joint_system_complex

SUBTYPE OF (joint_system);

systems : LIST [1:?] OF joint_system;

WHERE

WRJ2 : SIZEOF(QUERY(joint <* systems | joint :=: (SELF))) = 0;

END_ENTITY;

(*

Attribute definitions:**systems**

Declares the list of instances of joint system associated with the joint_system_complex which provides the specifications of the joint_systems that define the joint_system_complex. There must be at least one instance of joint_system referenced by each instance of joint_system_complex. One or more of those instances may be another instance of joint_system_complex, but it must not be the same instance; that is, entity type recursion is permissible, but instance recursion is not.

Formal propositions:**WRJ2**

The size of the set of instances of joint_system that are instance equal to this joint_system_complex shall equal zero. In other words, an instance of joint_system_complex must not reference itself. Entity type recursion is permissible, but instance recursion is not.

Notes:

New for CIS/2.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

10.3.34 joint_system_mechanical**Entity definition:**

A type of joint_system that is defined using a single instance of fastener_mechanism which is assumed to be placed with its longitudinal axis at each of the points declared by its layout. This entity would be used to represent, for example, a bolt group containing 4 bolts, each with an associated nut and washer. This would be done by associating the fastener_mechanism with the set of 4 points. The fastener_mechanism itself would be defined as the bolt, nut and washer.

It is assumed that each of the fasteners in the fastener_mechanism share a common longitudinal axis defined by the longitudinal axis of the first fastener in the mechanism – the bolt. The origin of this axis is defined as the extreme end of the first fastener (e.g. the top of the head of bolt). To reposition this origin (e.g. to set the origin of the faster_mechanism as the underside of the bolt head), the attribute mechanism needs to refer to an instance of the entity fastener_mechanism_with_position.

EXPRESS specification:

```
*)
ENTITY joint_system_mechanical
  SUBTYPE OF (joint_system);
    layout_points : LIST [1:?] OF point;
    mechanism : fastener_mechanism;
END_ENTITY;
(*
```

Attribute definitions:*layout_points*

Declares the list of one or more instances of point associated with the `joint_system_mechanical`, which specify the position of each copy of the `fastener_mechanism`. There must be at least one instance of point referenced by each instance of `joint_system_mechanical`. The point is a STEP Part 42 geometric entity which has many different SUBTYPES.

mechanism

Declares the instance of `fastener_mechanism` associated with the `joint_system_mechanical` that provides the specification of the fasteners that make up the mechanical aspects of the joint system. There must be one (and only one) instance of `fastener_mechanism` referenced by each instance of `joint_system_mechanical`.

Notes:

New for CIS/2; partially addressed by BOLT_SYSTEM in CIS/1.

The entity point and its SUBTYPES are defined in STEP Part 42.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

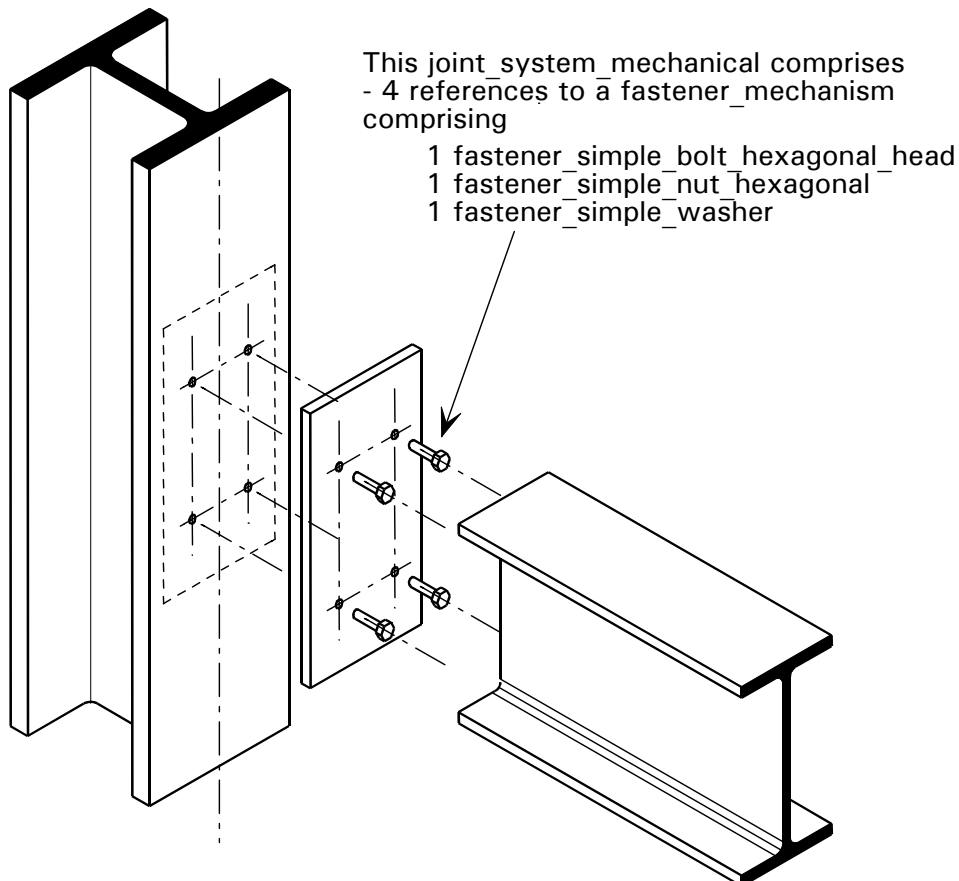


Figure 10.11 *Example of a joint_system_mechanical*

10.3.35 joint_system_welded

Entity definition:

A type of joint system that is defined using a weld_mechanism. The SUBTYPEs of this entity define the point, surface or curve associated with the weld_mechanism. The weld_mechanism provides the specification for the weld.

EXPRESS specification:

```
*)
ENTITY joint_system_welded
SUPERTYPE OF (ONEOF
    (joint_system_welded_point,
     joint_system_welded_linear,
     joint_system_welded_surface,
     joint_system_welded_with_shape))
SUBTYPE OF (joint_system);
    weld_specification : weld_mechanism;
END_ENTITY;
(*
```

Attribute definitions:

weld_specification

Declares the instance of weld_mechanism associated with the joint_system_welded, which provides the specification for the weld. There must be one (and only one) instance of weld_mechanism referenced by each instance of joint_system_welded.

Informal propositions:

This entity is existence dependent on the entity weld_mechanism.

Notes:

Known as WELD_SYSTEM in CIS/1.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition. (Subtype added)

10.3.36 joint_system_welded_linear

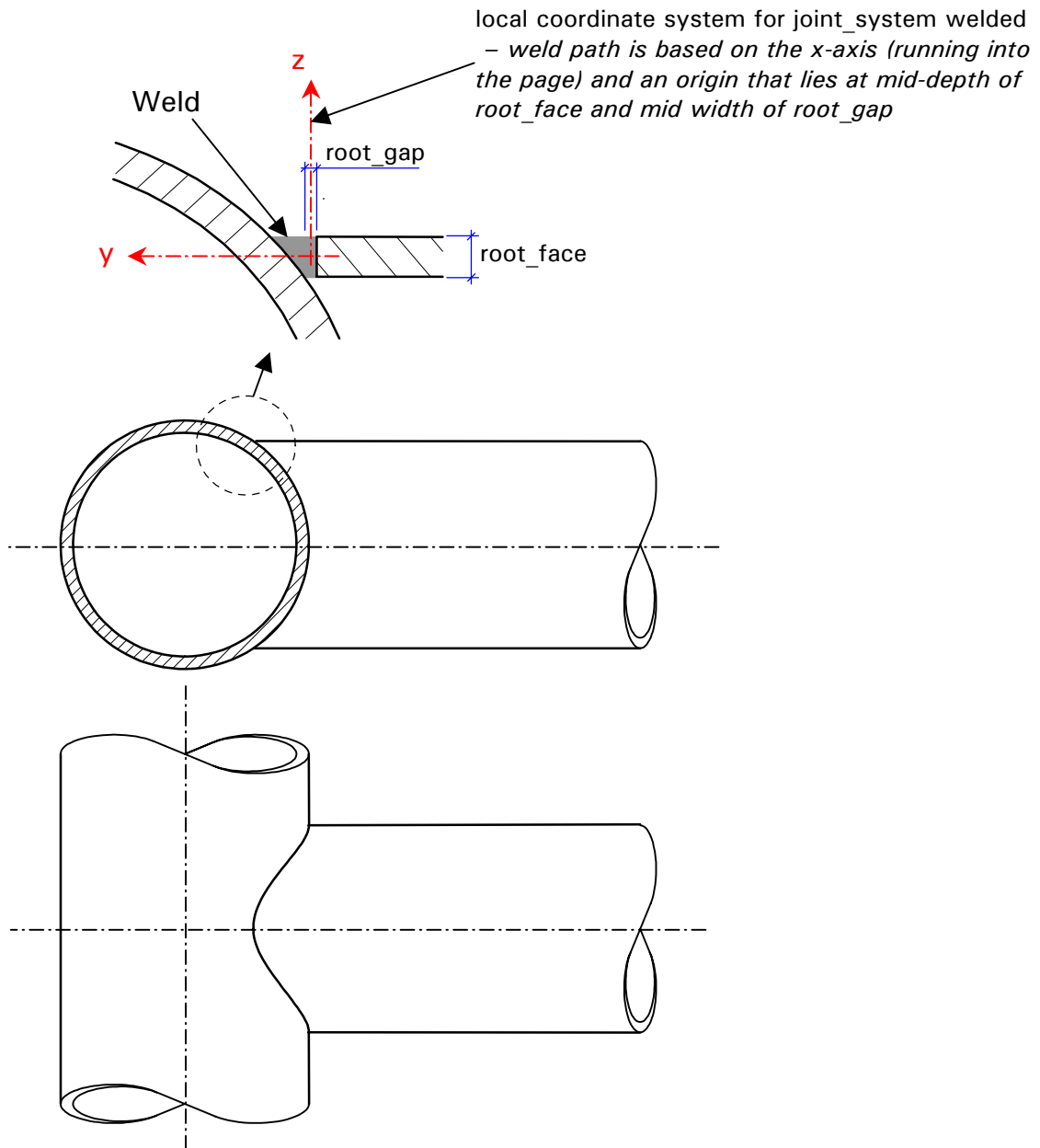
Entity definition:

A type of joint_system_welded that has an associated composite curve, which defines the path of the weld.

EXPRESS specification:

```
*)
ENTITY joint_system_welded_linear
SUBTYPE OF (joint_system_welded);
    weld_path : composite_curve;
WHERE
    WRJ3 : NOT((SELF\joint_system_welded.weld_specification.weld_mechanism_type =
                SPOT_WELD) OR
                (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
                PLUG_WELD));
END_ENTITY;
```

(*)



```

joint_system_welded_linear.weld_path = intersection_curve
weld_mechanism_groove.joint_configuration = tee_joint
weld_mechanism_groove.surface_shape = flush
weld_mechanism_groove_bevelled.end_cut_shape = flare_single_bevel
  
```

Figure 10.12 Example of a joint_system_welded_linear

Attribute definitions:

weld_path

Declares the instance of composite_curve associated with the joint_system_welded_linear, which provides the geometric definition of the weld path (where the weld will be placed). There must be one (and only one) instance of composite_curve referenced by each

instance of joint_system_welded. As can be seen in Figure 10.12, the weld path can be very complex.

Formal propositions:

WRJ3

The weld_mechanism_type of the instance of the entity weld_mechanism (referenced by the attribute weld_specification inherited from the SUPERTYPE entity joint_system_welded) shall not be assigned a value of either .PLUG_WELD. or .SPOT_WELD. In other words, a linear welded joint uses either a butt weld or a fillet weld.

Informal propositions:

This entity is existence dependent on the entity composite_curve.

Notes

New for CIS/2.

The entity composite_curve is defined in STEP Part 42.

See Diagram 39 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd edition – WHERE rule changed.

10.3.37 joint_system_welded_point

Entity definition:

A type of weld that is defined at one point (i.e. a spot weld).

EXPRESS specification:

*)

ENTITY joint_system_welded_point

SUBTYPE OF (joint_system_welded);

weld_position : point;

WHERE

WRJ4 : (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
SPOT_WELD) OR
(SELF\joint_system_welded.weld_specification.weld_mechanism_type =
PLUG_WELD) OR
(SELF\joint_system_welded.weld_specification.weld_mechanism_type =
STUD_WELD);

END_ENTITY;

(*

Attribute definitions:

weld_position

Declares the instance of point associated with the joint_system_welded_point, which defines the position of the spot weld.

Formal propositions:

WRJ4

The weld_mechanism_type of the instance of the entity weld_mechanism (referenced by the attribute weld_specification inherited from the SUPERTYPE entity joint_system_welded) shall be assigned a value of either .SPOT_WELD. or

.PLUG_WELD. or a .STUD_WELD. In other words, a point welded joint uses either a spot weld, a plug weld or a stud weld.

Informal propositions:

This entity is existence dependent on the entity point.

Notes

New for CIS/2.

The entity point is defined in STEP Part 42.

See Diagram 39 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd edition – WHERE rule changed.

10.3.38 joint_system_welded_surface

Entity definition:

A type of weld that is defined to cover a bounded area on a surface (e.g. a ‘plug’ weld as shown in Figure 10.13).

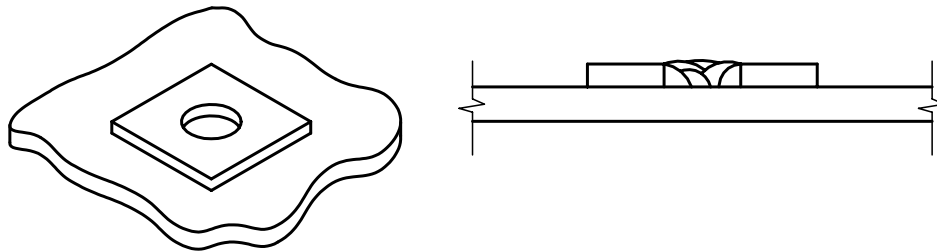


Figure 10.13 Example of a ‘plug weld’

EXPRESS specification:

*)

ENTITY joint_system_welded_surface

SUBTYPE OF (joint_system_welded);

weld_surface : bounded_surface;

WHERE

WRJ5 : (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
 PLUG_WELD) OR
 (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
 SURFACING_WELD) OR
 (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
 SLOT_WELD);

END_ENTITY;

(*

Attribute definitions:

weld_surface

Declares the instance of bounded_surface associated with the joint_system_welded_surface, which defines the geometry of the bounded area to which the weld is applied. There must be one, and only one, instance of bounded_surface associated with joint_system_welded_surface.

Formal propositions:**WRJ5**

The weld_mechanism_type of the instance of the entity weld_mechanism (referenced by the attribute weld_specification inherited from the SUPERTYPE entity joint_system_welded) shall be assigned a value of .PLUG_WELD., or a .SURFACING_WELD., or a SLOT_WELD. In other words, a surface welded joint uses a plug weld, a surfacing weld, or a slot weld.

Informal propositions:

This entity is existence dependent on the entity bounded_surface.

Notes

New for CIS/2.

The entity bounded_surface is defined in STEP Part 42.

See Diagram 39 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd edition – WHERE rule changed.

10.3.39 joint_system_welded_with_shape**Entity definition:**

A type of weld (joint_system_welded) whose shape is defined using the geometry constructs from STEP Part 42.

EXPRESS specification:

```
*)
ENTITY joint_system_welded_with_shape
  SUBTYPE OF (joint_system_welded);
    weld_shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:**weld_shape**

Declares the instance of shape_representation_with_units used to define the geometry of the welded joint.

Informal propositions:

This entity is existence dependent on the entity shape_representation_with_units.

Notes

New for CIS/2 2nd Edition

See Diagram 39 of the EXPRESS-G diagrams in Appendix B.

10.3.40 weld_mechanism**Entity definition:**

A type of structural frame product that provides the specification of a weld. This entity is normally used with the joint_system_welded. The weld can be categorized (through its subtypes) as either a complex, prismatic, fillet, groove, spot or seam weld.

As a SUBTYPE of structural_frame_product, a weld_mechanism inherits the three attributes item_number, item_name, and item_description (from structural_frame_item). It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a weld_mechanism may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the weld_mechanism to be described in detail
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A weld_mechanism may also be related to another weld_mechanism (or any other structural_frame_item) via the entity structural_frame_item_relationship.

As a SUBTYPE of structural_frame_product, an weld_mechanism may also be assigned a material (via the entity structural_frame_product_with_material).

EXPRESS specification:

*)

ENTITY weld_mechanism

SUPERTYPE OF (ONEOF(

weld_mechanism_complex,

weld_mechanism_prismatic,

weld_mechanism_fillet,

weld_mechanism_groove,

weld_mechanism_spot_seam))

SUBTYPE OF (structural_frame_product);

weld_mechanism_type : weld_type;

penetration : weld_penetration;

weld_dimension : OPTIONAL positive_length_measure_with_unit;

weld_dimension_name : OPTIONAL label;

weld_design_strength : OPTIONAL pressure_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:

weld_mechanism_type

Declares the classification of the weld_mechanism by the arrangement of the weld metal in relation to the joined parts (e.g. a butt weld, a fillet weld, a spot weld, etc). The weld_mechanism_type may be classified as undefined. See definition of the TYPE weld_type.

penetration

Declares the classification of the weld_mechanism by the extent of the penetration of the weld; i.e. full, partial, or deep. The penetration may be classified as undefined. (See Figure 10.14.)

weld_dimension

Declares the instance of `positive_length_measure_with_unit` associated with the `weld_mechanism`, which provides the numerical value with an appropriate unit of the dimension of the weld. This is the final dimension of the weld, regardless of the number of passes required to complete the weld. (See Figure 10.14.)

This attribute could represent, for example, the throat thickness or leg length of a linear fillet weld. The name of the dimension is given by the attribute `weld_dimension_name`.

There must be one (and only one) instance of `positive_length_measure_with_unit` referenced by each instance of `weld_mechanism`. An instance of `positive_length_measure_with_unit` can, of course, be used to specify the dimension of many instances of `weld_mechanism`.

Where the weld does not have a recognized ‘throat’, this attribute may be used to provide the maximum dimension of the weld measured perpendicular to the weld path and perpendicular to the exposed face of the weld. For example, where a plug weld is used to fill a hole, the throat thickness becomes the maximum thickness of the weld, and is equal to the depth of the hole.

weld_dimension_name

A human-readable reference for the `weld_dimension` given above. This attribute provides the name of the dimension being measured (or specified) by the attribute `weld_dimension`; e.g. “Throat thickness”, “Leg length”.

weld_design_strength

Declares the instance of `pressure_measure_with_unit` that may be associated with the `weld_mechanism` to provide the design strength of the weld. The design strength is specified or measured in accordance with the appropriate standard or code of practice.

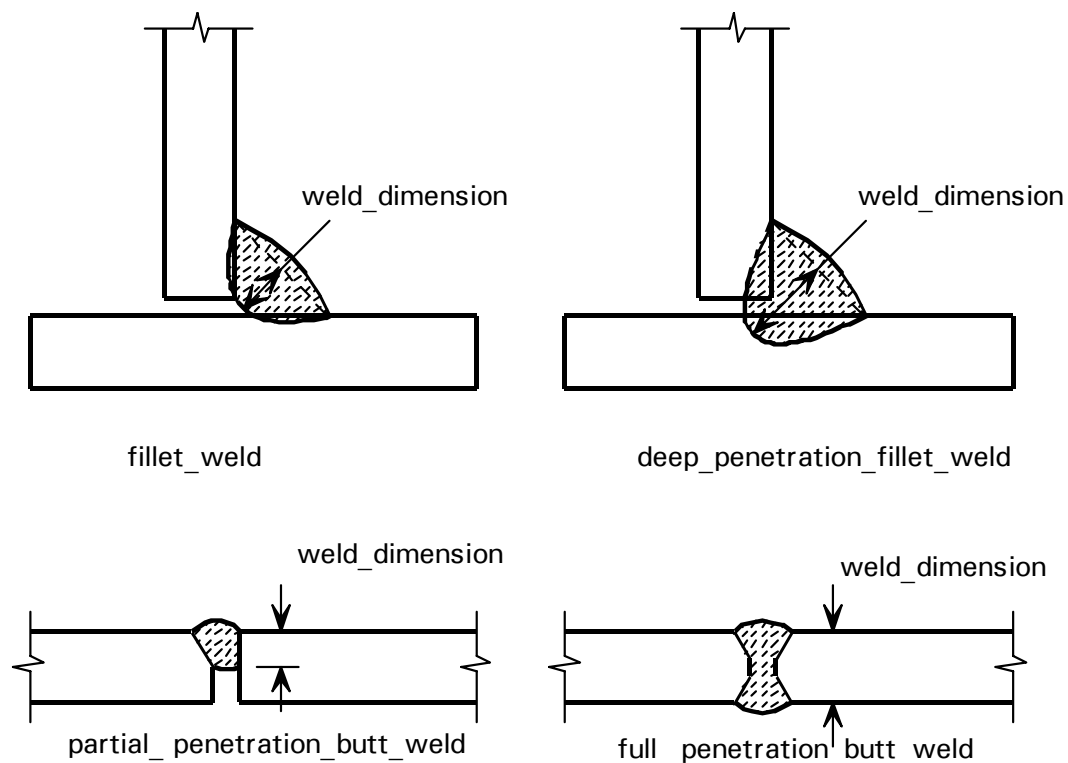


Figure 10.14 Examples of different weld types and penetrations



Notes:

Known as WELD_MECHANISM in CIS/1.

See diagram 39 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – SUBTYPEs added.

10.3.41 weld_mechanism_complex**Entity definition:**

A type of weld_mechanism whose shape is defined using the geometry constructs from STEP Part 42.

EXPRESS specification:

```
*)
ENTITY weld_mechanism_complex
SUBTYPE OF (weld_mechanism);
    weld_shape : shape_representation_with_units;
END_ENTITY;
(*
```

Attribute definitions:

weld_shape

Declares the instance of shape_representation_with_units used to define the geometry of the weld.

Informal propositions:

This entity is existence dependent on the entity shape_representation_with_units.

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.42 weld_mechanism_fillet**Entity definition:**

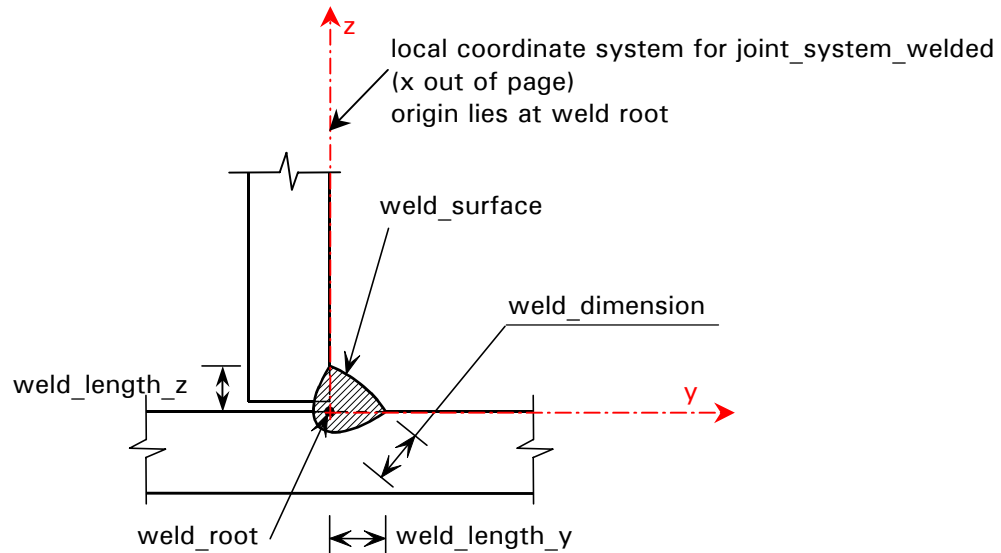
A type of weld_mechanism assigned a series of parameters used to define a fillet weld. A fillet weld is a weld where the weld bead is laid down along the outside of the part. The degree of penetration achieved by this type of weld is dependent on the thickness and arrangement of the parts being welded and usually, is less than that attainable with a groove weld. The advantage it offers in elimination of edge preparation makes this a highly desirable weld detail.

A fillet weld is used to make lap joints, corner joints, and T joints. Weld metal is deposited in a corner formed by the fit-up of the two members and penetrates and fuses with the base metal to form the joint.

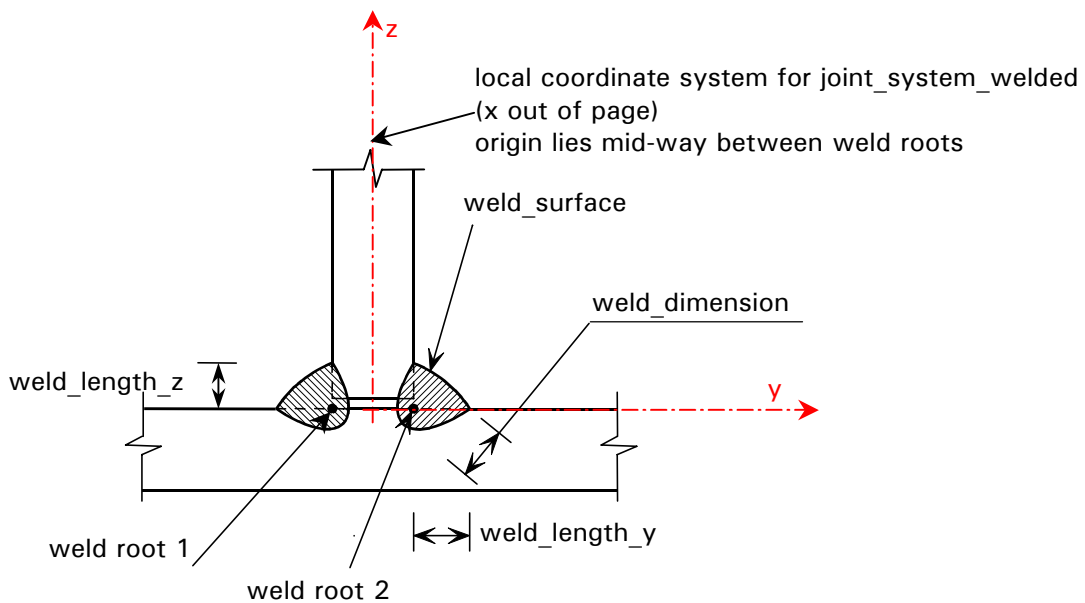
A weld_mechanism_fillet may be defined as continuous or intermittent through its subtypes.

A weld_mechanism_fillet may be specified with leg lengths in two directions. The throat thickness should be specified using the attribute weld_dimension inherited from the supertype weld_mechanism.

The local coordinate system for the `joint_system_welded` that uses this `weld_mechanism` has its origin at the root of the weld (for single welds) or mid way between the weld roots (for multiple welds). The weld path is based on this origin.



`weld_mechanism_fillet.sidedness` = one_side
`weld_mechanism_fillet.surface_shape` = convex
`weld_mechanism_fillet.joint_configuration` = tee_joint



`weld_mechanism_fillet.sidedness` = both_sides
`weld_mechanism_fillet.surface_shape` = convex
`weld_mechanism_fillet.joint_configuration` = tee_joint

Figure 10.15: *Examples of weld_mechanism_fillet*

Table 10.3: *Examples of weld_mechanism_fillet for butt, corner and tee joints*

sidedness	TYPE weld_configuration		
	butt_joint	corner_joint	tee_joint
one_side			
both_sides			
<p>Legend: l_y = leg_length_y, l_z = leg_length_z</p> <p>Local coordinate systems are shown with an x-axis running into the page and an origin that lies at the weld root (for single welds) or mid way between the weld roots (for multiple welds). Weld path is based on this origin.</p>			

Table 10.4: *Examples of weld_mechanism_fillet for butt, corner and tee joints*

sidedness	TYPE weld_configuration		
	lap_joint	edge_joint	cruciform_joint
one_side		-	
both_sides		-	
<p>Legend: l_y = leg_length_y, l_z = leg_length_z</p> <p>Local coordinate systems are shown with an x-axis running into the page and an origin that lies at the weld root (for single welds) or mid way between the weld roots (for multiple welds). Weld path is based on this origin.</p>			

EXPRESS specification:

```

*)
ENTITY weld_mechanism_fillet
SUPERTYPE OF (ONEOF(weld_mechanism_fillet_continuous,
    weld_mechanism_fillet_intermittent))
SUBTYPE OF (weld_mechanism);
    sidedness : weld_sidedness;
    surface_shape : weld_surface_shape;
    joint_configuration : weld_configuration;
    leg_length_y : OPTIONAL positive_length_measure_with_unit;
    leg_length_z : OPTIONAL positive_length_measure_with_unit;
WHERE
    WRW7 : SELF\weld_mechanism.weld_mechanism_type = FILLET_WELD;
    WRW8 : NOT ((SELF\weld_mechanism.penetration = FULL_PENETRATION) OR
        (SELF\weld_mechanism.penetration = PARTIAL_PENETRATION));
END_ENTITY;
(*

```

Attribute definitions:*sidedness*

An identifier as to whether the fillet weld is to be made at one side or at both sides of the welded joint.

endcut_shape_type

An indicator as to explain the endcut shape of parts being welded at a fillet weld joint.

surface_shape

An identifier as to the resulted top surface shape of the fillet weld as either flush, concave or convex.

joint_configuration

An identifier as to the arrangement of the pieces being joined by the weld_mechanism; i.e. butt, corner, tee, lap, edge, cruciform. The joint configuration may be declared as undefined. (See definition of the type weld_configuration.)

leg_length_y

Declares the instance of positive_length_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the length of the fillet weld in the y direction.

leg_length_z

Declares the instance of positive_length_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the length of the fillet weld in the z direction.

Formal propositions*WRW7*

The attribute weld_mechanism_type, inherited from the supertype weld_mechanism, shall be populated with the value FILLET_WELD.

WRW8

The attribute penetration, inherited from the supertype weld_mechanism, shall NOT be populated with the value FULL_PENETRATION or PARTIAL_PENETRATION.

Notes

New for 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.43 weld_mechanism_fillet_continuous**Entity definition:**

A type of weld_mechanism_fillet used to define a continuous fillet weld. This entity has no attributes of its own; it is merely used to distinguish between continuous and intermittent fillet welds.

EXPRESS specification:

```
*)
ENTITY weld_mechanism_fillet_continuous
SUBTYPE OF (weld_mechanism_fillet);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own, but inherits many from its SUPERTYPE.

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.44 weld_mechanism_fillet_intermittent**Entity definition:**

A type of weld_mechanism_fillet assigned a series of parameters used to define an intermittent (as opposed to a continuous) fillet weld. An intermittent fillet weld is one where the weld bead is deposited in a sequence of short continuous lengths, spaced at regular intervals. The combination of a weld length and spacing forms a pattern that is repeated along the length of the weld. The primary use of intermittent welds is to reduce distortion in thinner structural parts and to reduce manufacturing costs in lightly stressed members.

EXPRESS specification:

```
*)
ENTITY weld_mechanism_fillet_intermittent
SUBTYPE OF (weld_mechanism_fillet);
    end_rules : weld_intermittent_rule ;
    cutout_rules : weld_intermittent_rule ;
    penetration_rules : weld_intermittent_rule ;
    fillet_weld_length : length_measure_with_unit;
    fillet_weld_spacing : length_measure_with_unit;
    fillet_alignment : weld_alignment;
END_ENTITY;
```

(*)

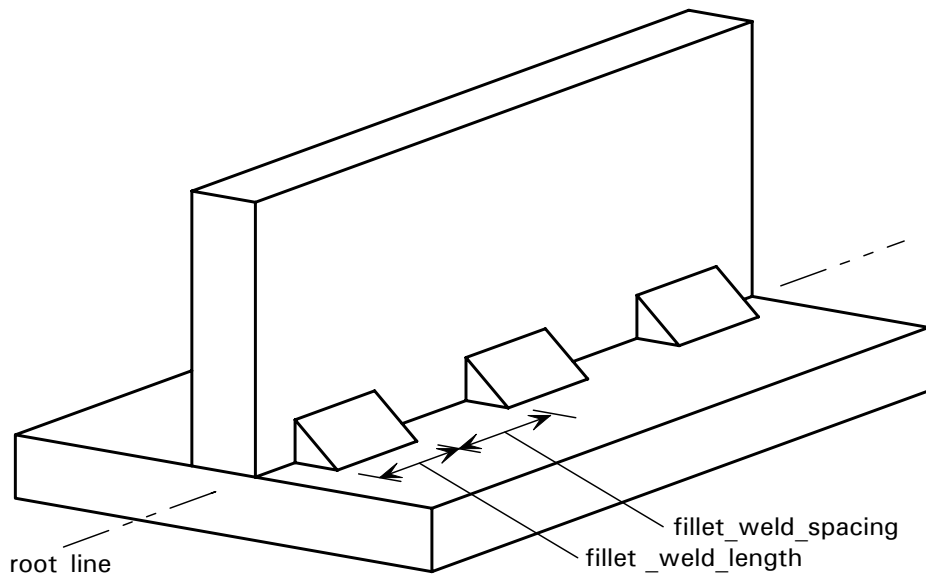


Figure 10.16: *Example of a weld_mechanism_fillet_intermittent*

Attribute definitions:

end_rules

An aspect of an intermittent fillet weld that defines the rule of length of weld that must be made on each end. (See the definition of the type weld_intermittent_rule.)

cutout_rules

An aspect of an intermittent fillet weld that defines the rule of length of weld that must be made on each side of a cutout. (See the definition of the type weld_intermittent_rule.)

penetration_rules

An aspect of an intermittent fillet weld that defines the rule of penetration of weld that must be made on the intermittent weld. (See the definition of the type weld_intermittent_rule.)

fillet_weld_length

Declares the instance of length_measure_with_unit used to define the length of continuous weld bead to be deposited. These lengths are spaced at intervals defined by fillet_weld_spacing and are adjusted at the ends of the weld and at the location of cutouts.

fillet_weld_spacing

Declares the instance of length_measure_with_unit used to define the length of the distance between intermittent fillet weld lengths. The value represents the greatest distance between weld lengths that is not welded. For example, if the weld length is 75 cm and the spacing is 150 cm then only 33% ($75/(75+150)$) of physical weld joint is actually welded.

fillet_alignment

An indicator as to whether or not each one of paired single fillet welds making up the double fillet weld are aligned with each other or offset from each other. (See the definition of the type weld_alignment.)

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.45 weld_mechanism_groove**Entity definition:**

A type of weld_mechanism assigned a series of parameters used to define a groove weld¹. A groove weld is a weld between the structural parts where, either due to increased part thickness or highly stressed joints, deeper penetration of weld materials is required. In order to achieve this penetration, the edges of the parts to be joined require edge preparation such as bevel/chamfer features or grinding features. This type of weld is usually stronger but more costly than a fillet weld. It is usually used in highly stressed and critical areas of the structure.

The groove weld is commonly used to make edge-to-edge butt joints, although it is also often used in corner joints, T joints, and joints between curved and flat pieces. There are many ways to make a groove weld, the differences depending primarily on the geometry of the parts to be joined and the preparation of their edges. Weld metal is deposited within the groove and penetrates and fuses with the base metal to form the joint.

A weld_mechanism_groove may be defined as butt or bevelled through its subtypes.

The local coordinate system for the joint_system_welded that uses this weld_mechanism has its origin at the mid-depth of the root_face and at mid width of the root_gap. The weld path is based on this origin. Where the root_face dimension is zero (as would typically be the case for edge joints) the origin of the local coordinate system lies at the base of the groove and at mid width of the root_gap (if any).

Examples of local coordinate systems for each type of joint and weld preparation are shown in Table 10.1 and Table 10.2.

EXPRESS specification:

*)

ENTITY weld_mechanism_groove

SUPERTYPE OF (ONEOF(weld_mechanism_groove_beveled,
weld_mechanism_groove_butt))

SUBTYPE OF (weld_mechanism);

sidedness : weld_sidedness;

backing_type : OPTIONAL weld_backing_type;

weld_joint_spacer : BOOLEAN;

surface_shape : weld_surface_shape;

joint_configuration : weld_configuration;

root_gap : OPTIONAL positive_length_measure_with_unit;

root_face : OPTIONAL positive_length_measure_with_unit;

WHERE

WRW9 : SELF\weld_mechanism.weld_mechanism_type = BUTT_WELD;

WRW10 : NOT (SELF\weld_mechanism.penetration = DEEP_PENETRATION);

END_ENTITY;

¹ The term 'butt weld' is commonly used in European practice in preference to 'groove weld'.

(*)

Attribute definitions:***sidedness***

An identifier that denotes whether the groove weld is to be made at one side or at both sides of the welded joint. (See the definition of the type weld_sidedness.)

backing_type

The type of required underside (back) of the weld in order to prevent “blow through” and/or eliminate the need for back gouging. Proper selection of backing reduces manufacturing costs by making it possible to joint parts by welding from a single side. (See the definition of the type weld_backing_type.)

weld_joint_spacer

Declares whether or not the joint requires the use of additional material to maintain the weld dimensions before or during the welding process. This is purely a Boolean identifier that indicates that a spacer is required. The geometric definition of the spacer must be derived from the joint detailed dimensions.

surface_shape

An identifier that denotes the shape of the surface of the groove weld as either flush, convex or concave. (See the definition of the type weld_surface_shape.)

joint_configuration

An identifier as to the configuration of the welded joint of the groove weld; e.g. lap, corner, tee, cruciform. (See definition of the type weld_configuration.)

root_gap

Declares the instance of positive_length_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the width of the gap between the welded faces at the weld root, measured parallel to the y-axis of the local coordinate system of the joint_system_welded. The root gap is defined in BS 499^[2] as the minimum distance at any cross section between edges, ends or surfaces to be joined.

root_face

Declares the instance of positive_length_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the length of the weld face at the root of the weld, measured parallel to the z-axis of the local coordinate system of the joint_system_welded. The root face is defined in BS 499^[2] as the portion of the fusion face that is not bevelled or grooved, while the fusion face is defined as the portion of a surface, or of an edge, that is to be fused in making a fusion weld.

Formal propositions**WRW9**

The attribute weld_mechanism_type, inherited from the supertype weld_mechanism, shall be populated with the value BUTT_WELD.

WRW10

The attribute penetration, inherited from the supertype weld_mechanism, shall NOT be populated with the value DEEP_PENETRATION.

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.46 weld_mechanism_groove_beveled

Entity definition:

A type of weld_mechanism_groove assigned a series of parameters used to define a beveled groove weld. A bevelled groove weld is one type of groove weld where the parts being welded are of significantly different thickness and the edge or edges are trimmed so as to provide a smooth transition across the joint.

The various types of beveled groove weld are:

- **V-groove**, in which the edges of both pieces are chamfered, either singly or doubly, to create the groove
- **Bevel groove**, in which the edge of one of the pieces is chamfered and the other is left square
- **U-groove**, in which the edges of both pieces are given a concave treatment
- **J-groove**, in which the edge of one of the pieces is given a concave treatment and the other is left square. (It is to the U-groove weld what the bevel groove weld is to the V-groove weld.)
- **Flare-V groove weld**, commonly used to join two round or curved parts
- **Flare bevel groove**, commonly used to join a round or curved piece to a flat piece.

Each of the above types may be single or double. The distinction is made using the attribute `endcut_shape`. (See also the definition of the type `weld_shape_bevel`.) Where groove welds are double sided – CIS/2 assumes that they are symmetrical, with dimensions on both sides being equal.

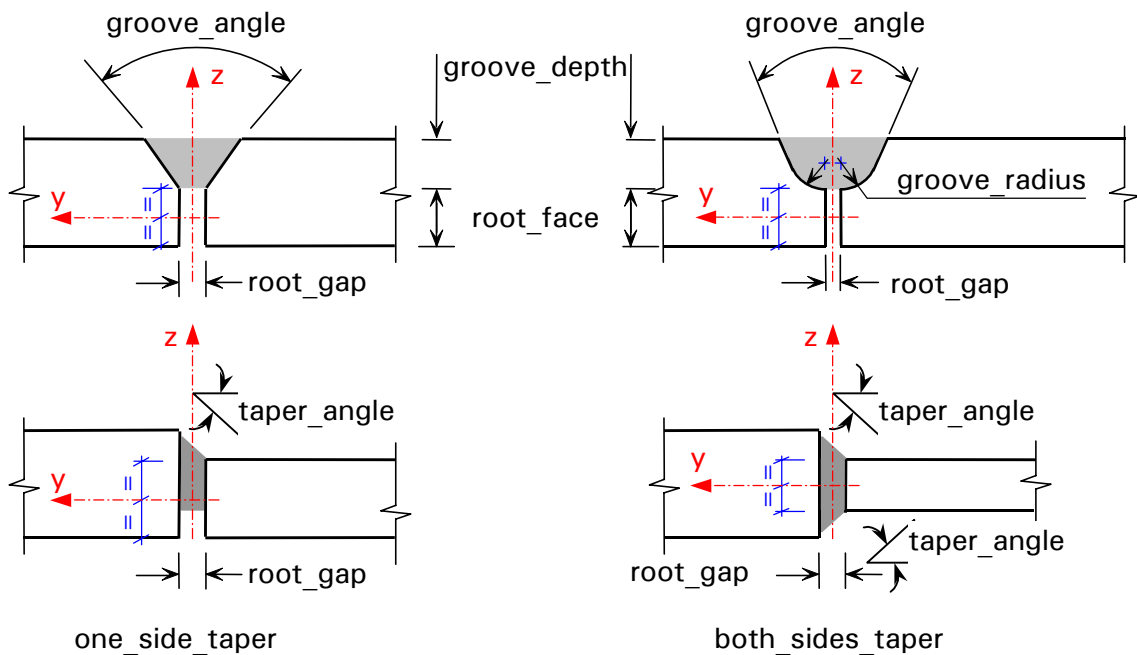


Figure 10.17: Defining the attributes of `weld_mechanism_groove_beveled`

EXPRESS specification:

```

*)
ENTITY weld_mechanism_groove_beveled
SUBTYPE OF (weld_mechanism_groove);
    endcut_shape : weld_shape_bevel;
    groove_angle : OPTIONAL plane_angle_measure_with_unit;
    groove_depth : OPTIONAL positive_length_measure_with_unit;
    groove_radius : OPTIONAL positive_length_measure_with_unit;
    taper : weld_taper_type;
    taper_angle : OPTIONAL plane_angle_measure_with_unit;
WHERE
    WRW12 : NOT ((taper = NON_TAPER) AND EXISTS(taper_angle));
END_ENTITY; END_ENTITY;
(*

```

Attribute definitions:***endcut_shape***

An identifier used to denote the particular configuration of bevel being represented. The configuration includes the specification of whether the bevel groove weld has a flat, flare or curved faces. (See definition of the type weld_shape_bevel.)

groove_angle

Declares the instance of plane_angle_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the angle between the planes of the fusion faces of the parts to be welded. This is also known as the '*included angle*' or the '*angle of preparation*'.

groove_depth

Declares the instance of positive_length_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the depth of the grooved or bevelled portion of the fusion face, measured paralld to the z-axis of the local coordinate sytem of the joint_system_welded.

groove_radius

Declares the instance of positive_length_measure_with_unit that may be used to specify the numerical value with an appropriate unit of the radius of the curved portion of the fusion face in a component prepared for a J or U weld. This is also known as the '*root radius*'

taper

An identifier used to denote the particular taper of the bevel groove weld being represented. It includes the specification of whether the bevel groove weld has a taper on one side, both sides or neither. If the weld is tapered, the welded surfaces are sloped relative to the parts being joined. (See definition of the type weld_taper_type.) European practice would consider this to be a combination of a butt weld and a fillet weld.

taper_angle

Declares the instance of plane_angle_measure_with_unit that may be associated with the weld_mechanism_groove_beveled to provide a measure of the taper angle (or slope of the welded surface).

Formal propositions:*WRW12*

The attribute `taper_angle` shall remain unpopulated if the attribute `taper` is assigned the value `NON_TAPER`.

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.47 weld_mechanism_groove_butt**Entity definition:**

A type of `weld_mechanism_groove` assigned a series of parameters used to define a butt groove weld. A butt groove weld is one type of groove weld where the parts being welded are of the same approximate thickness and no edge preparation (i.e. bevel or chamfer) is required. The square groove is created by either a tight fit or a slight separation of the edges.

```
(*
ENTITY weld_mechanism_groove_butt
SUBTYPE OF (weld_mechanism_groove);
    face_shape : weld_shape_butt;
END_ENTITY;
(*
```

Attribute definitions:**face_shape**

The face shape of parts of a butt groove weld. (See the definition of the type `weld_shape_butt`.)

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.48 weld_mechanism_prismatic**Entity definition:**

A type of `weld_mechanism` assigned a series of parameters used to define the explicit shape of the weld using series of cross sections.

```
(*
ENTITY weld_mechanism_prismatic
SUBTYPE OF (weld_mechanism);
    cross_sections : LIST [2:?] OF section_profile;
    points_defining_weld_path : LIST [2:?] OF UNIQUE point_on_curve;
    section_orientations : LIST [2:?] OF orientation_select;
    joint_configuration : weld_configuration;
DERIVE
    number_of_sections : INTEGER := SIZEOF(cross_sections);
    curve_defining_weld : curve :=
        points_defining_weld_path[1]\point_on_curve.basis_curve;
```

```

joints : SET [0:?] OF joint_system_welded := bag_to_set
  (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
    JOINT_SYSTEM_WELDED.WELD_SPECIFICATION'));
WHERE
  WRW3 : ( (SIZEOF (points_defining_weld_path) = number_of_sections) AND
    (SIZEOF (section_orientations) = number_of_sections) );
  WRW4 : SIZEOF(QUERY(temp <* points_defining_weld_path |
    (temp\point_on_curve.basis_curve) :<>: curve_defining_weld)) = 0;
  WRW5 : SIZEOF(QUERY(joint <* joints | (NOT('STRUCTURAL_FRAME_SCHEMA.
    JOINT_SYSTEM_WELDED_LINEAR' IN TYPE OF(joint))))=0;
  WRW6 : SIZEOF(QUERY(joint <* joints | (NOT(joint.weld_path :=
    curve_defining_weld)))) = 0;
END_ENTITY;
(*

```

Attribute definitions:

cross_sections

Declares the list of instances of *section_profile* associated with the *weld_mechanism_prismatic*. It is implied that the each of the section profiles referenced defines the shape of the complex weld. It is assumed that between each defining point there is a linear transition between each section profile. The LIST data type allows a reference to a *section_profile* to be repeated. If the cross section of the weld is constant throughout, then this attribute will reference the same instance of *section_profile* (at least) twice. The section profile is located such that its cardinal point sits on the defining curve of the weld path. The cardinal point may (but need not) be the geometric centroid of the section profile. The cardinal point may (be need not) be constant for each instance of *section_profile* referenced. In the simplest case, the same cardinal point will be used throughout the weld.

points_defining_weld_path

Declares the list of unique instances of *point_on_curve* associated with the *weld_mechanism_prismatic*. It is implied that the weld path is defined by the curve that is referenced by the *point_on_curve*.

It is assumed that all the instances of *point_on_curve* are all defined on the same curve, as referenced by the attribute *point_on_curve* of the entity *basis_curve*. That is, all the points for which the cross-section is defined all lie on the same curve. It is also assumed that this curve defines the weld path, upon which the cardinal points of the section profiles are placed.

section_orientations

Declares the list of two or more instances of *plane_angle_measure_with_unit* or *direction* associated with this instance of *weld_mechanism_prismatic*. The choice of which entity is used is made through the SELECT type *orientation_select*. The order of the instances in the list is significant. Each instance gives the numerical value of the orientation of the weld at the corresponding point on the weld path. As the cardinal point of the section profile lies on the weld path, the rotation of the section profile is defined as being about its cardinal point.

If this attribute references an instance of *plane_angle_measure_with_unit* (via the SELECT type) then the orientation of the weld is the angle measured (with an appropriate unit) from the weld's z-axis to the plane generated by the projection of the

weld's local x-axis in the global Z-axis direction. In the case where the weld's x-axis lies parallel to the global Z-axis, the orientation of the weld will be the angle measured from the weld's z-axis to the global X-axis.

If this attribute references an instance of direction (via the SELECT type) then the orientation of the weld is defined by the three real number values of the direction_ratios attribute (of the entity direction). These numbers define the 'orientation vector' for the weld. That is, the direction of the re-oriented local z-axis relative to the global coordinate system. For example, the normal (unrotated) orientation for a horizontal weld is given as (0.0, 0.0, 1.0), while the orientation vector for a vertical weld is given as (1.0, 0.0, 0.0).

It should be noted that if a direction with three values is used to define the orientation, one of the values assigned to the direction_ratios attribute should be zero, while the other two values should be non-zero. As the orientation is defined in one plane (perpendicular to the weld path), a direction assigned three non-zero values would be incorrect. The number of values assigned to the direction_ratios attribute is governed by the global rule compatible_dimension. This ensures that the count of direction_ratios matches the coordinate_space_dimension of the geometric_context in which the direction is geometrically_founded. This should be the same geometric_context used for the other geometric_representation_items in the model; e.g. the points defining the weld_path.

joint_configuration

An identifier as to the configuration of the welded joint of the groove weld; e.g. lap, corner, tee, cruciform. (See definition of the type weld_configuration.)

number_of_sections

This attribute derives the total number of cross sections that are used to define the instance of weld_mechanism_prismatic. The value is derived by calculating the size of the list of instances of section_profile in the populated attribute cross_sections.

curve_defining_weld

This attribute derives the curve upon which all the defining points of the weld_mechanism_prismatic lie. The curve is derived by examining the first instance in the list of instances of point_on_curve referenced by the attribute points_defining_weld_path.

joints

Derives the set of instances of joint_system_welded that reference this weld_mechanism_prismatic via its attribute weld_specification. This gives an indication of the use of the weld_mechanism_prismatic in forming representation of welded joints in the design or physical models.

Formal propositions:

WRW3

The number of cross sections, points and orientations used to define the weld shall all be equal. In other words, for every point defining the weld path there must be a corresponding section profile and angle of orientation.

WRW4

The size of the list of instances of point_on_curve having a different basis_curve from the first instance of point_on_curve shall equal zero. In other words, all the points used to define the weld path must lie on the same curve.

WRW5

The joint systems in which this weld is used shall all be linear.

WRW6

The size of the set of instances of `joint_system_welded` that use a curve to define a weld path (via the attribute `weld_path`) not equal to the curve specified here (via the derived attribute `curve_defining_weld`) shall equal zero. In other words, the weld path declared for the joint system shall be instance equal to curve defining the weld path here.

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

10.3.49 weld_mechanism_spot_seam**Entity definition:**

A type of `weld_mechanism` assigned a series of parameters used to define a spot weld or a seam weld. Spot welds and seam welds are types of welds where the weld is formed by locally heating and simultaneously pressurizing through the electrode, a comparatively small area formed by holding the lapped base metal with a properly formed electrode.

EXPRESS specification:

```
(*
ENTITY weld_mechanism_spot_seam
SUBTYPE OF (weld_mechanism);
    joint_configuration : weld_configuration;
WHERE
    WRW11 : (SELF.weld_mechanism.weld_mechanism_type = SPOT_WELD) OR
            (SELF.weld_mechanism.weld_mechanism_type = SEAM_WELD);
END_ENTITY;
*)
```

Attribute definitions:**joint_configuration**

An identifier as to the configuration of the welded joint of the spot or seam weld; e.g. lap, corner, tee, cruciform. (See definition of the type `weld_configuration`.)

Formal propositions:**WRW11**

The attribute `weld_mechanism_type`, inherited from the supertype `weld_mechanism`, shall be populated with the value `SPOT_WELD` or the value `SEAM_WELD`.

Notes

New for CIS/2 2nd Edition.

See Diagram 77 of the EXPRESS-G diagrams in Appendix B.

11 LPM/6 MATERIALS

11.1 Materials concepts and assumptions

Analytical elements and physical products (such as parts and fasteners) may be described with or without a material. Materials may be described as isotropic, orthotropic or anisotropic. LPM/6 supports a comprehensive range of material properties including density, strength, elasticity and hardness. These properties may be described as being dependent on strain, loading or temperature.

The materials referred to in this subject area are concerned with structural steelwork. LPM/6 is intended to be flexible enough to be extended to include other materials.

Each `structural_frame_product` and `finite_element` may be associated with a material through the use of ANDOR SUPERTYPE constructs. A material has a set of material properties, which are valid for a particular temperature range. A particular instance of material can have a material strength, which may be valid for a particular thickness range. It may also have a number of constituent materials. A set of elasticities may also be associated with the material. Each instance of elasticity contains values for Poisson's ratio, Young's modulus and the shear modulus. These values may be specified as being valid for a particular strain range. Thus, if the material is linear elastic, there will only be one instance of elasticity. If the material has tri-linear stress-strain properties, then there will be 3 instances of elasticity.

11.2 Materials type definitions

No types have been modified for the 2nd Edition in this subject area.

11.2.1 `loading_status`

Type definition:

Classifies the status of the load as either increasing, decreasing, constant or unloaded. This is used where the material properties are considered to be dependent upon the loading of the material. This would be the case for 'plastic' material properties.

EXPRESS specification:

*)

```
TYPE loading_status
= ENUMERATION OF
    (load_increasing,
    load_decreasing,
    load_constant,
    unloaded);
END_TYPE;
(*)
```

Notes:

New for CIS/2. (Used with `material_loading_context`.)

Unchanged in 2nd Edition.

11.3 Materials entity definitions

No entities have been added to this subject area for the 2nd Edition. The following entities have been modified for the 2nd Edition in this subject area:

- material

11.3.1 material

Entity definition:

A type of structural_frame_item that is used to specify the properties and/or the composition of a material. The described material may be structural (load-bearing) or non-structural. This entity is primarily aimed representing (structural) steel from a structural engineering point of view. It can of course, be used to describe other materials. The material can be described (through its SUBTYPEs) as being either anisotropic, isotropic, or orthotropic. It can also be described as being a constituent of another (parent) material.

As a SUBTYPE of structural_frame_item, a material inherits the three attributes item_number, item_name, and item_description. It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a material may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the material to be described in detail
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A material may also be related to another material (or any other structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

*)

ENTITY material

SUPERTYPE OF (ONEOF

(material_anisotropic,
material_isotropic,
material_orthotropic) ANDOR
material_constituent)

SUBTYPE OF (structural_frame_item);

DERIVE

material_number : INTEGER := SELF\structural_frame_item.item_number;
material_name : label := SELF\structural_frame_item.item_name;
material_grade : BAG OF identifier := SELF\structural_frame_item.item_ref;

UNIQUE

URM6 : material_number, material_name;

END_ENTITY;

(*

Attribute definitions:

material_number

A numerical reference for the material derived from the attribute item_number inherited from the SUPERTYPE structural_frame_item.

material_name

A short text reference for the material derived from the attribute item_name inherited from the SUPERTYPE structural_frame_item.

material_grade

A bag of short alphanumerical references for the grade of the material derived from the attribute item_ref from the SUPERTYPE structural_frame_item. If a grade is specified for a material it should be represented using the attribute ref in the entity item_reference. A relationship is then made using the entity item_assignment. If the material is not assigned an item reference, an empty bag will be returned here.

In most cases, a material will be assigned only one grade and that would be taken from a list of standard references from a flavour file. However, there may be times when users wish to add their own references for the material grade. These could be taken from an industry agreed library, or from a proprietary list of material grades.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPEs. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

URM6

The combination of the values of material_number and material_name shall be unique to this material. In other words, the combination of material_number and material_name form a unique reference for the material.

Informal propositions:

This entity is used on its own to specify a material without reference to its properties. If details of its properties are required, this entity would be instanced with one of its SUBTYPEs.

As a SUBTYPE of structural_frame_item, it may be given an item reference (a standard product identifier) by being associated with an instance of item_reference_assigned.

Notes:

Known as MATERIAL in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE and UNIQUE clauses added.

11.3.2 material_anisotropic

Entity definition:

A type of material that is categorized as anisotropic; that is, a material that possesses different properties in different directions. Its material properties are described by a list of two or more material_representation. Each of these material representations are given an orientation through a list of placements. That is an isotropic material requires a set of coordinate systems to define its properties in the context of the material's orientation.

EXPRESS specification:

*)

ENTITY material_anisotropic

SUBTYPE OF (material);

properties : LIST [2:?] OF UNIQUE material_representation;

material_location : LIST [2:?] OF UNIQUE placement;

WHERE

WRM41 : SIZEOF(properties) = SIZEOF(material_location);

END_ENTITY;

(*

Attribute definitions:

properties

Declares the list of two or more instances of material_representation associated with the material_anisotropic, which describe the properties of the material in the orientations given by the associated placements. There must be at least two instances of material_representation referenced by each instance of material_anisotropic. The LIST data type implies that the order of the instances are important; i.e. the sequence corresponds to the sequence of placements.

material_location

Declares the list of two or more instances of placement associated with the material_isotropic, which describe the orientation of the material_properties. There must be at least two instances of placement referenced by each instance of material_anisotropic. The LIST data type implies that the order of the instances are important. Here the instances of placement are placed in the same order as the instances of material_representation.

Formal propositions:

WRM41

The size of the aggregation associated with the attribute properties shall be equal to the size of the aggregation associated with the attribute material_location. In other words, there must be as many instances of material_representation as there are instances of placement associated with this instance of material_anisotropic.

Informal propositions:

The placements and material_representations are instanced in corresponding sequences.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.3 material_constituent

Entity definition:

A type of material that is used to describe the composition of a material; that is each instance of material_constituent represents one of the constituents of the parent composite material.

EXPRESS specification:

```
*)
ENTITY material_constituent
SUBTYPE OF (material);
    parent_material : material;
    constituent_amount : ratio_measure_with_unit;
    composition_basis : OPTIONAL label;
    class : label;
    determination_method : OPTIONAL text;
WHERE
    WRM19 : parent_material :<>: (SELF);
END_ENTITY;
(*
```

Attribute definitions:

parent_material

Declares the instance of material associated with the material_constituent, for which this material is a constituent. There must be one (and only one) instance of material associated with each instance of material_constituent. The WHERE rule prevents instance recursion; that is, a material cannot be a constituent of itself. This does not prevent entity type recursion such that a material_constituent can act as a parent to another separate and distinct instance of material_constituent.

constituent_amount

Declares the instance of ratio_measure_with_unit associated with the material_constituent, which provides the numerical value with an appropriate unit of the amount of constituent material expressed as a ratio of the total amount of the parent material. The amount may be expressed in terms of mass, volume, or some other basis depending on the populated value of the associated attribute composition_basis. There must be one (and only one) instance of ratio_measure_with_unit referenced by each instance of material_constituent.

composition_basis

An optional text description of the basis upon which the composition is specified; e.g. “by volume”, “by mass”.

class

A short text description of the class of the material_constituent; e.g. “primary component”.

determination_method

An optional text description of the method used to determine the amount of the constituent material.

Formal propositions:**WRM19**

A material cannot be a constituent of itself. This WHERE rule prevents instance recursion, it does not prevent entity type recursion such that a material_constituent can act as a parent to another separate and distinct instance of material_constituent.

Informal propositions:

Because of the ANDOR declared in the SUPERTYPE, this entity may be instanced with one of material_anisotropic, material_isotropic, or material_orthotropic. That is, a constituent material can be described as being anisotropic, isotropic, or orthotropic, and instanced with the corresponding material properties.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.4 material_elasticity**Entity definition:**

A type of material_representation_item that provides one of the physical characteristics for the material for use in analysis and design. This entity specifies the elasticity of the material, which is a function of the stress - strain characteristics of the material concerned. A strain range may be specified through an associated instance of material_strain_context. Where no strain range is specified, the material is assumed to be elastic (with a constant E value). For bi-linear (or tri-linear) stress-strain curves, 2 (or 3) instances of material_elasticity need to be populated and associated with 2 (or 3) instances of material_strain_context.

EXPRESS specification:

*)

ENTITY material_elasticity

SUBTYPE OF (material_representation_item);

poisson_ratio : OPTIONAL REAL;

young_modulus : OPTIONAL pressure_measure;

shear_modulus : OPTIONAL pressure_measure;

secant_modulus : OPTIONAL pressure_measure;

WHERE

WRM20 : EXISTS (poisson_ratio) OR EXISTS (young_modulus)
OR EXISTS (shear_modulus) OR EXISTS (secant_modulus);

END_ENTITY;

(*

Attribute definitions:**poisson_ratio**

Provides the numerical value of the Poisson's ratio of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice.

young_modulus

Provides the numerical value of the Young's modulus of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice.

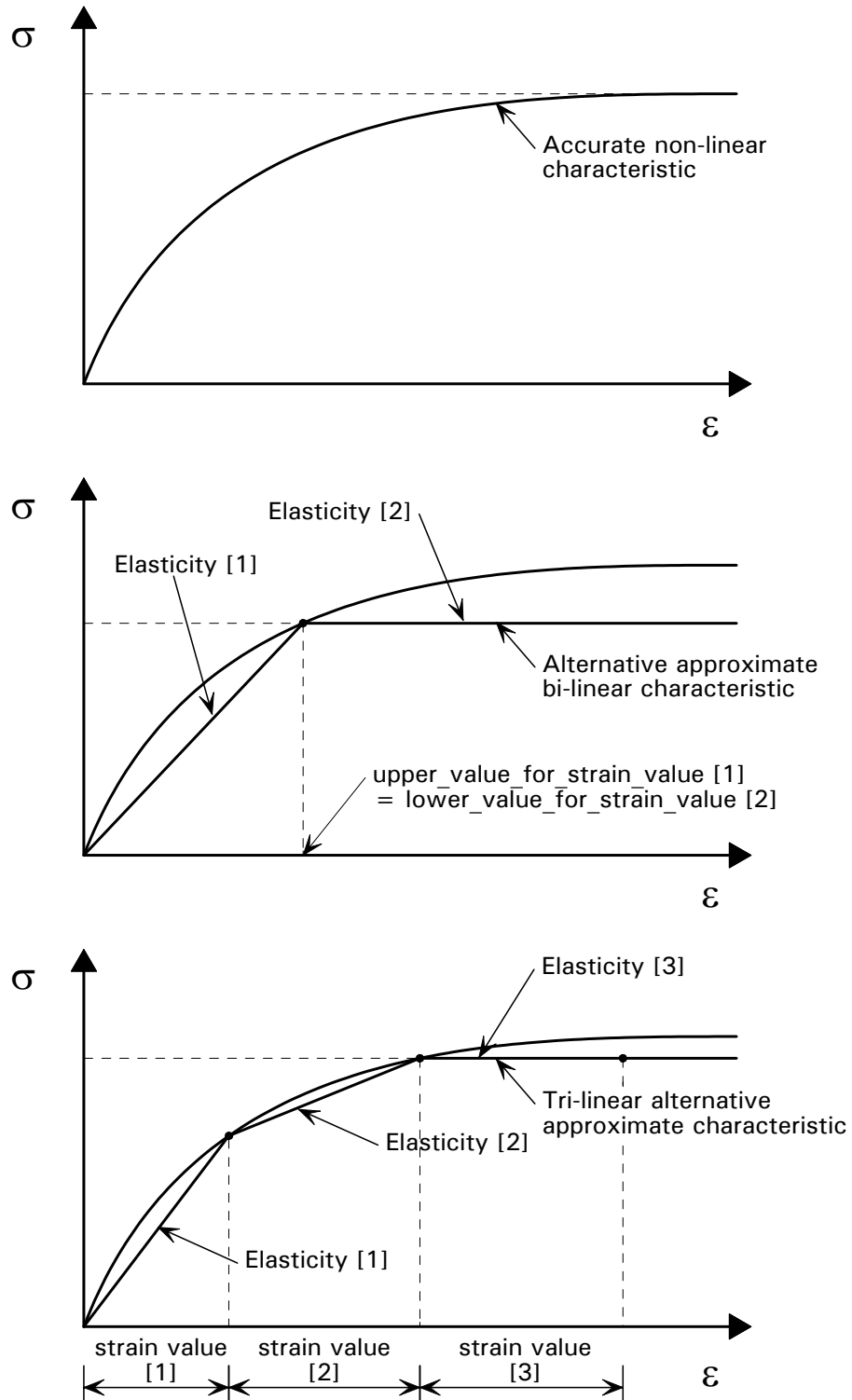


Figure 11.1 *Examples of bi-linear and tri-linear stress-strain characteristics*

shear_modulus

Provides the numerical value of the elastic torsional modulus of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice. This is a function of Young's Modulus and Poisson's ratio.

secant_modulus

Provides the numerical value of the secant modulus of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice. This parameter is used for materials such as stainless steel that do not have a linear stress-strain relationship or a clear yield point.

Formal propositions:*WRM20*

At least one of the attributes *poisson_ratio*, *young_modulus*, *shear_modulus*, or *secant_modulus* shall be populated.

Informal propositions:

Because the *material_representation_item* requires a *representation_context*, it may be associated with an instance of *global_unit_assigned_context* to provide the unit that should be associated with the parameter values. This is due to the WHERE rules in the entities *material_representation_item* and *material_representation*.

The unit of elasticity is defined as force per unit area and is therefore equivalent to the unit for pressure defined in the *LPM/6 Units & Measures* subject area. Thus, the parameters for elasticity have the data type *pressure_measure*, rather than an 'elasticity_measure'.

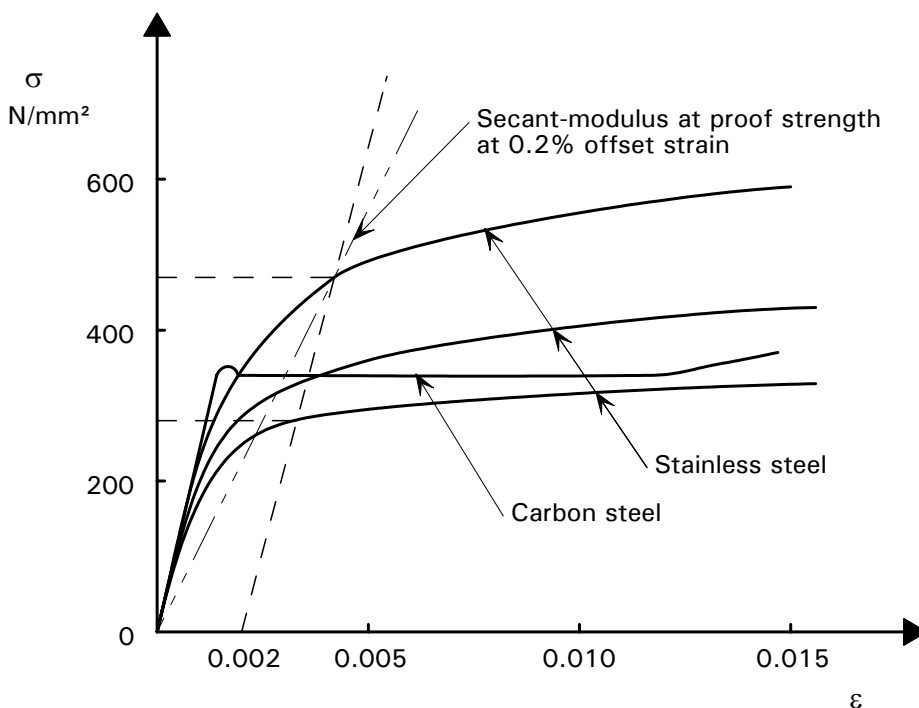


Figure 11.2 *Example of the secant modulus*

Notes:

New for CIS/2; partially addressed by ELASTICITY in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.5 material_hardness

Entity definition:

A type of material_representation_item that specifies the hardness of the material associated through the material_representation measured in accordance with the appropriate standard or code of practice. Because the ‘hardness’ of a material is subjective and context dependent, the type of the hardness value provided here is declared using the name attribute it inherits from its SUPERTYPE representation_item.

EXPRESS specification:

```
*)
ENTITY material_hardness
SUBTYPE OF (material_representation_item);
    hardness_values : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

hardness_values

Provides the numerical value of the hardness of the material measured in accordance with (or specified by) the appropriate standard or code of practice. The ‘type’ of hardness provided is declared using the attribute name (inherited from the SUPERTYPE representation_item). This could be, for example, a ‘surface hardness’ derived from a standard cone penetration test. The value of the hardness and its associated unit are both dependent on the context in which it is measured.

Informal propositions:

Because the material_representation_item requires a representation_context, it may be associated with an instance of global_unit_assigned_context to provide the unit that should be associated with the context_dependent_measure. The unit used is dependent upon the context in which the hardness is measured. This is due to the WHERE rules in the entities material_representation_item and material_representation.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.6 material_isotropic

Entity definition:

A type of material that has the same properties in all directions, which are defined using a single instance of material_representation. For most purposes, structural steel used in building construction will be considered to be isotropic.

EXPRESS specification:

```
*)
ENTITY material_isotropic
SUBTYPE OF (material);
```

```

    definition : material_representation;
END_ENTITY;
(*)

```

Attribute definitions:

definition

Declares the instance of material_representation associated with this material_isotropic, which specifies the material properties (through the associated set of material_representation_items) of the isotropic material. There must be one (and only one) instance of material_representation referenced by each instance of material_isotropic.

Notes:

New for CIS/2; addressed by MATERIAL in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.7 material_mass_density

Entity definition:

A type of material_representation_item that defines the mass density of the material, measured in accordance with the appropriate standard or code of practice.

EXPRESS specification:

```

*)
ENTITY material_mass_density
SUBTYPE OF (material_representation_item);
    mass_density : context_dependent_measure;
END_ENTITY;
(*)

```

Attribute definitions:

mass_density

Provides the numerical value of the mass density of the associated material. The mass density may be defined in a number of ways depending upon the context; e.g. mass per unit volume, or mass per unit area to define a surface density.

Informal propositions:

Because the material_representation_item requires a representation_context, it may be associated with an instance of global_unit_assigned_context to provide the unit that should be associated with the context_dependent_measure. This is due to the WHERE rules in the entities material_representation_item and material_representation. The unit used is dependent upon the context in which the mass density is measured.

Notes:

New for CIS/2; partially addressed in MATERIAL_PROPERTIES in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.8 material_orthotropic

Entity definition:

A type of material whose properties differ depending upon the plane under consideration. That is, the properties are different in two orthogonal directions. Thus, its material properties are defined using two instances of material_representation, one for in plane properties, one for out of plane properties. This entity is used to describe the material properties of planar components such as steel sheets and plates. While for most purposes, structural steel used in building construction will be considered to be isotropic, some specialist structures such as bridges and offshore structures will need to consider steel components as being composed of an orthotropic material. As a result of rolling, for example, a steel plate may have a through-thickness strength different from its longitudinal strength.

The in-plane properties are assumed to constant; i.e. the transverse material properties of a plate are the same as its longitudinal properties.

Where transverse, longitudinal and through-thickness properties of a steel plate are all different, the entity material_anisotropic should be used in preference to this entity.

EXPRESS specification:

```

*)
ENTITY material_orthotropic
SUBTYPE OF (material);
    in_plane_properties : material_representation;
    out_of_plane_properties : material_representation;
WHERE
    WRM40 : in_plane_properties :<>: out_of_plane_properties;
END_ENTITY;
(*

```

Attribute definitions:

in_plane_properties

Declares the first instance of material_representation associated with this material_orthotropic, which specifies the material properties (through the associated set of material_representation_items) of the orthotropic material parallel to the plane of the (sheet or plate) material.

out_of_plane_properties

Declares the second instance of material_representation associated with this material_orthotropic, which specifies the material properties (through the associated set of material_representation_items) of the orthotropic material perpendicular to the plane of the (sheet or plate) material.

Formal propositions:

WRM40

The instance of material_representation referenced by the attribute in_plane_properties shall not be the same instance of material_representation referenced by the attribute out_of_plane_properties. In other words, there must be two separate and distinct instances of material_representation associated with each instance of material_orthotropic.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.9 material_property_context**Entity definition:**

A type of representation_context and a complex abstract SUPERTYPE entity that provides (through its SUBTYPES) the context for the properties of the material defined through the associated set of SUBTYPES of material_representation and material_representation_item.

EXPRESS specification:

*)

ENTITY material_property_context

ABSTRACT SUPERTYPE OF (

material_property_context_dimensional ANDOR

material_property_context_loading ANDOR

material_property_context_strain ANDOR

material_property_context_stress ANDOR

material_property_context_temperature)

SUBTYPE OF (representation_context);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE representation_context. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity representation_context and collect together the SUBTYPES under a common category.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.10 material_property_context_dimensional

Entity definition:

A type of material property context that declares upper and lower limits for the values of the dimensions for which the associated material properties are valid. The material properties are contained is the SUBTYPEs of material_representation_item. These properties must be placed in a context. This entity provides one of these contexts. For example, certain material properties, such as the material strength, may only be valid for a particular range of thicknesses of steel.

EXPRESS specification:

```
*)
ENTITY material_property_context_dimensional
  SUBTYPE OF (material_property_context);
    lower_value_for_dimension : length_measure;
    upper_value_for_dimension : length_measure;
END_ENTITY;
(*
```

Attribute definitions:

lower_value_for_dimension

Provides the numerical value of the lower bound of dimension for which the material properties (as defined by the associated instances of the SUBTYPEs of material_representation_item) are valid.

upper_value_for_dimension

Provides the numerical value of the upper bound of dimension for which the material properties (as defined by the associated instances of the SUBTYPEs of material_representation_item) are valid.

Informal propositions:

Because the material_context is a SUBTYPE of representation_context, it may be instanced with the global_unit_assigned_context to provide the unit that should be associated with the length_measures. (See also, the definitions of material_representation and material_representation_item.)

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.11 material_property_context_loading

Entity definition:

A type of material_property_context that declares the status of loading for which the associated material properties are valid. The material properties are contained is the SUBTYPEs of material_representation_item. These properties must be placed in a context. This entity provides one of these contexts, and is used where the material properties are considered to be dependent upon the loading of the material. This would be the case for 'plastic' material properties. For example, certain material properties, such as the material elasticity, may only be valid when the material is loaded.

EXPRESS specification:

```

*)
ENTITY material_property_context_loading
SUBTYPE OF (material_property_context);
    loading : loading_status;
END_ENTITY;
(*)

```

Attribute definitions:*loading*

Declares the classification of the status of the loading on the material when the associated material properties are specified, calculated, or measured. The material's loading status may be classified as either increasing, decreasing, constant or unloaded. If the loading status is classed as increasing, the associated material properties are being specified as the load applied is increasing in magnitude. Similarly, if the loading status is classed as unloaded, the material is not loaded or the load has been removed when the associated material properties were calculated.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.12 material_property_context_strain**Entity definition:**

A type of material_property_context that declares upper and lower limits for the values of strain for which the associated material properties are valid. The material properties are contained is the SUBTYPEs of material_representation_item, and must be placed in a context. This entity provides one of these contexts. For example, certain material properties, such as the material elasticity, may only be valid for a particular range of strain. For bi-linear (or tri-linear) stress-strain curves, 2 (or 3) instances of material_elasticity need to be populated and associated with 2 (or 3) instances of material_strain_context.

EXPRESS specification:

```

*)
ENTITY material_property_context_strain
SUBTYPE OF (material_property_context);
    lower_value_for_strain : ratio_measure;
    upper_value_for_strain : ratio_measure;
END_ENTITY;
(*)

```

Attribute definitions:*lower_value_for_strain*

Provides the numerical value of the lower bound of strain range for which the material properties (as defined by the associated instances of the SUBTYPEs of material_representation_item) are valid.

upper_value_for_strain

Provides the numerical value of the upper bound of strain range for which the material properties (as defined by the associated instances of the SUBTYPES of material_representation_item) are valid.

Informal propositions:

Because the representation_context is an ANDOR SUPERTYPE, a material_strain_context may be instanced with a global_unit_assigned_context to provide the unit that should be associated with the ratio_measures.

The unit of strain is defined as ratio and therefore, the parameters for strain have the data type ratio_measure, rather than 'strain_measure'.

Notes:

New for CIS/2; partially addressed by ELASTICITY in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.13 material_property_context_stress**Entity definition:**

A type of material_property_context that declares upper and lower limits for the values of stress for which the associated material properties are valid. The material properties are contained is the SUBTYPES of material_representation_item, and must be placed in a context. This entity provides one of these contexts. For example, certain material properties, such as the material elasticity, may only be valid for a particular range of stress. For bi-linear (or tri-linear) stress-strain curves, 2 (or 3) instances of material_elasticity need to be populated and associated with 2 (or 3) instances of material_strain_context.

EXPRESS specification:

```
*)
ENTITY material_property_context_stress
  SUBTYPE OF (material_property_context);
    lower_value_for_stress : pressure_measure;
    upper_value_for_stress : pressure_measure;
END_ENTITY;
(*)
```

Attribute definitions:*lower_value_for_stress*

Provides the numerical value of the lower bound of stress range for which the material properties (as defined by the associated instances of the SUBTYPES of material_representation_item) are valid.

upper_value_for_stress

Provides the numerical value of the upper bound of stress range for which the material properties (as defined by the associated instances of the SUBTYPES of material_representation_item) are valid.

Informal propositions:

Because the `representation_context` is an ANDOR SUPERTYPE, a `material_strain_context` may be instanced with a `global_unit_assigned_context` to provide the unit that should be associated with the `ratio_measures`.

The unit of stress is defined as force per unit area and is therefore equivalent to the unit for pressure defined in the *LPM/6 Units & Measures* subject area. Thus, the parameters for stress have the data type `pressure_measure`, rather than ‘`stress_measure`’.

Notes

New for CIS/2.

See Diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.14 material_property_context_temperature**Entity definition:**

A type of `material_property_context` that declares upper and lower limits for the values of temperature for which the associated material properties are valid. The material properties are contained is the SUBTYPEs of `material_representation_item`, and must be placed in a context. This entity provides one of these contexts. For example, certain material properties, such as the mass density, may only be valid for a particular range of temperature.

EXPRESS specification:

*)

ENTITY `material_property_context_temperature`

SUBTYPE OF (`material_property_context`);

`temperature_lower_bound` : `thermodynamic_temperature_measure`;

`temperature_upper_bound` : `thermodynamic_temperature_measure`;

END_ENTITY;

(*

Attribute definitions:*temperature_lower_bound*

Provides the numerical value of the lower bound of temperature range for which the material properties (as defined by the associated instances of the SUBTYPEs of `material_representation_item`) are valid.

temperature_upper_bound

Provides the numerical value of the upper bound of temperature range for which the material properties (as defined by the associated instances of the SUBTYPEs of `material_representation_item`) are valid.

Informal propositions:

Because the `representation_context` is an ANDOR SUPERTYPE, a `material_temperature_context` may be instanced with a `global_unit_assigned_context` to provide the unit that should be associated with the `thermodynamic_temperature_measure`.

Notes:

New for CIS/2; partially addressed by MATERIAL_PROPERTIES in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.15 material_representation

Entity definition:

A type of representation where the associated set of representation_items are all of the type 'material_representation_item'. This entity brings together a set of one or more material properties and places them in a context.

EXPRESS specification:

*)

ENTITY material_representation

SUBTYPE OF (representation);

WHERE

```
WRM21 : ('STRUCTURAL_FRAME_SCHEMA.
MATERIAL_PROPERTY_CONTEXT' IN
TYPE OF (SELFrepresentation.context_of_items)) AND
('STRUCTURAL_FRAME_SCHEMA.
GLOBAL_UNIT_ASSIGNED_CONTEXT' IN TYPEOF
(SELFrepresentation.context_of_items));
WRM22 : SIZEOF(QUERY(temp <* SELFrepresentation.items |
('STRUCTURAL_FRAME_SCHEMA.
MATERIAL_REPRESENTATION_ITEM') IN TYPE OF(temp)))) > 0;
```

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE representation. The purpose of this entity is to constrain (or specialize) the STEP Part 43 entity representation when it is used in LPM/6 to represent materials and their properties.

Formal propositions:

WRM21

The context provided for the set of instances of representation_item associated with the material_representation shall be of the type material_property_context **and** global_unit_assigned_context. In other words, material properties must be placed in a context that provides the units and (upper and lower bounds for) the range of applicability of those material properties. The resulting complex entity instance is encoded in a STEP Part 21 file using external mapping.

WRM22

The size of the set of instances of representation_item referenced by the attribute items (inherited from the SUPERTYPE representation) that are of the type material_representation_item shall be greater than zero. In other words, at least one item associated with the material_representation shall all be of the type 'material_representation_item'.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.16 material_representation_item

Entity definition:

A type of representation_item that represents one of the properties of the associated material. The material_representation_item can be categorized (through its SUBTYPEs) as either elasticity, mass density, thermal expansion, strength, hardness, or toughness. This entity would normally be instanced as one of its SUBTYPEs.

Because many of the properties of a material are subjective and context dependent, the type of the value provided is declared using the attribute name inherited from the SUPERTYPE representation_item.

EXPRESS specification:

*)

ENTITY material_representation_item

SUPERTYPE OF (ONEOF

(material_elasticity,
material_mass_density,
material_thermal_expansion,
material_strength,
material_hardness,
material_toughness))

SUBTYPE OF (representation_item);

WHERE

WRM23 : SIZEOF (QUERY (tmp_rep <* using_representations (SELF) | NOT
('STRUCTURAL_FRAME_SCHEMA.MATERIAL_PROPERTY_CONTEXT' IN
TYPE OF (tmp_rep.context_of_items)))) = 0;

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE representation_item. The purpose of this entity is to constrain (or specialize) the STEP Part 43 representation_item when it is used in LPM/6 to represent material properties.

Formal propositions:

WRM23

The size of the set of instances of representation_context referenced by the attribute context_of_items of the entity representation (derived by the function using_representations) that are not of the type material_representation_context shall equal zero. In other words, the context of any representation referencing a material_representation_item shall be of the type material_representation_context.

Informal propositions:

This entity would normally be instanced as one of its SUBTYPEs.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.17 material_strength

Entity definition:

A type of material_representation_item that provides one value of the strength of the associated material. This could be, for example, a yield strength, an ultimate tensile strength, a 2% proof strength, or a notional design strength. Because the ‘strength’ of a material is subjective and context dependent, the type of the strength value provided here is declared using the name attribute it inherits from the SUPERTYPE representation_item.

EXPRESS specification:

```
*)
ENTITY material_strength
  SUBTYPE OF (material_representation_item);
    material_strength_value : pressure_measure;
END_ENTITY;
(*
```

Attribute definitions:

material_strength_value

Provides the numerical value of the strength of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice. The ‘type of strength’ provided is declared using the attribute name inherited from the SUPERTYPE representation_item. This could be, for example, a yield strength, an ultimate tensile strength, a 2% proof strength, or a notional design strength. The value of the strength and its associated unit are both dependent on the context in which the strength is measured.

Informal propositions:

Because the material_representation_item requires a representation_context, the material_strength may be associated with an instance of global_unit_assigned_context to provide the unit that should be associated with the context_dependent_measure.

The unit of strength is defined as force per unit area and is therefore equivalent to the unit for pressure defined in the *LPM/6 Units & Measures* subject area. Thus, the parameter for strength has the data type pressure_measure, rather than ‘strength_measure’.

Notes:

New for CIS/2; partially addressed by MATERIAL_STRENGTH in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.18 material_thermal_expansion

Entity definition:

A type of material_representation_item that provides the value of the coefficient of thermal expansion for the associated material. Because the thermal expansion of a material is subjective and context dependent, the type of the thermal expansion coefficient value provided here is declared using the name attribute it inherits from its SUPERTYPE

representation_item. This could be, for example, a coefficient of linear thermal expansion, or a coefficient of volumetric thermal expansion.

EXPRESS specification:

```
*)
ENTITY material_thermal_expansion
  SUBTYPE OF (material_representation_item);
    thermal_expansion_coeff : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

thermal_expansion_coeff

Provides the value of the coefficient of thermal expansion of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice. The ‘type’ of coefficient provided is declared using the attribute name inherited from the SUPERTYPE representation_item. This could be, for example, a ‘linear’ or ‘volumetric’ coefficient of thermal expansion. The value of the coefficient and its associated unit are both dependent on the context in which the thermal expansion is measured.

Informal propositions:

Because the material_representation_item requires a representation_context, the material_thermal_expansion may be associated with an instance of global_unit_assigned_context to provide the unit that should be associated with the context_dependent_measure.

Notes:

New for CIS/2; partially addressed by MATERIAL_PROPERTIES in CIS/1.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

11.3.19 material_toughness

Entity definition:

A type of material_representation_item that provides the value of the toughness for the associated material. Because the ‘toughness’ of a material is subjective and context dependent, the type of the toughness value provided here is declared using the attribute name it inherits from its SUPERTYPE representation_item. This could be, for example, a fracture toughness derived from a Charpy notch test. The toughness of a material is related to its ductility and brittleness.

EXPRESS specification:

```
*)
ENTITY material_toughness
  SUBTYPE OF (material_representation_item);
    toughness_values : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

toughness_values

Provides the numerical value of the toughness of the associated material, measured in accordance with (or specified by) the appropriate standard or code of practice. The 'type' of toughness provided is declared using the attribute name inherited from the SUPERTYPE representation_item. This could be, for example, a 'fracture toughness'. The value of the toughness and its associated unit are both dependent on the context in which the toughness is measured.

Informal propositions:

Because the material_representation_item requires a representation_context, the material_toughness may be associated with an instance of global_unit_assigned_context to provide the unit that should be associated with the context_dependent_measure.

Notes:

New for CIS/2.

See diagram 69 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

12 LPM/6 LOCATION

12.1 Location concepts and assumptions

12.1.1 Coordinate Systems & Transformations

In LPM/6, a coordinate system is represented by the entity `coord_system`. The different types of coordinate systems - Cartesian, cylindrical and spherical are represented by SUBTYPES of the entity `coord_system`. (See diagram 50 of the EXPRESS-G diagrams in Appendix B.) An ANDOR SUPERTYPE construct is used to represent 'nested' coordinate systems.

In general, most systems will use either 2D or 3D Cartesian coordinate systems; the cylindrical and spherical coordinate systems being reserved for the more advanced or specialist systems. In particular, prismatic parts (those 'long products' commonly used in steel construction) will almost always use 3D Cartesian coordinate systems. Furthermore, the features defined for the ends of these prismatic parts (represented by the entity `feature_volume_prismatic` and its SUBTYPES) are defined using Cartesian coordinate systems.

12.1.2 Hierarchical locations

LPM/6 defines a hierarchical system of locations such that an item is located with respect to the higher level item to which it belongs. This is illustrated in Figure 12.1.

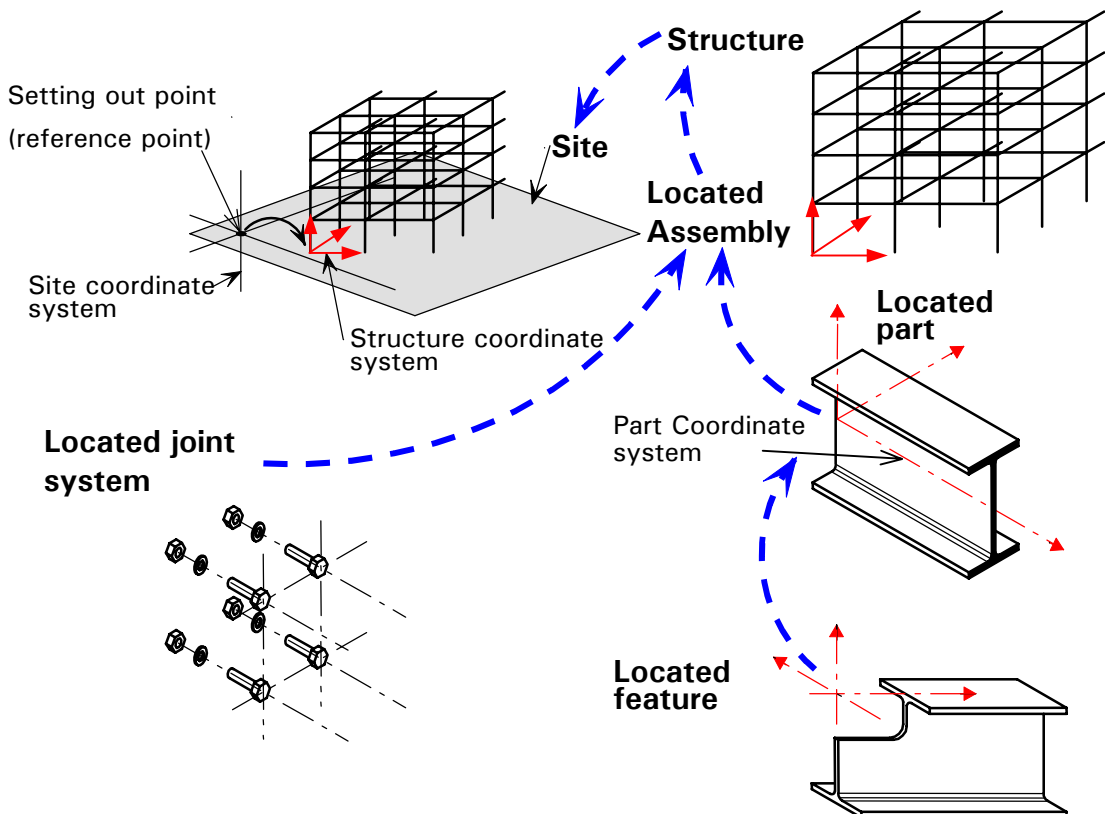


Figure 12.1 *The hierarchical system of location within LPM/6*

Thus, in the simplest case, a feature is located with respect to the part it modifies. Parts and joint systems are located with respect to the assembly to which they belong. An assembly is located with respect to the structure of which it forms one component. A

structure is located with respect to the site on which it is built. Finally, the site can be located with respect to the world (via the entity *geographical_location* described in the *LPM/6 Project Definition* subject area. The result of this hierarchical system of locations is a hierarchical system of coordinate systems; one located within another. Thus, the feature's coordinate system is located within part's coordinate system. Similarly, the part's coordinate system may be transformed and relocated within the coordinate system of its parent assembly.

12.1.3 The level discriminator

LPM/6 contains a concept known as the level discriminator. Using this mechanism, entities are defined as being at one of two levels: Specific, or Occurrence. Entities at the occurrence level, such as *located_part* are defined as located and oriented implementations of the corresponding specific level. Entities at the specific level, such as *part*, define the characteristics of a specific component, which is then used many times at the occurrence level.

The effect of using the level discriminator is illustrated in the definitions of the entities in the *LPM/6 Location* subject area. Many of the entities at the occurrence level are 'associative' in nature such that they only contain attributes whose data types are other entities. The SUBTYPES of the entity *located_item* are seen as associations between a specific item and a coordinate system.

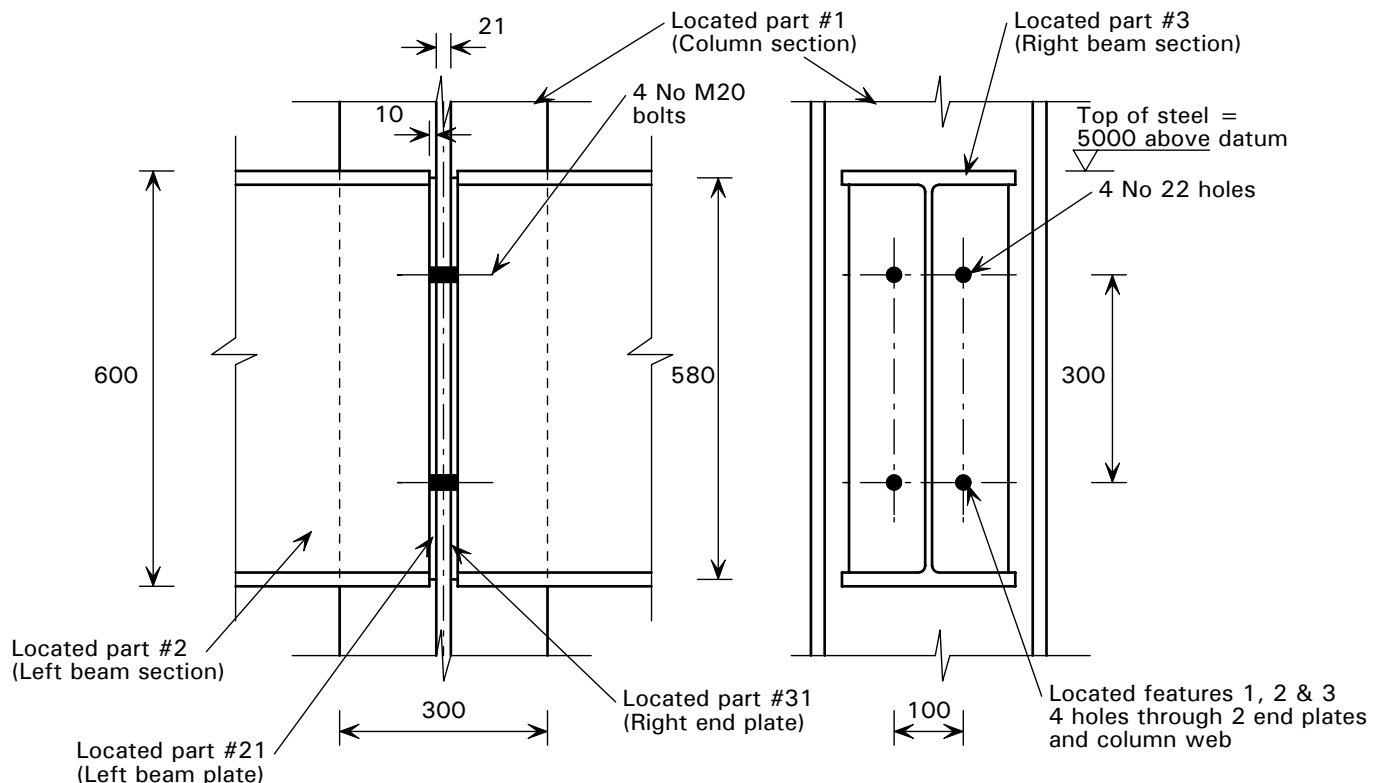


Figure 12.2 *Located parts and located joint systems*

Located parts are joined by located part joints. The *located_part_joint* is a logical entity describing the local connectivity of parts (i.e. which parts are joined together). The real connectivity (i.e. how the parts are joined) is described by the entity *located_joint_system*, which physically fulfils a *located_part_joint*.

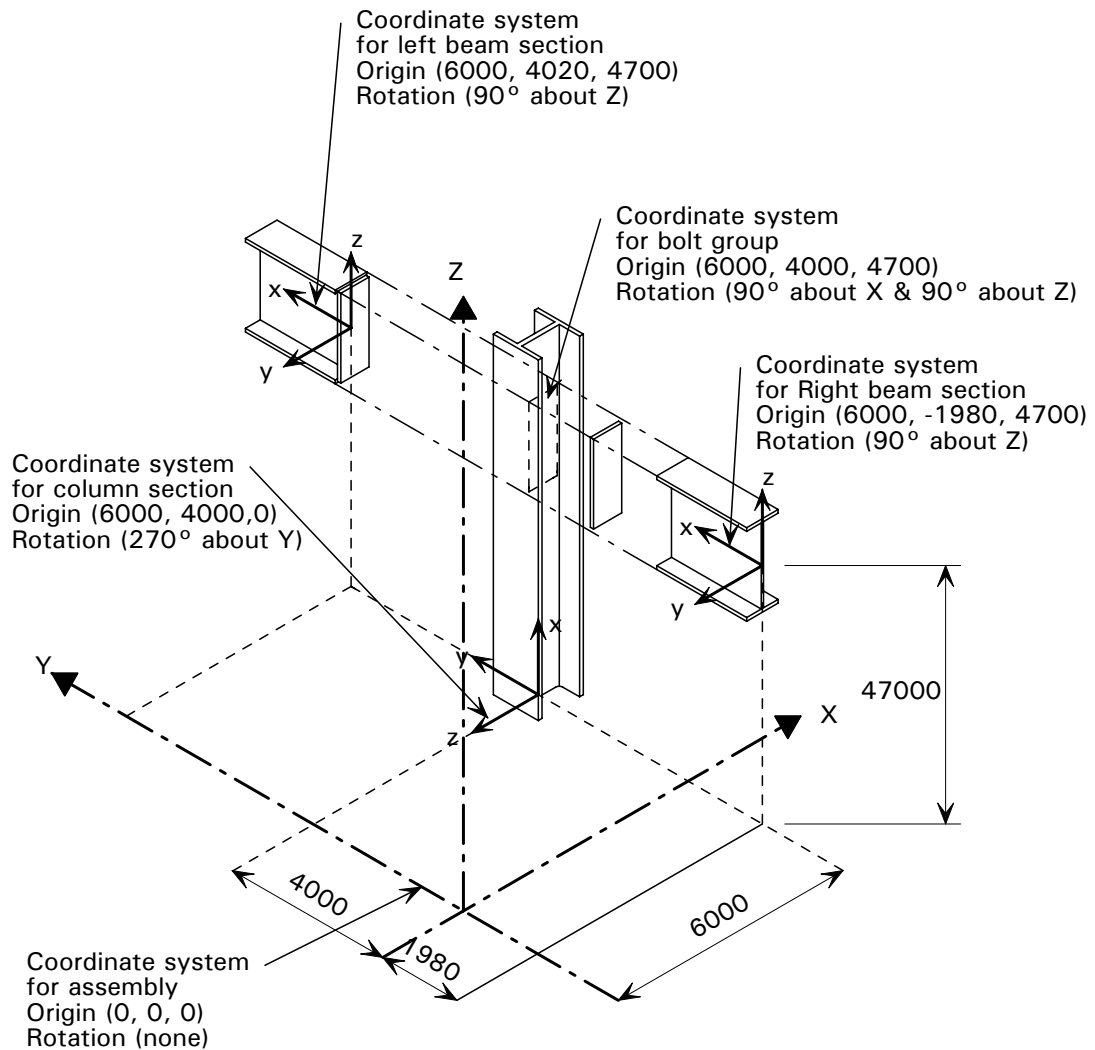


Figure 12.3 *Locating parts within an assembly*

A located joint system is defined as a located and oriented implementation of the corresponding (specific) joint system. A (specific) joint system is the combination of the means by which a structural connection is realized, together with the geometric description of that connection. It could be, for example, a bolted mechanism arranged in accordance with a given layout (or bolt pattern) or a welded mechanism arranged in accordance with a given weld path.

12.1.4 Units assigned to coordinate systems

Every coordinate system has an origin, and that origin is represented using one of the SUBTYPES of point. The coordinates of points in a coordinate system are measurable physical quantities and as such have units associated with them. Rather than assigning the units individually, LPM/6 uses the normal STEP method for globally assigning units to these measurements. Because the point is a SUBTYPE of `geometric_representation_item`, it requires a `geometric_representation_context`. Furthermore, the ANDOR SUPERTYPE `representation_context`, allows the `geometric_representation_context` to be a `global_unit_assigned_context`. This construct provides the unit that should be associated with the instances of `length_measure` and `plane_angle_measure` used to define the coordinates of the point.

12.2 Location type definitions

No types were modified or added in this subject area for the 2nd Edition.

12.2.1 project_select

Type definition:

Declares the instance referenced to be either a project or a zone_of_project. (See the definition of the entity located_site.)

EXPRESS specification:

```
*)
TYPE project_select
= SELECT
    (project, zone_of_project);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged for 2nd Edition.

12.2.2 site_select

Type definition:

Declares the instance referenced to be either a site, a located_site, a zone_of_site, a zone_of_building, or a building_complex. (See the definition of the entity located_structure.)

EXPRESS specification:

```
*)
TYPE site_select
= SELECT
    (site,
    located_site,
    zone_of_site,
    zone_of_building,
    building_complex);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged for 2nd Edition.

12.2.3 structure_select

Type definition:

Declares the instance referenced to be either a structure, a located_structure, a zone_of_structure, a zone_of_building or a located_assembly. (See the definition of the entity located_assembly.)

EXPRESS specification:

```

*)
TYPE structure_select
= SELECT
    (structure,
     located_structure,
     zone_of_structure,
     zone_of_building,
     located_assembly);
END_TYPE;
(*)

```

Notes

New for CIS/2. Unchanged for 2nd Edition.

12.3 Location entity definitions

The following entities have been added to this subject area for the 2nd Edition:

- located_assembly_marked
- located_part_marked

The following entities have been modified in this subject area for the 2nd Edition:

- located_assembly
- located_assembly_child

The following entities have been replaced with an equivalent entity of the same name from STEP Part 42 in this subject area for the 2nd Edition:

- cylindrical_point
- spherical_point

Since the definitions of these entities can be found in STEP Part 42, they have been removed from this subject area.

12.3.1 coord_system

Entity definition:

An abstract SUPERTYPE entity that provides the definition of a coordinate system. The coordinate system may be categorized (through its SUBTYPES) as being either a Cartesian, spherical or cylindrical. The coordinate system may also be categorized as a child (or local) coordinate system that is located with a parent (or global) coordinate system. This mechanism allows for the definition of nested coordinate systems. The coordinate system must be given a name, a dimensionality and a description of its use. A description of the sign convention may also be provided.

EXPRESS specification:

```

*)
ENTITY coord_system
ABSTRACT SUPERTYPE OF (ONEOF
    (coord_system_cartesian_2d,
     coord_system_cartesian_3d,

```

```

        coord_system_spherical,
        coord_system_cylindrical) ANDOR
        coord_system_child);
    coord_system_name : label;
    coord_system_use : label;
    sign_convention : OPTIONAL text;
    coord_system_dimensionality : dimension_count;
END_ENTITY;
(*)

```

Attribute definitions:***coord_system_name***

A short text reference used to identify the coordinate system.

coord_system_use

An text description of the use (or role) of the coordinate system. The population of this attribute is contrained when the entity is used to specify the location of a located_item. The value assigned to this attribute is checked by the where rules in the SUBTYPEs of located_item. For example, if the coordinate system is being used to provide the position of a part, its use would have to be described as a 'Part Coordinate System'.

sign_convention

An optional text description of the sign convention used in the coordinate system (e.g. 'Z-axis vertically upwards, right handed', etc.)

coord_system_dimensionality

Provides the integer value of the dimensionality of the coord_system; e.g. 2 for two-dimensional coordinate systems, or 3 for three-dimensional coordinate systems. The value of the attribute is checked by where rules in other entities.

Formal propositions:***ABSTRACT SUPERTYPE***

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPEs. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

Known as COORD_SYSTEM in CIS/1.

See diagram 50 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

12.3.2 coord_system_cartesian_2d**Entity definition:**

A type of coordinate system, which is defined using 2 orthogonal axes, labelled x and y, x and z, or y and z. This entity references the STEP Part 42 geometric entity

axis2_placement_2d to complete its definition, which defines the orientation of two orthogonal axes placed in 2 dimensional space. (See also definition of the entity coord_system_cartesian_3d.)

STEP Part 42 defines the axes for the entity axis2_placement_2d as x and y. Where the instance of coord_system_cartesian_2d represents y and z axes, then the attributes of axis2_placement_2d are reinterpreted accordingly. The labelling of the axes should be given using the attribute sign_convention inherited from the SUPERTYPE coord_system. If the attribute sign_convention has a NULL value, then the coord_system_cartesian_2d represents x and y axes.

EXPRESS specification:

```
*)
ENTITY coord_system_cartesian_2d
SUBTYPE OF (coord_system);
    axes_definition : axis2_placement_2d;
DERIVE
    origin_1 : REAL := axes_definition.location\cartesian_point.coordinates[1];
    origin_2 : REAL := NVL(axes_definition.location\cartesian_point.coordinates[2], 0.0);
WHERE
    WRC10 : SELF\coord_system.coord_system_dimensionality = 2;
    WRC11 : SIZEOF (axes_definition.location\cartesian_point.coordinates) = 2;
END_ENTITY;
(*
```

Attribute definitions:

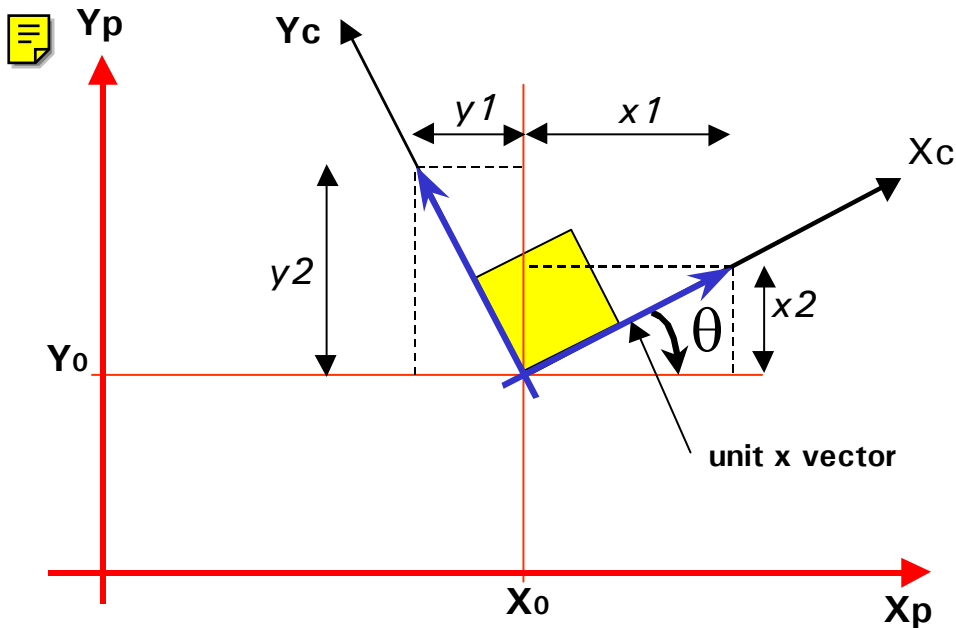
axes_definition

Declares the instance of axis2_placement_2d associated with the coord_system_cartesian_2d, which provides the definition of one of the 2 orthogonal axes located in 2 dimension space (the second axis is derived from the first). There must be one (and only one) instance of axis2_placement_2d referenced by each instance of coord_system_cartesian_2d. (See Figure 12.4.)

It should be noted that axis2_placement_2d inherits a reference to a cartesian_point from its SUPERTYPE placement. This cartesian_point is used as the origin for the coord_system_cartesian_2d. If the coord_system_cartesian_2d is not located within another (parent) coordinate system, then the cartesian_point shall be assigned the values (0.0, 0.0). Non-zero values represent a shift in the origin from the parent to the child coordinate system; i.e. the referenced cartesian_point provides the location of the origin of the coord_system_cartesian_2d within the parent coordinate system.

Further, the coordinate system may be rotated about its new origin relative to the parent coordinate system. This reorientation could occur even without a relocation of the origin; i.e. even when the origin shift is zero. The reorientation or transformation of the axes is defined using the instance of direction referenced by the attribute ref_direction of the entity axis2_placement_2d. The ref_direction defines the direction of the local x-axis. (For axes labelled y and z, the ref_direction is interpreted as defining the direction of the local y-axis.) The direction of the y (or z) axis is derived from the other normalized direction using the function build_2axes. Each direction is specified using 2 real numbers. The direction for the local x-axis will default to (1.0, 0.0) – the x unit vector – if values for ref_direction are not input.

An instance of `coord_system_cartesian_2d` may not refer to parent coordinate system of the type of `coord_system_cartesian_3d`. In other words, a 2D Cartesian coordinate system must be located within another 2D Cartesian coordinate system.



$$[T]^T = \begin{bmatrix} x1 & x2 \\ y1 & y2 \end{bmatrix} \begin{array}{l} \leftarrow \text{the x vector (from ref_direction)} \\ \leftarrow \text{the y vector (derived from build_2axes)} \end{array}$$

$$[T] = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \text{where } \theta \text{ is the angle of rotation from child to parent axes}$$

Figure 12.4 General transformation matrix for 2D coordinate systems
origin_1

This attribute derives the value of the first coordinate of the origin of the `coord_system_cartesian_2d` from the instance of `cartesian_point` referenced by the attribute location (inherited from the entity placement). When used here, the entity `cartesian_point` is defined by reference to 2 real number values through its attribute coordinates. The first of these real number values is taken as the first coordinate of the origin. For axes labelled x and y (or x and z), this is the x-coordinate of the origin. For axes labelled y and z, this is the y-coordinate of the origin.

origin_2

This attribute derives the value of the second coordinate of the origin of the `coord_system_cartesian_2d` from the instance of `cartesian_point` referenced by the attribute location (inherited from the entity placement). When used here, the entity `cartesian_point` is defined by reference to 2 real number values through its attribute coordinates. The second of these real number values (if provided) is taken as the second coordinate of the origin. If a value is not provided, this attribute is assigned a value of 0.0. For axes labelled x and y this is the y-coordinate of the origin. For axes labelled y and z (or x and z), this is the z-coordinate of the origin.

Formal propositions:**DERIVE**

The derived attributes origin_1 and origin_2 do not appear in the STEP Part 21 file.

WRC10

The value assigned to the attribute coord_system_dimensionality (inherited from the SUPERTYPE coord_system) shall equal 2. In other words, this coordinate system exists (and is defined in) two-dimensional space.

WRC11

The size of the list of real numbers referenced by the attribute coordinates of the entity cartesian_point (referenced by the by the attribute location of the entity placement) shall equal 2. In other words, the origin of this coordinate system is defined by two (and only two) coordinates.

Notes

New for CIS/2.

The entities axis2_placement_2d, placement, direction, and cartesian_point are all defined in STEP Part 42.

See Diagram 50 of the EXPRESS-G diagrams in Appendix B.

EXPRESS Unchanged for 2nd Edition – Figure 12.4 modified.

12.3.3 coord_system_cartesian_3d**Entity definition:**

A type of coordinate system, which is defined using 3 orthogonal axes, labelled x, y, and z. This entity references the STEP Part 42 geometric entity axis2_placement_3d to complete its definition.

Figure 12.5 illustrates two Cartesian coordinate systems, where one is the child of the other. This is accommodated in LPM/6 by an instance of coord_system_cartesian_3d AND coord_system_child. The transformation of the coordinate systems shown in Figure 12.5 is created through the reference to axis2_placement_3d. (See also the definition of the entity coord_system_child.)

EXPRESS specification:

*)

ENTITY coord_system_cartesian_3d

SUBTYPE OF (coord_system);

axes_definition : axis2_placement_3d;

DERIVE

origin_x : REAL := axes_definition.location\cartesian_point.coordinates[1];

origin_y : REAL := NVL(axes_definition.location\cartesian_point.coordinates[2], 0.0);

origin_z : REAL := NVL(axes_definition.location\cartesian_point.coordinates[3], 0.0);

WHERE

WRC12 : SELF\coord_system.coord_system_dimensionality = 3;

WRC13 : SIZEOF (axes_definition.location\cartesian_point.coordinates) = 3;

END_ENTITY;

(*

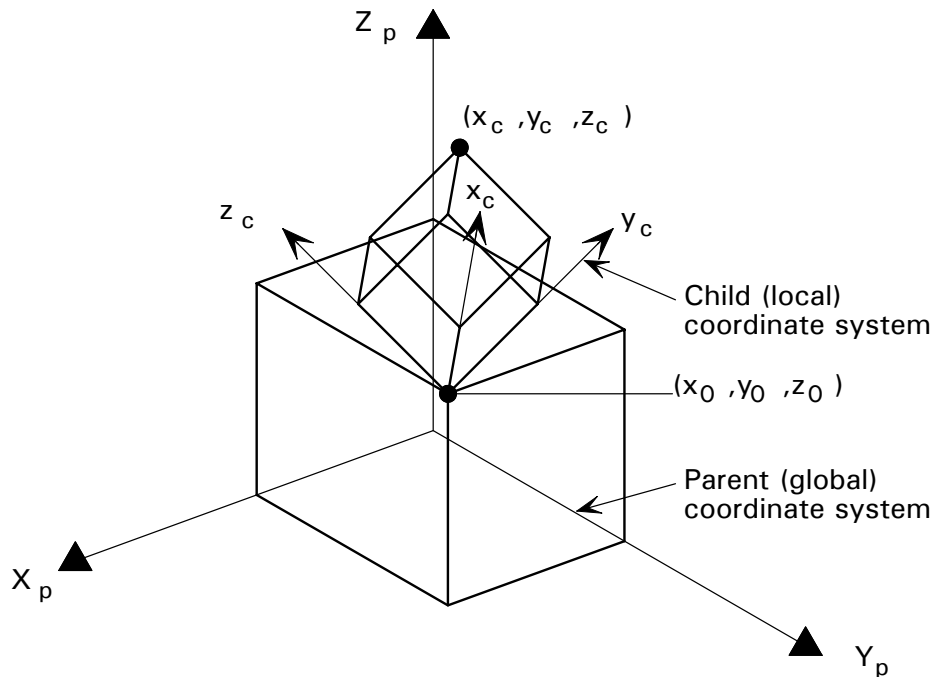


Figure 12.5 *Nested coordinate systems*

Attribute definitions:

axes_definition

Declares the instance of `axis2_placement_3d` associated with the `coord_system_cartesian_3d`, which provides the definition of 2 orthogonal axes located in 3 dimensional space (the third axis is derived from the other two). There must be one (and only one) instance of `axis2_placement_3d` referenced by each instance of `coord_system_Cartesian_3d`.

It should be noted that `axis2_placement_3d` inherits a reference to a `cartesian_point` from its SUPERTYPE `placement`. This `cartesian_point` is used as the origin for the `coord_system_cartesian_3d`. If the `coord_system_cartesian_3d` is not located within another (parent) coordinate system, then the `cartesian_point` shall be assigned the values (0.0, 0.0, 0.0). Non-zero values represent a shift in the origin from the parent to the child coordinate system; i.e. the referenced `cartesian_point` provides the location of the origin of the `coord_system_cartesian_3d` within the parent coordinate system.

Further, the coordinate system may be rotated about its new origin relative to the parent coordinate system. This reorientation could occur even without a relocation of the origin; i.e. even when the origin shift is zero. The reorientation or transformation of the axes is defined using the two instances of `direction` referenced by the attributes `axis` and `ref_direction` of the entity `axis2_placement_3d`. The attribute `axis` provides the exact direction for the local z axis, while the `ref_direction` is used to determine the direction of the local x -axis. The direction of the third (y) axis is derived from the two other normalized directions using the function `build_axes`. Each direction is specified using 3 real numbers. The `axis` (the direction for the local z axis) will default to (0.0, 0.0, 1.0) – the z unit vector – if values are not input.

An instance of `coord_system_cartesian_3d` may not refer to parent coordinate system of the type of `coord_system_cartesian_2d`. In other words, a 3D Cartesian coordinate system must be located within another 3D Cartesian coordinate system.

The reorientation may be described by a transformation matrix $[T]$, obtained from transformation of the derived attribute \mathbf{p} . That is, $[T] = \mathbf{p}^T$. The 9 real number values derived from \mathbf{p} are assigned as follows:

$$\mathbf{p} = \begin{bmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ z1 & z2 & z3 \end{bmatrix} \begin{array}{l} \leftarrow \text{the } x \text{ vector (from } \mathbf{ref_direction}) \\ \leftarrow \text{the } y \text{ vector (derived from } \mathbf{build_axes}) \\ \leftarrow \text{the } z \text{ vector (from } \mathbf{axis}) \end{array}$$

If the origin of child (local) coordinate system in Figure 12.5 is the point (x_o, y_o, z_o) in the parent (global) coordinate system, then a point in the child coordinate system (x_c, y_c, z_c) becomes (x_p, y_p, z_p) in the parent coordinate system, where;

$$\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} + \begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{bmatrix} \times \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

$$x_p = x_o + (x_c.x1 + y_c.y1 + z_c.z1)$$

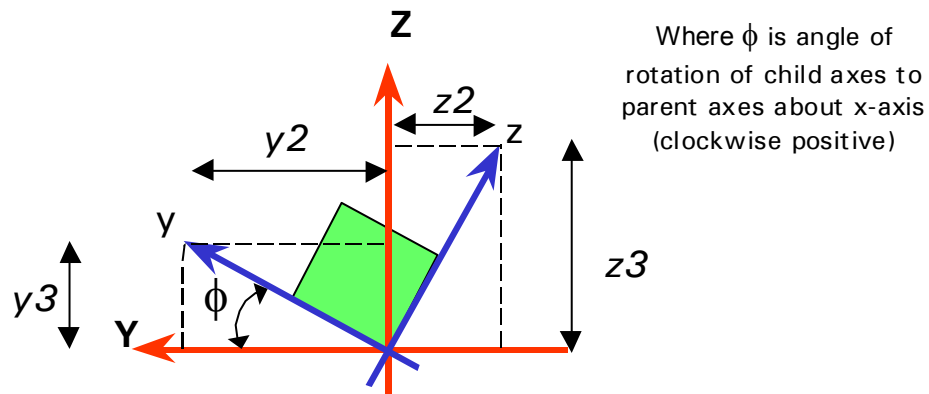
$$y_p = y_o + (x_c.x2 + y_c.y2 + z_c.z2)$$

$$z_p = z_o + (x_c.x3 + y_c.y3 + z_c.z3)$$

To transform the point from the parent to the child coordinate system, we need to invert the matrix, thus;

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = [T]^{-1} \times \begin{bmatrix} -x_o \\ -y_o \\ -z_o \end{bmatrix} + [T]^{-1} \times \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

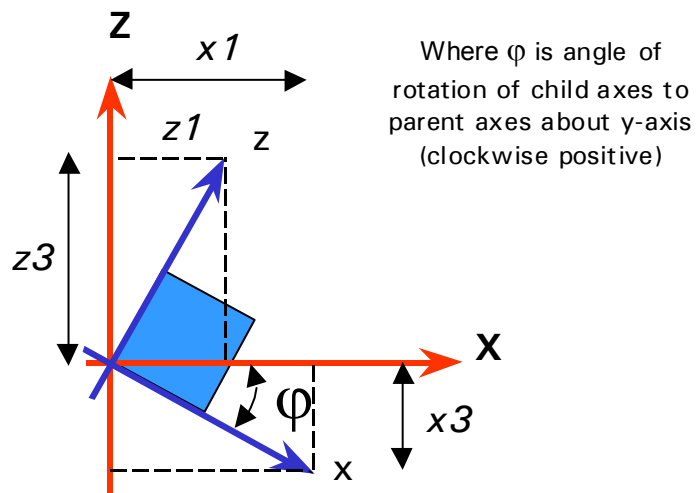
For rotation about one axis only, the transformations become more obvious as shown in Figure 12.6, Figure 12.8, and Figure 12.7, which illustrate rotations about the X, Y, and Z axes, respectively.



View on YZ plane (X-axis into page)

$$[T] = \begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

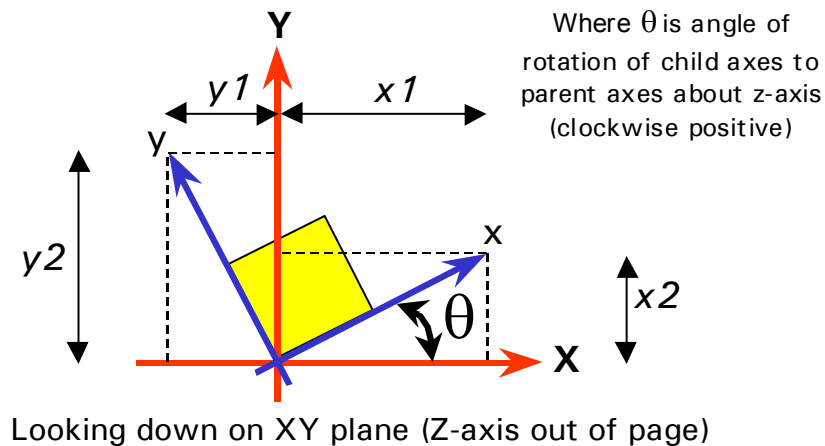
Figure 12.6 General transformation matrix for rotation about X-axis only



View on XZ plane (Y-axis into page)

$$[T] = \begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

Figure 12.7 General transformation matrix for rotation about Y-axis only



$$[T] = \begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 12.8 General transformation matrix for rotation about Z-axis only

origin_x

This attribute derives the value of the x-coordinate of the origin of the coord_system_cartesian_3d from the instance of cartesian_point referenced by the attribute location (inherited from the entity placement). When used here, the entity cartesian_point is defined by reference to 3 real number values through its attribute coordinates. The first of these real number values is taken as the x-coordinate of the origin.

origin_y

This attribute derives the value of the second coordinate of the origin of the coord_system_cartesian_3d from the instance of cartesian_point referenced by the attribute location (inherited from the entity placement). When used here, the entity cartesian_point is defined by reference to 3 real number values through its attribute coordinates. The second of these real number values (if provided) is taken as the y-coordinate of the origin. If a value is not provided, this attribute is assigned a value of 0.0.

origin_z

This attribute derives the value of the z-coordinate of the origin of the coord_system_cartesian_3d from the instance of cartesian_point referenced by the attribute location (inherited from the entity placement). When used here, the entity cartesian_point is defined by reference to 3 real number values through its attribute coordinates. The third of these real number values (if provided) is taken as the z-coordinate of the origin. If a value is not provided, this attribute is assigned a value of 0.0.

Formal propositions:**DERIVE**

The derived attributes `origin_x`, `origin_y`, and `origin_z` do not appear in the STEP Part 21 file.

WRC12

The value assigned to the attribute `coord_system_dimensionality` (inherited from the SUPERTYPE `coord_system`) shall equal 3. In other words, this coordinate system exists (and is defined in) three-dimensional space.

WRC13

The size of the list of real numbers referenced by the attribute coordinates of the entity `cartesian_point` (referenced by the attribute location of the entity placement) shall equal 3. In other words, the origin of this coordinate system is defined by three (and only three) coordinates.

Notes:

New for CIS/2; addressed by `COORD_SYSTEM` in CIS/1. The entity `TRANSFORMATION` in CIS/1 is replaced by the entity `axis2_placement_3d`. The 9 separate coefficient attributes of the entity `TRANSFORMATION` are replaced by the 3 `LISTs` of 3 `REALs` derived from the attributes of the entity `axis2_placement_3d`. Further, some of the values are `OPTIONAL`.

The entities `axis2_placement_3d`, `placement`, `direction`, and `cartesian_point` are all defined in STEP Part 42.

See diagram 50 of the EXPRESS-G diagrams in Appendix B. See also diagram 49 for the EXPRESS-G diagram of the placement and `axis2_placement_3d` constructs.

For 2nd Edition; EXPRESS unchanged, attribute `axes_definition` corrected, diagrams corrected (Figure 12.6, Figure 12.7, and Figure 12.8).

12.3.4 coord_system_child**Entity definition:**

A type of coordinate system that is located within another (parent) coordinate system. This entity allows for the definition of nested coordinate systems. The simplest case would be a local coordinate system located within a global coordinate system.

EXPRESS specification:

```

*)
ENTITY coord_system_child
  SUBTYPE OF (coord_system);
    parent_coord_system : coord_system;
  WHERE
    WRC14 : parent_coord_system :<>: (SELF);
    WRC15 : SELF.coord_system.coord_system_dimensionality <=
      parent_coord_system.coord_system_dimensionality;
  END_ENTITY;
(*

```

Attribute definitions:*parent_coord_system*

Declares the instance of `coord_system` associated with the `coord_system_child`, which is used to locate the `coord_system_child`; that is, the (global) coordinate system that acts as the parent of the (local) coordinate system. There must be one (and only one) instance of `coord_system` referenced by each instance of `coord_system_child`. The WHERE rule prevents instance recursion; that is, a child coordinate system cannot be a parent of itself. It does not prevent entity type recursion; that is, a child coordinate system can be a parent of another child coordinate system. This would produce a nested coordinate system.

Formal propositions:*WRC14*

The `coord_system_child` shall not act as its own `parent_coord_system`. This rule prevents instance recursion rather than entity type recursion. Thus, a child coordinate system can be a parent of another child coordinate system. This would produce a nested coordinate system.

WRC15

The value assigned to the attribute `coord_system_dimensionality` (inherited from the SUPERTYPE `coord_system`) of the `coord_system_child` shall be less than or equal to the value assigned to the attribute `coord_system_dimensionality` of the instance of `coord_system` referenced by the attribute `parent_coord_system`. In other words, the dimensionality of the child coordinate system must be less than or equal to that of the parent coordinate system. This allows a two-dimensional child coordinate system to be located within a three-dimensional parent coordinate system, while preventing a three-dimensional child coordinate system being located in a two-dimensional parent coordinate system. However, see the definitions of the entities `coord_system_cartesian_2d` and `coord_system_cartesian_3d`, which effectively demand equality of the dimensionality of the parent and child coordinate systems.

Informal propositions:

This entity is instanced when the parent coordinate system needs to be identified. The origin of the child coordinate system is located with the parent coordinate system, and the axes of child coordinate system are reoriented with respect to the parent coordinate system.

This entity can be instanced on its own. However, this would be of limited semantic value as the axes would not be defined. This entity would normally be instanced with one of its sibling SUBTYPES (e.g. `coord_system_cartesian_3d`). In such a case, the resulting complex entity instance would be encoded in a STEP Part 21 file using external mapping.

Notes:

New for CIS/2; addressed by `COORD_SYSTEM` in CIS/1.

See diagram 50 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

12.3.5 coord_system_cylindrical

Entity definition:

A type of coordinate system that is defined by an origin and a list of 3 axes. The Z-axis is taken to be the central longitudinal axis of the 'cylinder'. Cylindrical points are then defined from a base axis using a theta angle, a radial distance from the Z axis and a z coordinate. This entity references the STEP Part 42 geometric entities cylindrical_point and direction to complete its definition. (See Figure 12.9.)

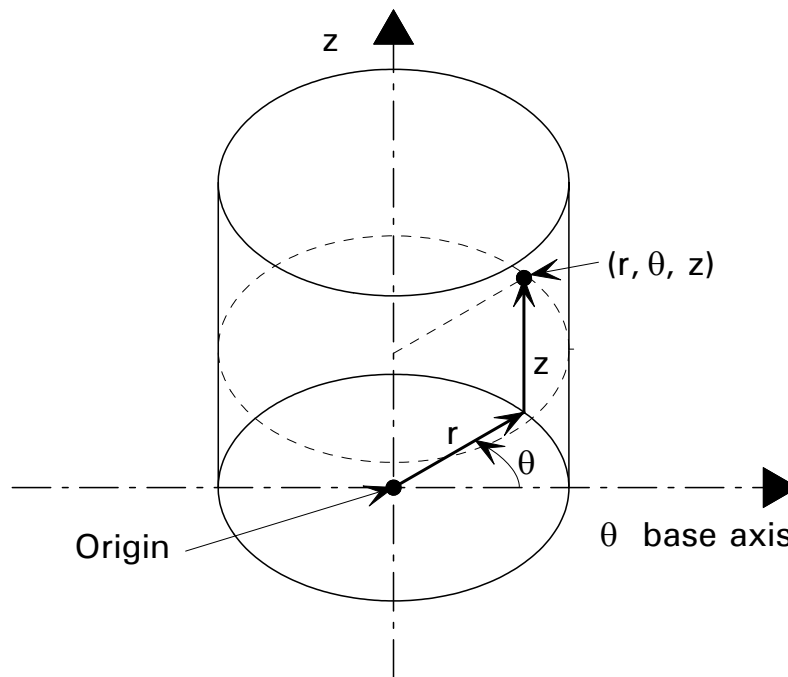


Figure 12.9 A cylindrical point defined in a cylindrical coordinate system

EXPRESS specification:

```
*)
ENTITY coord_system_cylindrical
SUBTYPE OF (coord_system);
  origin : cylindrical_point;
  axes_definition : LIST [2:3] OF direction;
END_ENTITY;
(*
```

Attribute definitions:

origin

Declares the instance of cylindrical_point associated with the coord_system_cylindrical, which provides the origin point of the cylindrical coordinate system. The origin may be based in a parent coordinate system if this entity is instantiated with the entity coord_system_child. For a stand alone cylindrical coordinate system, the origin would be specified as (0.0, 0.0, 0.0). There must be one (and only one) instance of cylindrical_point referenced by each instance of coord_system_cylindrical. However, the same cylindrical_point can act as the origin for several instances of coord_system_cylindrical.

axes_definition

Declares the list of 2 or 3 instances of direction associated with the coord_system_cylindrical, which provide the orientation of the 2 or 3 axes of the cylindrical coordinate system. For a stand alone cylindrical coordinate system, the directions would be specified as the unit vectors that make up the 3x3 transformation matrix:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Which is populated as ((1.0, 0.0, 0.0), (0.0, 1.0, 0.0), (0.0, 0.0, 1.0)). The definition of the axes may be based on an orientation from a parent coordinate system if this entity is instantiated with the sibling SUBTYPE entity coord_system_child. In this case, appropriate values would have to be provided for the transformation matrix. There must be three (and only three) instances of direction associated with each instance of coord_system_cylindrical. However, the same direction can be reused for several instances of coord_system_cylindrical. The LIST data type implies that the order of the instances is important. As can be seen from the example transformation matrix, the 3 instances of direction populate rows 1, 2 and 3 of the matrix in sequence, and the 3 real numbers populated for each instance of direction populate columns 1, 2 and 3 of the matrix in sequence.

Notes:

New for CIS/2.

See STEP Part 42 for the definition of the entities cylindrical_point and direction.

See diagram 50 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

12.3.6 coord_system_spherical***Entity definition:***

A type of coordinate system that is defined by an origin and a list of 3 axes. Spherical points are defined from an origin axis using a phi angle, a theta angle, and a radial distance. This entity references the STEP Part 42 geometric entities spherical_point and direction to complete its definition. (See Figure 12.10.)

EXPRESS specification:

```
*)
ENTITY coord_system_spherical
  SUBTYPE OF (coord_system);
    origin : spherical_point;
    axes_definition : LIST [3:3] OF direction;
END_ENTITY;
(*
```

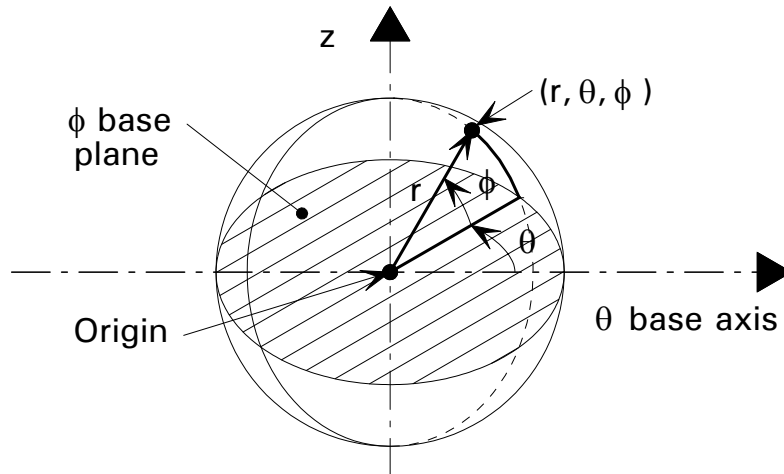


Figure 12.10 A spherical point located in a spherical coordinate system

Attribute definitions:

origin

Declares the instance of `spherical_point` associated with the `coord_system_spherical`, which provides the origin point of the spherical coordinate system. The origin may be based in a parent coordinate system if this entity is instantiated with the entity `coord_system_child`. For a stand alone spherical coordinate system, the origin would be specified as (0.0, 0.0, 0.0). There must be one (and only one) instance of `spherical_point` referenced by each instance of `coord_system_spherical`. However, the same `spherical_point` can act as the origin for several instances of `coord_system_spherical`.

axes_definition

Declares the list of 3 instances of direction associated with the `coord_system_cylindrical`, which provide the orientation of the 3 axes of the spherical coordinate system. For a stand alone spherical coordinate system, the directions would be specified as the unit vectors that make up the 3x3 transformation matrix:

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Which is populated as ((1.0, 0.0, 0.0), (0.0, 1.0, 0.0), (0.0, 0.0, 1.0)). The definition of the axes may be based on an orientation from a parent coordinate system if this entity is instantiated with the entity `coord_system_child`. In this case, appropriate values would have to be provided for the transformation matrix. There must be three (and only three) instances of direction associated with each instance of `coord_system_spherical`. However, the same direction can be reused for several instances of `coord_system_spherical`. The LIST data type implies that the order of the instances is important. As can be seen from the example transformation matrix, the 3 instances of direction populate rows 1, 2 and 3 of the matrix in sequence, and the 3 real numbers populated for each instance of direction populate columns 1, 2 and 3 of the matrix in sequence.

Notes:

New for CIS/2.

See STEP Part 42 for the definition of the entities `spherical_point` and `direction`.

See diagram 50 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

12.3.7 located_assembly**Entity definition:**

A type of located item that provides a coordinate system for an assembly and places that assembly within a structure. The assembly may also be given a position relative to a grid. This entity is effectively an intersection entity providing the association between the entities `coord_system`, `assembly` and one of the entities that may be chosen through the `SELECT` type `structure_select`.

EXPRESS specification:

*)

ENTITY `located_assembly`

SUPERTYPE OF (`located_assembly_child` ANDOR `located_assembly_marked`)

SUBTYPE OF (`located_item`);

`location_on_grid` : OPTIONAL SET [1:?] OF `grid_offset`;

`descriptive_assembly` : `assembly`;

`parent_structure` : `structure_select`;

DERIVE

`component_parts` : SET [0:?] OF `located_part` := `bag_to_set`
 (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
 LOCATED_PART.PARENT_ASSEMBLY'));

`sub_assemblies` : SET [0:?] OF `located_assembly` := `bag_to_set`
 (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
 LOCATED_ASSEMBLY.PARENT_STRUCTURE'));

UNIQUE

 URL2 : SELF\`located_item.location`, `descriptive_assembly`, `parent_structure`;

WHERE

 WRL22 : SELF\`located_item.location.coord_system_use`
 = 'Assembly Coordinate System';

 WRL46 : `parent_structure` :<>: (SELF);

END_ENTITY;

(*

Attribute definitions:***location_on_grid***

Declares the set of instances of `grid_offset` that may be associated with the `located_assembly` to provide the position of the assembly relative to a grid.

descriptive_assembly

Declares the instance of `assembly` associated with the `located_assembly`, for which the coordinate system is defined. There must be one (and only one) instance of `assembly` referenced by each instance of `located_assembly`. However, one instance of `assembly` (a specific level entity) may have many locations and may therefore be associated with many instances of `located_assembly` (an occurrence level entity).

parent_structure

Declares the instance of structure, located_structure, zone_of_structure, zone_of_building or other located_assembly associated with the located_assembly, to which the assembly belongs. The entity is referenced through the SELECT data type structure_select. There must be one (and only one) instance of one of the entities of the SELECT list referenced by each instance of located_assembly. This allow the assembly to be located within an unlocated structure, a located structure, a zone of a structure, a zone of a building or another located_assembly. Only when the assembly is located within a located assembly does the coordinate system of the assembly need to be located within the parent coordinate system of the located assembly.

component_parts

Derives the set of instances of located_part that reference this located_assembly via the attribute parent_assembly. Together with the attribute sub_assemblies, this attribute indicates the use of this located_assembly in the composition of the structure.

sub_assemblies

Derives the set of instances of located_assembly that reference this located_assembly via the attribute parent_structure. Together with the attribute component_parts, this attribute indicates the use of this located_assembly in the composition of the structure.

Formal propositions:*URL2*

The combination of the instances of assembly (referenced by the attribute descriptive_assembly) structure, located_structure, zone_of_structure, zone_of_building or located_assembly (referenced by the attribute parent_structure) and coord_system (referenced by the attribute location inherited from the SUPERTYPE located_item) shall be unique to this instance of located_assembly. (This prevents repetition of an association.) In other words, the located_assembly should have a unique location within the structure or building. This does not prevent the same (specific) assembly being located at two or more different places in the same structure.

WRL22

The attribute coord_system_use shall be assigned the value ‘Assembly Coordinate System’. (Note the rule is case sensitive – the value must be exactly as shown.)

WRL46

The attribute parent_structure shall not make a reference to this instance of located_assembly. In other words, a located_assembly cannot be located within itself.

Informal propositions:

If the assembly is located within another located_assembly, then the assembly’s coordinate system shall be declared as a child coordinate system that is located within the (parent) coordinate system of the higher level assembly. Furthermore, the attribute parent_structure shall refer to the same entity instance in both the higher and the lower level assemblies; i.e. both assemblies belong to the same structure. (See also located_assembly_child.)

If the assembly is located within a located_structure, then the assembly’s coordinate system shall be declared as a child coordinate system that is located within the (parent) coordinate system of the structure.

Notes:

New for CIS/2, partially addressed by ASSEMBLY in CIS/1.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE clause added.

12.3.8 located_assembly_child**Entity definition:**

A type of located_assembly that provides a coordinate system for an assembly_manufacturing_child and places that assembly within a (parent) assembly_manufacturing. The assembly may also be given a position relative to a grid. This entity is effectively an intersection entity providing the association between the entities coord_system, assembly_manufacturing_child and assembly_manufacturing.

EXPRESS specification:

*)

ENTITY located_assembly_child

SUBTYPE OF (located_assembly);

WHERE

WRL47 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING_CHILD'
IN TYPE OF (SELF\located_assembly.descriptive_assembly);

WRL48 : ('STRUCTURAL_FRAME_SCHEMA.LOCATED_ASSEMBLY' IN TYPE
OF (SELF\located_assembly.parent_structure)) AND
('STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN
TYPEOF (SELF\located_assembly.parent_structure.descriptive_assembly));

WRL49 : SELF\located_assembly.descriptive_assembly.parent_assembly :=:
SELF\located_assembly.parent_structure.descriptive_assembly;

WRL50 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN TYPE OF
(SELF\located_item.location);

WRL51 : SELF\located_item.location.parent_coord_system :=:
SELF\located_assembly.parent_structure\located_item.location;

WRL52 : NOT ((SIZEOF (SELF\located_assembly.component_parts) = 0)
AND (SIZEOF (SELF\located_assembly.sub_assemblies) = 0));

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE located_assembly. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity located_assembly to ensure that only an assembly_manufacturing_child is located within an assembly_manufacturing.

Formal propositions:**WRL47**

The instance of assembly referenced by the attribute descriptive_assembly of the (inherited from the SUPERTYPE entity located_assembly) shall be of the type assembly_manufacturing_child. In other words, an instance of located_assembly_child represents the location of an instance of an assembly_manufacturing_child. This implies that the located_assembly_child is one of the components of the physical model (a representation of the structure for manufacturing purposes).

WRL48

The entity instance chosen through the SELECT type structure_select (referenced by the attribute parent_structure) shall be of the type located_assembly, and the instance of assembly referenced by the attribute descriptive_assembly of the entity located_assembly (referenced by the attribute parent_structure) shall be of the type assembly_manufacturing. In other words, an instance of located_assembly_child represents the location of an instance of an assembly_manufacturing_child within (and is part of) an assembly_manufacturing.

WRL49

The instance of assembly_manufacturing referenced by the instance of assembly_manufacturing_child (by its attribute parent_assembly) shall be the same instance referenced by the attribute parent_structure. In other words, an instance of located_assembly_child places the assembly_manufacturing_child within the coordinate system of its parent assembly.

WRL50

The instance of coord_system referenced by the attribute location (inherited from the SUPERTYPE located_item) shall be of the type coord_system_child. In other words, the coordinate system associated with (and locating) this assembly_manufacturing_child must be a child coordinate system. This does not prevent the coordinate system being declared as a cartesian_coord_system_3d and a coord_system_child, since the SUPERTYPE coord_system is an ANDOR SUPERTYPE. When the instance of coord_system is encoded in a STEP Part 21 file external mapping will be required.

WRL51

The instance of coord_system referenced by the attribute parent_coord_system (of the entity coord_system_child) shall be the same instance of coord_system referenced by the attribute location of the entity located_assembly (referenced by the attribute parent_assembly). In other words, the coordinate system of the located_assembly that 'owns' this assembly_manufacturing_child must also act as the parent to the coordinate system of this assembly_manufacturing_child. That is, the assembly_manufacturing_child is located with respect to the assembly_manufacturing to which it belongs.

WRL52

The size of the set of instances of located_part (derived by the attribute component_parts) shall not be equal zero if the size of the set of instances of located_assembly (derived by the attribute sub-assemblies) is also equal to zero (and vice versa). In other words, the sub-assembly must be composed of either parts or sub-assemblies.

Notes:

New for CIS/2, partially addressed by ASSEMBLY in CIS/1.

See diagram 37 of the EXPRESS -G diagrams in Appendix B.

Modified for 2nd Edition – WHERE rules corrected, DERIVE clause added.

12.3.9 located_assembly_marked**Entity definition:**

A type of located_assembly assigned a set of parameters normally associated with assemblies at various stages in the production life. Only one of these attributes (assembly_mark) needs to be populated for an instance of located_assembly_marked to be

valid. However, other optional marks may be assigned as required at the appropriate stage in the production cycle.

EXPRESS specification:

```
*)
ENTITY located_assembly_marked
SUBTYPE OF (located_assembly);
    assembly_mark : label;
    client_mark : OPTIONAL label;
    prelim_mark : OPTIONAL label;
    shipping_mark : OPTIONAL label;
    bar_code : OPTIONAL label;
END_ENTITY;
(*
```

Attribute definitions:

assembly_mark

A short text reference for the assembly. A means of identifying the assembly within the fabrication shop.

client_mark

A short text reference that may be used to identify the assembly for the client.

prelim_mark

A short text reference that may be used to identify the assembly at the bidding or tender stages of the assembly's production life.

shipping_mark

A short text reference that may be used to identify the assembly at the shipping stage of the assembly's production life. A means of identifying the assembly once it leaves the fabrication shop.

bar_code

A numerical interpretation of the bar code associated with the assembly. The bar code is sometimes used to identify the assembly in the fabrication shop, during transportation, and during erection using a bar code scanner.

Notes

New for 2nd Edition (adds extra properties to located_assembly).

See Diagram 59 of the EXPRESS-G diagrams in Appendix B.

12.3.10 located_feature

Entity definition:

An abstract type of located item that provides the coordinate system for a feature. This entity is effectively an intersection entity providing the association between the entities coord_system and feature. The located_feature can be categorized (through its SUBTYPEs) as being located on either an assembly, a design_part, a located_assembly, a located_part, or a part. It is implied that the feature is therefore applied to each of the different items. In this way, a generic feature can be applied to a generic part or to a specific part. Similarly, a specific feature can be applied to a located_part (an occurrence level entity).

EXPRESS specification:

```

*)
ENTITY located_feature
ABSTRACT SUPERTYPE OF (ONEOF
    (located_feature_for_assembly,
    located_feature_for_design_part,
    located_feature_for_located_assembly,
    located_feature_for_located_part,
    located_feature_for_part))
SUBTYPE OF (located_item);
    descriptive_feature : feature;
WHERE
    WRL23 : SELFlocated_item.location.coord_system_use =
        'Feature Coordinate System';
END_ENTITY;
(*

```

Attribute definitions:*descriptive_feature*

Declares the instance of feature associated with the located_feature, for which the coordinate system is provided. There must be one (and only one) instance of feature referenced by each instance of located_feature. However, one instance of feature (a specific level entity) may have many locations and may therefore be associated with many instances of located_feature (an occurrence level entity).

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

WRL23

The attribute of the instance of coord_system referenced by the attribute location (inherited from the SUPERTYPE located_item) shall be assigned the value 'Feature Coordinate System'. (Note, the rule is case sensitive, the value must be exactly as shown.)

Notes:

Known as LOCATED_FEATURE in CIS/1; SUBTYPEs added.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.11 located_feature_for_assembly**Entity definition:**

A type of located feature where the coordinate system is provided for a feature that is applied to an assembly. This entity is effectively an intersection entity providing the association between the entities coord_system, feature and assembly.

EXPRESS specification:

```

*)
ENTITY located_feature_for_assembly
SUBTYPE OF (located_feature);
    modified_assembly : assembly;
UNIQUE
    URL3 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
        modified_assembly;
END_ENTITY;
(*

```

Attribute definitions:*modified_assembly*

Declares the instance of assembly associated with the located_feature_for_assembly, to which the feature is applied. There must be one (and only one) instance of assembly referenced by each instance of located_feature_for_assembly. However, one instance of assembly (a specific level entity) may have many modifications and may therefore be associated with many instances of located_feature_for_assembly (an occurrence level entity).

Formal propositions:*URL3*

The combination of the instances of assembly (referenced by the attribute modified_assembly), feature (referenced by the attribute descriptive_feature inherited from the SUPERTYPE located_feature) and coord_system (referenced by the attribute location inherited from the SUPERTYPE located_item) shall be unique to this instance of located_feature_for_assembly. (This prevents repetition of an association.) In other words, the located feature should have a unique location upon the assembly. This does not prevent the same (specific) feature being applied to two or more different places on the same assembly.

Notes:

New for CIS/2.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.12 located_feature_for_design_part**Entity definition:**

A type of located feature where the coordinate system is provided for a feature that is applied to a design_part. This entity is effectively an intersection entity providing the association between the entities coord_system, feature and design_part.

EXPRESS specification:

```

*)
ENTITY located_feature_for_design_part
SUBTYPE OF (located_feature);
    modified_part : design_part;
UNIQUE

```

```

        URL4 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
            modified_part;
    END_ENTITY;
    (*

```

Attribute definitions:

modified_part

Declares the instance of *design_part* associated with the *located_feature_for_design_part*, to which the feature is applied. There must be one (and only one) instance of *design_part* referenced by each instance of *located_feature_for_design_part*. However, one instance of *design_part* may have many modifications and may therefore be associated with many instances of *located_feature_for_design_part*. **Formal propositions:**

URL4

The combination of the instances of *design_part* (referenced by the attribute *modified_part*), *feature* (referenced by the attribute *descriptive_feature* inherited from the SUPERTYPE *located_feature*), and *coord_system* (referenced by the attribute *location* inherited from the SUPERTYPE *located_item*) shall be unique to this instance of *located_feature_for_design_part*. (This prevents repetition of an association.) In other words, the located feature should have a unique location upon the *design_part*. This does not prevent the same (specific) feature being applied to two or more different places on the same *design_part*.

Notes:

New for CIS/2.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.13 located_feature_for_located_assembly

Entity definition:

A type of located feature where the coordinate system is provided for a feature that is applied to a *located_assembly*. This entity is effectively an intersection entity providing the association between the entities *coord_system*, *feature* and *located_assembly*.

EXPRESS specification:

```

    *)
    ENTITY located_feature_for_located_assembly
    SUBTYPE OF (located_feature);
        modified_assembly : located_assembly;
    UNIQUE
        URL5 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
            modified_assembly;
    WHERE
        WRL24 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN TYPE OF
            (SELF\located_item.location);
        WRL25 : SELF\located_item.location.parent_coord_system :=:
            modified_assembly\located_item.location;
    END_ENTITY;
    (*

```

Attribute definitions:***modified_assembly***

Declares the instance of `located_assembly` associated with the `located_feature_for_located_assembly`, to which the feature is applied. There must be one (and only one) instance of `located_assembly` referenced by each instance of `located_feature_for_located_assembly`. However, one instance of `located_assembly` (an occurrence level entity) may have many modifications and may therefore be associated with many instances of `located_feature_for_located_assembly` (an occurrence level entity).

Formal propositions:***URL5***

The combination of the instances of `located_assembly` (referenced by the attribute `modified_assembly`), `feature` (referenced by the attribute `descriptive_feature` inherited from the SUPERTYPE `located_feature`) and `coord_system` (referenced by the attribute `location` inherited from the SUPERTYPE `located_item`) shall be unique to this instance of `located_feature_for_located_assembly`. (This prevents repetition of an association.) In other words, the located feature should have a unique location upon the located assembly. This does not prevent the same (specific) feature being applied to two or more different places on the same located assembly.

WRL24

The instance of `coord_system` referenced by the attribute `location` (inherited from the SUPERTYPE `located_item`) shall be of the type `coord_system_child`. In other words, the coordinate system associated with (and locating) this feature must be a child coordinate system. This does not prevent the coordinate system being declared as a `cartesian_coord_system_3d` and a `coord_system_child`, since the SUPERTYPE `coord_system` is an ANDOR SUPERTYPE. When the instance of `coord_system` is encoded in a STEP Part 21 file external mapping will be required.

WRL25

The instance of `coord_system` referenced by the attribute `parent_coord_system` (of the entity `coord_system_child`) shall be the same instance of `coord_system` referenced by the attribute `location` of the entity `located_assembly` (referenced by the attribute `modified_assembly`). In other words, the coordinate system of the `located_assembly` modified by this feature must also act as the parent to the coordinate system of this feature. That is, the feature is located with respect to the assembly it modifies.

Notes:

New for CIS/2.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.14 located_feature_for_located_part**Entity definition:**

A type of located feature where the coordinate system is provided for a feature that is applied to a `located_part`. This entity is effectively an intersection entity providing the association between the entities `coord_system`, `feature` and `located_part`.

EXPRESS specification:

*)

ENTITY located_feature_for_located_part

SUPERTYPE OF (located_feature_joint_dependent)

SUBTYPE OF (located_feature);

modified_part : located_part;

UNIQUE

URL6 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
modified_part;

WHERE

WRL26 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN TYPE OF
(SELF\located_item.location);WRL27 : SELF\located_item.location.parent_coord_system :=:
modified_part\located_item.location;

END_ENTITY;

(*)

Attribute definitions:*modified_part*

Declares the instance of located_part associated with the located_feature_for_located_part, to which the feature is applied. There must be one (and only one) instance of located_part referenced by each instance of located_feature_for_located_part. However, one instance of located_part (an occurrence level entity) may have many modifications and may therefore be associated with many instances of located_feature_for_located_part (an occurrence level entity).

Formal propositions:*URL6*

The combination of the instances of located_part (referenced by the attribute modified_part), feature (referenced by the attribute descriptive_feature inherited from the SUPERTYPE located_feature) and coord_system (referenced by the attribute location inherited from the SUPERTYPE located_item) shall be unique to this instance of located_feature_for_located_part. (This prevents repetition of an association.) In other words, the located feature should have a unique location upon the located_part. This does not prevent the same (specific) feature being applied to two or more different places on the same located_part.

WRL26

The instance of coord_system referenced by the attribute location (inherited from the SUPERTYPE located_item) shall be of the type coord_system_child. In other words, the coordinate system associated with (and locating) this feature must be a child coordinate system. This does not prevent the coordinate system being declared as a cartesian_coord_system_3d and a coord_system_child, since the SUPERTYPE coord_system is an ANDOR SUPERTYPE. When the instance of coord_system is encoded in a STEP Part 21 file external mapping will be required.

WRL27

The instance of coord_system referenced by the attribute parent_coord_system (of the entity coord_system_child) shall be the same instance of coord_system referenced by the attribute location of the entity located_part (referenced by the attribute modified_part). In other words, the coordinate system of the located_part modified by this feature must also

act as the parent to the coordinate system of this feature. That is, the feature is located with respect to the part it modifies.

Notes:

New for CIS/2; addressed by LOCATED_PART in CIS/1.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.15 located_feature_for_part

Entity definition:

A type of located feature where the coordinate system is provided for a feature that is applied to a (specific) part. This entity is effectively an intersection entity providing the association between the entities coord_system, feature and part.

EXPRESS specification:

*)

ENTITY located_feature_for_part

SUBTYPE OF (located_feature);

modified_part : part;

UNIQUE

URL7 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
modified_part;

END_ENTITY;

(*

Attribute definitions:

modified_part

Declares the instance of part associated with the located_feature_for_part, to which the feature is applied. There must be one (and only one) instance of part referenced by each instance of located_feature_for_part. However, one instance of part (a specific level entity) may have many modifications and may therefore be associated with many instances of located_feature_for_part (an occurrence level entity).

Formal propositions:

URL7

The combination of the instances of part (referenced by the attribute modified_part), feature (referenced by the attribute descriptive_feature inherited from the SUPERTYPE located_feature) and coord_system (referenced by the attribute location inherited from the SUPERTYPE located_item) shall be unique to this instance of located_feature_for_part. (This prevents repetition of an association.) In other words, the located feature should have a unique location upon the part. This does not prevent the same (specific) feature being applied to two or more different places on the same part.

Informal propositions:

Located features are normally located within the coordinate system of the part to which they are applied. Since a part does not have an explicitly defined coordinate system, the coordinate system used with the located_feature_for_part would normally be defined without reference to a parent coordinate system. The feature is therefore defined in the implied local coordinate system of the part.

Parts acquire an explicit coordinate system through their use with the entity `located_part`. If the coordinate system used with the `located_feature_for_part` is declared as a child coordinate system, then the feature coordinate system it is located within the coordinate system specified by the `located_part` referring to the part.

Notes:

New for CIS/2.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.16 `located_feature_joint_dependent`

Entity definition:

A type of `located_feature_for_located_part` where the coordinate system is provided for a feature that is applied to a `located_part` and that feature is required to accommodate a joint system. This entity is effectively an intersection entity providing the association between the entities `coord_system`, `feature`, `located_part` and `located_joint_system`.

EXPRESS specification:

```
*)
ENTITY located_feature_joint_dependent
SUBTYPE OF (located_feature_for_located_part);
    feature_for_joint : located_joint_system;
END_ENTITY;
(*
```

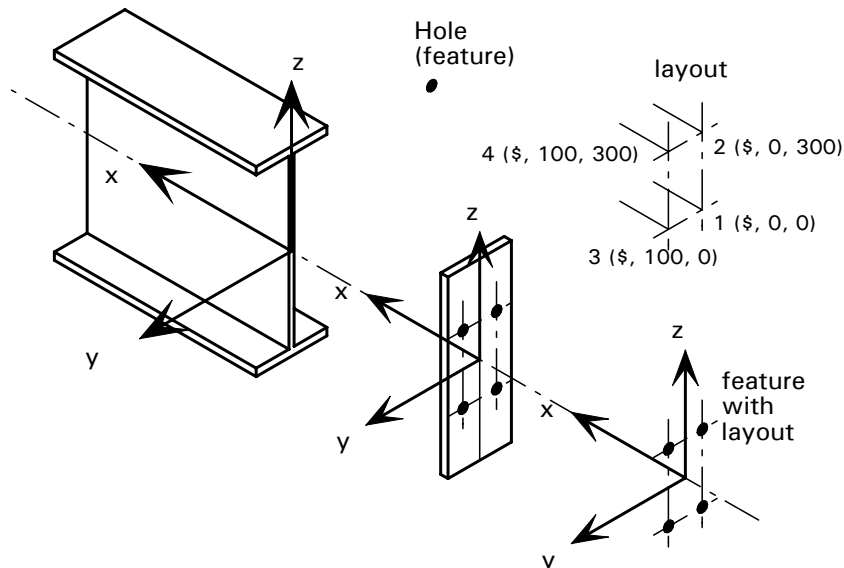


Figure 12.11 *An example of a `located_feature_joint_dependent`*

Attribute definitions:

feature_for_joint

Declares the instance of `located_joint_system` associated with the `located_feature_joint_dependent`. It is implied that it is this instance of `located_joint_system` that demands the feature to be applied to the `located_part`. For example, a joint system composed of 4 M20 bolts will require a feature composed of 4

No 22mm diameter holes to be applied to the connected part. This requires the use of the entity `feature_volume_with_layout`, which places one hole at four positions.

There must be one (and only one) instance of `located_joint_system` referenced by each instance of `located_feature_joint_dependent`. However, one instance of `located_joint_system` may require many modifications to be made to several `located_parts` and may therefore be associated with many instances of `located_feature_joint_dependent`.

Notes:

Known as `JOINT_DEP_FEATURE` in CIS/1.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.17 `located_item`

Entity definition:

An abstract SUPERTYPE entity and a type of `structural_frame_item` that provides the location of an item through a reference to `coord_system`. The entity must be categorized (through its SUBTYPES) as providing the location of an assembly, a feature, a joint_system, a part, a site, or a structure.

As a SUBTYPE of `structural_frame_item`, a `located_item` inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a `located_item` may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A `located_item` may also be related to another `located_item` (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

EXPRESS specification:

*)

ENTITY `located_item`

ABSTRACT SUPERTYPE OF (ONEOF

(`located_assembly`,
`located_feature`,
`located_joint_system`,
`located_part`,
`located_site`,
`located_structure`))

SUBTYPE OF (`structural_frame_item`);

`location` : `coord_system`;

END_ENTITY;

(*

Attribute definitions:*location*

Declares the instance of coord_system associated with the located_item, which provides the location of the item specified in the SUBTYPE. There must be one (and only one) instance of coord_system referenced by each instance of located_item. However, an instance of coord_system may provide the location for many instances of located_item.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Informal propositions:

The purpose of this entity is to specialize the SUPERTYPE structural_frame_item and to bring together the SUBTYPES under a common category. As a SUBTYPE of structural_frame_item this entity inherits the attributes item_number, item_name and item_description. It may also be used (and referenced) in the same way as other SUBTYPES of structural_frame_item; e.g. it may be the subject of similar assignments.

Notes:

New for CIS/2.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.18 located_joint_system**Entity definition:**

A type of located item that provides a coordinate system for a joint_system and places that joint_system within an assembly. The located_joint_system may also be declared as being assembled either on the construction site or in the fabrication shop. This entity is effectively an intersection entity providing the association between the entities coord_system, joint_system and located_assembly.

EXPRESS specification:

*)

ENTITY located_joint_system

SUBTYPE OF (located_item);

 descriptive_joint_system : joint_system;

 parent_assembly : located_assembly;

UNIQUE

 URL8 : SELF\located_item.location, descriptive_joint_system, parent_assembly;

WHERE

 WRL28 : SELF\located_item.location.coord_system_use =
 'Joint System Coordinate System';

 WRL29 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
 TYPE OF(SELF\located_item.location);

 WRL30 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN
 TYPE OF(parent_assembly.descriptive_assembly);

 WRL31 : SELF\located_item.location.parent_coord_system

```

      :=: parent_assembly\located_assembly\located_item.location;
END_ENTITY;
(*)

```

Attribute definitions:

descriptive_joint_system

Declares the instance of joint_system associated with the located_joint_system, for which the coordinate system is defined. There must be one (and only one) instance of joint_system referenced by each instance of located_joint_system. However, one instance of joint_system (a specific level entity) may have many locations and may therefore be associated with many instances of located_joint_system (an occurrence level entity).

parent_assembly

Declares the instance of located_assembly associated with the located_joint_system, to which the joint_system belongs. There must be one (and only one) instance of located_assembly referenced by each instance of located_joint_system. The located_assembly can, of course, contain many joint systems and can therefore be referenced by many instances of located_joint_system.

Formal propositions:

URL8

The combination of the instances of located_assembly (referenced by the attribute parent_assembly), joint_system (referenced by the attribute descriptive_joint_system) and coord_system (referenced by the attribute location inherited from the SUPERTYPE located_item) shall be unique to this instance of located_joint_system. (This prevents repetition of an association.) In other words, the joint system should have a unique location within the located_assembly. This does not prevent the same (specific) joint_system being located at two or more different places within the same located assembly.

WRL28

The attribute of the instance of coord_system referenced by the attribute location (inherited from the SUPERTYPE located_item) shall be assigned the value 'Joint System Coordinate System'. (Note, the rule is case sensitive, the value must be exactly as shown.)

WRL29

The instance of coord_system referenced by the attribute location (inherited from the SUPERTYPE located_item) shall be of the type coord_system_child. In other words, the coordinate system associated with (and locating) this joint_system must be a child coordinate system. This does not prevent the coordinate system being declared as a cartesian_coord_system_3d and a coord_system_child, since the SUPERTYPE coord_system is an ANDOR SUPERTYPE. When the instance of coord_system is encoded in a STEP Part 21 file external mapping will be required.

WRL30

The instance of assembly referenced by the attribute descriptive_assembly of the entity located_assembly (referenced by the attribute parent_assembly) shall be of the type assembly_manufacturing. In other words, the joint_system is located within (and is part of) an assembly_manufacturing. This implies that the located_joint_system is one of the components of the physical model (a representation of the structure for manufacturing purposes).

WRL31

The instance of `coord_system` referenced by the attribute `parent_coord_system` (of the entity `coord_system_child`) shall be the same instance of `coord_system` referenced by the attribute `location` of the entity `located_assembly` (referenced by the attribute `parent_assembly`). In other words, the coordinate system of the `located_assembly` that 'owns' this joint system must also act as the parent to the coordinate system of this joint system. That is, the `joint_system` is located with respect to the assembly to which it belongs.

Notes:

Known as `LOCATED_JOINT_SYSTEM` in CIS/1.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.19 located_part**Entity definition:**

A type of located item that provides a coordinate system for a part and places that part within an `assembly_manufacturing`. This entity is effectively an intersection entity providing the association between the entities `coord_system`, `part` and `assembly_manufacturing`.

EXPRESS specification:

```

*)
ENTITY located_part
SUBTYPE OF (located_item);
    descriptive_part : part;
    parent_assembly : located_assembly;
UNIQUE
    URL9 : SELF\located_item.location, descriptive_part, parent_assembly;
WHERE
    WRL32 : SELF\located_item.location.coord_system_use =
        'Part Coordinate System';
    WRL33 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
        TYPE OF (SELF\located_item.location);
    WRL34 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING'
        IN TYPE OF (parent_assembly.descriptive_assembly);
    WRL35 : SELF\located_item.location.parent_coord_system :=:
        parent_assembly\located_assembly\located_item.location;
END_ENTITY;
(*

```

Attribute definitions:**descriptive_part**

Declares the instance of `part` associated with the `located_part`, for which the coordinate system is defined. There must be one (and only one) instance of `part` referenced by each instance of `located_part`. However, one instance of `part` (a specific level entity) may have many locations and may therefore be associated with many instances of `located_part` (an occurrence level entity).

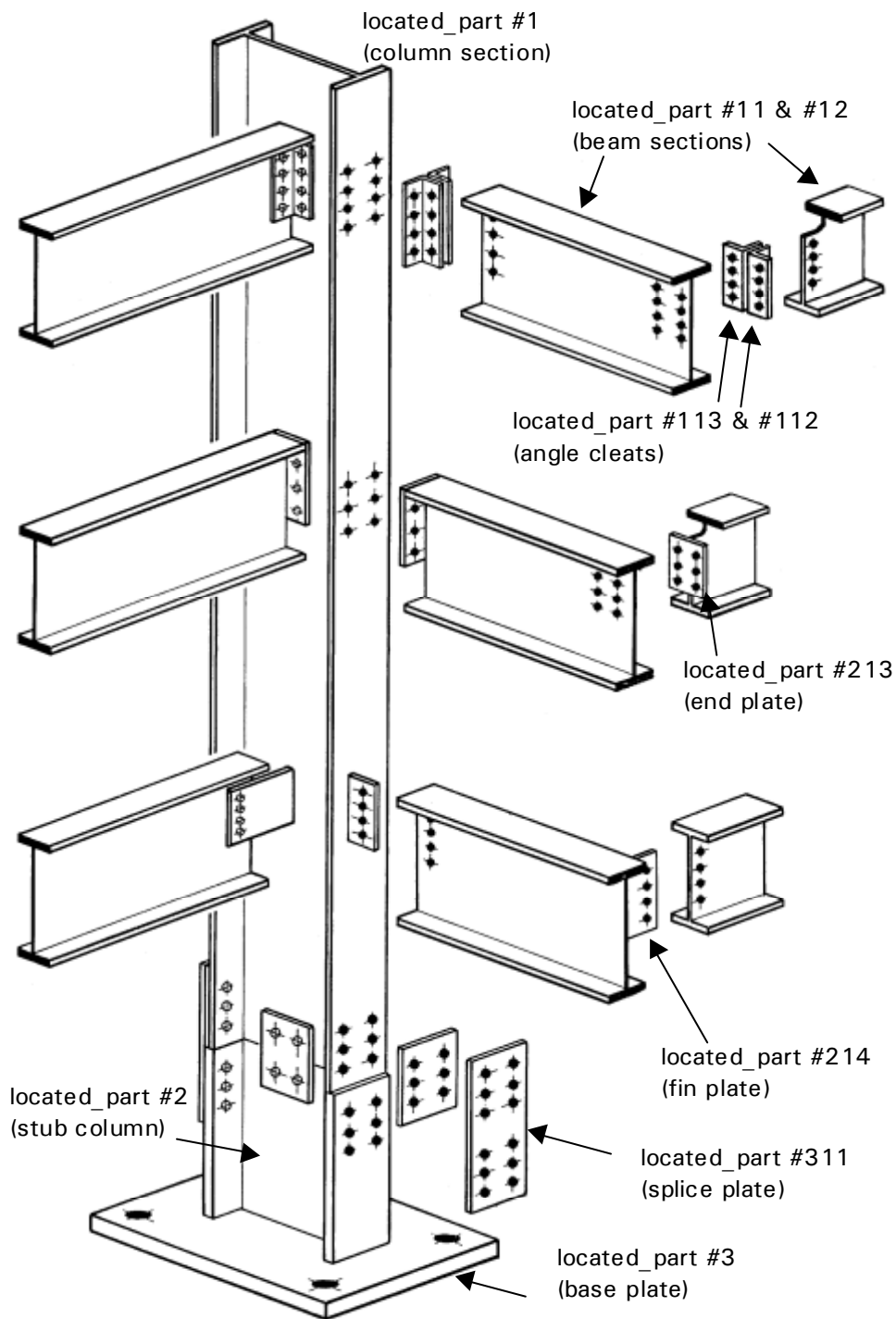


Figure 12.12 *Located parts forming an assembly*

parent_assembly

Declares the instance of `assembly_manufacturing` associated with the `located_part`, to which the part belongs. There must be one (and only one) instance of `assembly_manufacturing` referenced by each instance of `located_part`. The `assembly_manufacturing` can, of course, contain many parts and can therefore be referenced by many instances of `located_part`.

Formal propositions:**URL9**

The combination of the instances of `located_assembly` (referenced by the attribute `parent_assembly`), `part` (referenced by the attribute `descriptive_part`) and `coord_system` (referenced by the attribute `location` inherited from the SUPERTYPE `located_item`) shall be unique to this instance of `located_part`. (This prevents repetition of an association.) In other words, the part should have a unique location within the `located_assembly`. This does not prevent the same (specific) part being located at two or more different places within the same located assembly.

WRL32

The attribute of the instance of `coord_system` referenced by the attribute `location` (inherited from the SUPERTYPE `located_item`) shall be assigned the value 'Part Coordinate System'. (Note, the rule is case sensitive, the value must be exactly as shown.)

WRL33

The instance of `coord_system` referenced by the attribute `location` (inherited from the SUPERTYPE `located_item`) shall be of the type `coord_system_child`. In other words, the coordinate system associated with (and locating) this part must be a child coordinate system. This does not prevent the coordinate system being declared as a `cartesian_coord_system_3d` and a `coord_system_child`, since the SUPERTYPE `coord_system` is an ANDOR SUPERTYPE. When the instance of `coord_system` is encoded in a STEP Part 21 file external mapping will be required.

WRL34

The instance of `assembly` referenced by the attribute `descriptive_assembly` of the entity `located_assembly` (referenced by the attribute `parent_assembly`) shall be of the type `assembly_manufacturing`. In other words, the part is located within (and is part of) an `assembly_manufacturing`. This implies that the `located_part` is one of the components of the physical model (a representation of the structure for manufacturing purposes).

WRL35

The instance of `coord_system` referenced by the attribute `parent_coord_system` (of the entity `coord_system_child`) shall be the same instance of `coord_system` referenced by the attribute `location` of the entity `located_assembly` (referenced by the attribute `parent_assembly`). In other words, the coordinate system of the `located_assembly` that 'owns' this part must also act as the parent to the coordinate system of this joint system. That is, the part is located with respect to the assembly to which it belongs.

Notes:

Known as `LOCATED_PART` in CIS/1.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.20 located_part_joint**Entity definition:**

A logical connection between a pair of `located_parts` which require the existence of one or more `located_joint_systems`. It defines the low level topology of a structure; i.e. `located_parts` are linked by a network of `located_part_joints`. The `located_part_joint` must

be given a label and may also be given a description of its nature. This entity is effectively an intersection entity providing the association between the entities `located_part` and `located_joint_system`.

EXPRESS specification:

```
*)
ENTITY located_part_joint;
    part_joint_label : label;
    part_joint_nature : OPTIONAL text;
    logically_joined_parts : SET [2:2] OF located_part;
    required_joints : SET [1:?] OF located_joint_system;
END_ENTITY;
(*
```

Attribute definitions:

part_joint_label

A short text reference used to identify the `located_part_joint`.

part_joint_nature

An optional text description of the functional role of the (logical) `located_part_joint`.

logically_joined_parts

Declares the set of 2 separate instances of `located_part` associated with the `located_part_joint`, which are logically joined by the `located_part_joint`. There must be two (and only two) separate instances of `located_part` referenced by each instance of `located_part_joint`. (The SET data type prohibits repetition of the members of the aggregation; thus the two instances must be separate.)

required_joints

Declares the set of 1 or more separate instances of `located_joint_system` associated with the `located_part_joint`, which physically fulfil the (logical) `located_part_joint`. There must be at least one instance of `located_joint_system` referenced by each instance of `located_part_joint`. (The SET data type prohibits repetition of the members of the aggregation; thus the instances must be separate.)

Notes:

Known as `PART_JOINT` in CIS/1.

This is not a SUBTYPE of `located_item`. It does not have a location, and therefore it does not have an associated `coord_system`. It represents the logical connectivity, rather than physical connections, but is an aspect of the physical model that may be used during the manufacturing process. It may be used to map the design intent specified by `assembly_design_structural_connection` onto the physical pieces that form the structure.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

12.3.21 `located_part_marked`

Entity definition:

A type of `located_part` assigned a set of parameters normally associated with parts (pieces) at various stages in the production life. Only one of these attributes (`main_part`) needs to be populated for an instance of `located_part_marked` to be valid. However,

other optional marks may be assigned as required at the appropriate stage in the production cycle.

EXPRESS specification:

```
*)
ENTITY located_part_marked
SUBTYPE OF (located_part);
    piece_mark : OPTIONAL label;
    prelim_mark : OPTIONAL label;
    bar_code : OPTIONAL label;
    quantity : OPTIONAL INTEGER;
    main_piece : LOGICAL;
END_ENTITY;
(*
```

Attribute definitions:

piece_mark

A short text reference for the located_part (piece). A means of identifying the piece within the fabrication shop.

prelim_mark

A short text reference that may be used to identify the located_part (piece) at the bidding or tender stages of its production life.

bar_code

A numerical interpretation of the bar code associated with the located_part (piece). The bar code is sometime used to identify the piece in the fabrication shop, during transportation, and during erection using a bar code scanner.

quantity

An optional number that may be used to declare the quantity of pieces of the same type as this instance of located_part_marked.

main_piece

A logical flag providing an indication as to the position of this piece in the assembly composition. This attribute is set to TRUE if this instance of located_part_marked forms the basis of the assembly to which other pieces are added; FALSE if it is attached to the main piece in a subordinate manner (e.g. a fitting); and UNKNOWN if the assembly sequence is unknown. Only one piece in each assembly should be flagged as being the main piece.

Notes

New for 2nd Edition (add extra properties to located_part) .

See Diagram 59 of the EXPRESS-G diagrams in Appendix B.

12.3.22 located_site

Entity definition:

A type of located_item that provides the coordinate system for a construction site. The site must also be associated with a particular project, either directly, or indirectly through the zone_of_project entity. This entity is effectively an intersection entity providing the

association between the entities `coord_system`, `site` and `project`, resolving the many-to-many relationship between `project` and `site`.

EXPRESS specification:

```

*)
ENTITY located_site
SUBTYPE OF (located_item);
    descriptive_site : site;
    parent_project : project_select;
UNIQUE
    URL10 : SELF\located_item.location, descriptive_site, parent_project;
WHERE
    WRL36 : SELF\located_item.location.coord_system_use =
        'Site Coordinate System';
END_ENTITY;
(*

```

Attribute definitions:

descriptive_site

Declares the instance of `site` associated with the `located_site`, for which the coordinate system is defined. There must be one (and only one) instance of `site` referenced by each instance of `located_site`. However, the same `site` (a specific level entity) can be referenced by more than one instance of `located_site` (an occurrence level entity).

parent_project

Declares the instance of `project` or `zone_of_project` associated with the `located_site`, to which the site belongs. The reference is made through the `SELECT` type `project_select`. There must be one (and only one) instance of `project` or `zone_of_project` referenced by each instance of `located_site`. However, there is no restriction in the `INVERSE` direction. For example, the same `project` can be referenced by more than one instance of `located_site`. In this way, a `project` may involve many `sites` and a `site` may be involved with many `projects`.

Formal propositions:

URL10

The combination of the instances of `project` or `zone_of_project` (referenced by the attribute `parent_project`), `site` (referenced by the attribute `descriptive_site`) and `coord_system` (referenced by the attribute `location` inherited from the SUPERTYPE `located_item`) shall be unique to this instance of `located_site`. (This prevents repetition of an association.) In other words, for a given `project` or `zone of project`, the `site` can only have one coordinate system.

WRL36

The attribute of the instance of `coord_system` referenced by the attribute `location` (inherited from the SUPERTYPE `located_item`) shall be assigned the value 'Site Coordinate System'. (Note, the rule is case sensitive, the value must be exactly as shown.)

Notes:

New for CIS/2.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

12.3.23 located_structure

Entity definition:

A type of located_item that provides the coordinate system for a structure. The structure may also be associated with a particular construction site, either directly, or indirectly through any of the entities located_site, zone_of_site, or building_complex. This entity is effectively an intersection entity providing the association between the entities coord_system, structure and site, resolving the many-to-many relationship between structure and site.

EXPRESS specification:

```
*)
ENTITY located_structure
SUBTYPE OF (located_item);
    descriptive_structure : structure;
    parent_site : site_select;
UNIQUE
    URL11 : SELF\located_item.location, descriptive_structure, parent_site;
WHERE
    WRL37 : SELF\located_item.location.coord_system_use =
        'Structure Coordinate System';
END_ENTITY;
(*
```

Attribute definitions:

descriptive_structure

Declares the instance of structure associated with the located_structure, for which the coordinate system is defined. There must be one (and only one) instance of structure referenced by each instance of located_structure. However, the same structure (a specific level entity) can be referenced by more than one instance of located_structure (an occurrence level entity). In this way, a structure may be located in many places, either on the same site, or on different sites.

parent_site

Declares the instance of site, located_site, zone_of_site, zone_of_building, or building_complex associated with the located_structure, on which the structure is located. The reference is made through the SELECT type site_select. Thus, an association may be made to any one of the members of the SELECT list. There must be one (and only one) instance of the entities of the SELECT list referenced by each instance of located_structure. However, there is no restriction in the INVERSE direction. For example, the same site can be referenced by more than one instance of located_structure. In this way, a site may contain many structures.

Formal propositions:

URL11

The combination of the instances of site, located_site, zone_of_site, zone_of_building or building_complex (referenced by the attribute parent_site), structure (referenced by the attribute descriptive_structure) and coord_system (referenced by the attribute location inherited from the SUPERTYPE located_item) shall be unique to this instance of

located_structure. (This prevents repetition of an association.) In other words, the structure should be associated with only one site.

WRL37

The attribute of the instance of coord_system referenced by the attribute location (inherited from the SUPERTYPE located_item) shall be assigned the value 'Structure Coordinate System'. (Note, the rule is case sensitive, the value must be exactly as shown.)

Notes:

New for CIS/2; addressed by STRUCTURE in CIS/1.

See diagram 37 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13 LPM/6 GROUPING

13.1 Grouping concepts and assumptions

The Grouping Subject Area of the LPM/6 schema allows certain entities in the LPM to be grouped for a particular purpose. It uses the `group`, `group_assignment` and `group_relationship` entities from STEP Part 41 (see ISO 10303-41: 1994 for the definitions of these entities). It further expands the grouping constructs of STEP by adding specialist SUBTYPES of `group_assignment` that allow the grouping of particular information. In line with the six main areas of the LPM, data is grouped as either, analysis data, design data, physical data, project definition data, structural data, or generic data.

A group may contain any number of sub-groups. A group may also be related to any number of other groups using the `group_relationship` entity.

It can be seen that the group is a very powerful and flexible construct. While LPM/6 does not have an entity named 'Bill of Materials', it does have constructs that allow applications to represent the equivalent information. That is, in LPM/6 a 'Bill of Materials' is merely a 'conveyor' of information; it is the information content that is of interest to the CIS, rather than the document itself. Thus, a group could represent a 'Bill of Materials' and the information content of the 'Bill of Materials' is assigned through the `group_assignment` entity and its SUBTYPES.

CAD drawings often contain 'layers' that are used to separate the different types of information captured by a drawing. In such a case, a group can represent a single layer of a drawing and the `group_name` would represent the 'layer number'. The `group_relationship` is then used to assign the layer to its parent drawing. For example, if 'Layer 1' of a drawing contains the building grid, then the `grid_of_building` instance is assigned to the group representing 'Layer 1' using the `group_of_structural_data` SUBTYPE of `group_assignment`.

A similar argument holds for 'schedules', or any other 'container of information'.

The STEP entity group is SUBTYPEd in LPM/6 as `media_file`. SUBTYPES of this entity can be used to represent an electronic CAD file or CNC file whose content is captured by SUBTYPES of the entity `media_content`.

13.2 Grouping type definitions

Note that these TYPES are also used in the Data Management subject area.

The following types have been modified in this subject area for the 2nd Edition.

- `select_generic_item`
- `select_structural_item`

The following types have been added to this subject area for the 2nd Edition:

- `drawing_class`

13.2.1 `drawing_class`

Type definition:

An indication of the class of information contained in the drawing represented by a `media_file_drawing`.

EXPRESS specification:

```
*)
TYPE drawing_class
= ENUMERATION OF
    (assembly_drawing,
     part_drawing,
     placement_drawing,
     undefined);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

13.2.2 select_analysis_item

Type definition:

Allows the selection of an analysis data item from the general classes of ‘analysis model’, ‘loading’, or ‘response’ data items; each of which are also SELECT data types.

EXPRESS specification:

```
*)
TYPE select_analysis_item
= SELECT
    (select_analysis_model_item,
     select_loading_item,
     select_response_item);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

13.2.3 select_analysis_model_item

Type definition:

A subclass of the analysis data that allows the selection of one of a number of entities that fall under the classification of ‘analysis model data’.

EXPRESS specification:

```
*)
TYPE select_analysis_model_item
= SELECT
    (analysis_method,
     boundary_condition,
     element_eccentricity,
     element_node_connectivity,
     analysis_model,
     analysis_model_mapping,
     analysis_model_relationship,
     element,
```



```

    element_mapping,
    node,
    node_dependency,
    release);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

13.2.4 select_design_item

Type definition:

Allows the selection of one of a number of entities that fall under the classification of ‘design data’.

EXPRESS specification:

```

*)
TYPE select_design_item
= SELECT
    (assembly_design,
    assembly_map,
    assembly_relationship,
    design_criterion,
    design_joint_system,
    design_part,
    design_result ,
    effective_buckling_length,
    functional_role,
    resistance,
    restraint);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

13.2.5 select_generic_item

Type definition:

Allows the selection of one of the entities that fall under the classification of ‘generic data’. These are the entities that have been taken from the STEP Generic Resources. This type is selected through the SELECT type select_data_item, and is used when data management conformant applications need to populate the entity managed_data_item to SELECT which item of generic data (taken from STEP) is be given meta data. On a more basic level, it is also used with the group_of_generic_data entity to collate a group of generic data.

EXPRESS specification:

```

*)
TYPE select_generic_item
= SELECT

```

(action,
action_directive,
action_method,
address,
approval,
approval_status,
box_domain,
certification,
certification_type,
contract,
contract_type,
coordinated_universal_time_offset,
date,
date_and_time,
derived_unit,
derived_unit_element,
description_attribute,
dimensional_exponents,
document,
document_type,
document_relationship,
document_representation_type,
document_usage_constraint,
founded_item,
functionally_defined_transformation,
group,
group_assignment,
group_relationship,
id_attribute,
item_defined_transformation,
local_time,
measure_qualification,
measure_with_unit,
name_attribute,
named_unit,
object_role,
organization,
organization_relationship,
person,
person_and_organization,
person_and_organization_role,
representation,
representation_context,
representation_item,
representation_map,
representation_relationship,
role_association,

```

        surface_patch,
        value_qualifier,
        versioned_action_request);
END_TYPE;
(*)

```

Notes:

New for CIS/2. Extended for 2nd Edition

13.2.6 select_loading_item**Type definition:**

A subclass of the analysis data that allows the selection of one of a number of entities that fall under the classification of 'loading data'.

EXPRESS specification:

```

*)
TYPE select_loading_item
= SELECT (
    applied_load,
    load_case,
    load,
    loaded_product,
    loading_combination,
    load_combination_occurrence,
    physical_action);
END_TYPE;
(*)

```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

13.2.7 select_physical_item**Type definition:**

Allows the selection of one of the entities that fall under the classification of 'physical model data'. As the located_item is a SUPERTYPE of most of the physical model data, this SELECT type contains only two members - located_item and located_part_joint. (The located_item entity is the SUPERTYPE of several other aspects of the physical model, such as located_part and located_assembly. Thus, these aspects may be also be referenced via this type.)

EXPRESS specification:

```

*)
TYPE select_physical_item
= SELECT
    (located_item, located_part_joint);
END_TYPE;
(*)

```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

13.2.8 select_project_definition_item

Type definition:

Allows the selection of one of the entities that fall under the classification of ‘project definition data’.

EXPRESS specification:

```
*)
TYPE select_project_definition_item
= SELECT
    (assembly,
    building,
    building_complex,
    currency_measure_with_unit,
    project,
    project_plan,
    project_plan_item,
    project_plan_item_relationship,
    project_organization,
    site,
    structure);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

13.2.9 select_response_item

Type definition:

A subclass of the analysis data that allows the selection of one of a number of entities that fall under the classification of ‘structural response data’.

EXPRESS specification:

```
*)
TYPE select_response_item
= SELECT (
    analysis_result,
    analysis_results_set,
    design_result,
    reaction);
END_TYPE;
(*
```

13.2.10 select_structural_item

Type definition:

Allows the selection of one of the entities that fall under the classification of ‘general structural data’. This type is selected through the select_data_item type and is used when data management conformant applications need to populate the entity managed_data_item to SELECT which item of structural data is associated with the meta data. On a more

basic level, this type is also used with the entity `group_of_structural_data` to collate a group of structural data.

EXPRESS specification:

*)

```

TYPE select_structural_item
= SELECT
    (coord_system,
    grid,
    grid_intersection,
    grid_offset,
    geographical_location,
    item_cost_code,
    item_cost_code_assigned,
    item_property,
    item_property_assigned,
    item_reference,
    item_reference_assigned,
    item_ref_source,
    item_ref_source_documented,
    section_properties,
    setting_out_point,
    structural_frame_item,
    structural_frame_item_approved,
    structural_frame_item_certified,
    structural_frame_item_documented,
    structural_frame_item_priced,
    structural_frame_item_relationship,
    zone);
END_TYPE;
(*)

```

Notes:

New for CIS/2. Modified for 2nd Edition.

13.3 Grouping entity definitions

The following entities have been modified for the 2nd Edition:

- `media_file`
- `step_file`

The following entities have been added for the 2nd Edition:

- `media_content_drawing`
- `media_file_cnc`
- `media_file_drawing`
- `project_data_group`

13.3.1 group_assignment_actioned

Entity definition:

A type of group_assignment that is associated with an action. According to STEP Part 41, an action is “the effort made to realize a specific result”. In other words, it is something that is done to the product data.

EXPRESS specification:

```
*)
ENTITY group_assignment_actioned
SUBTYPE OF (group_assignment);
    assigned_action : action;
UNIQUE
    URG3 : SELF\group_assignment.assigned_group, assigned_action;
END_ENTITY;
(*
```

Attribute definitions:

assigned_action

Declares the instance of action associated with the group_assignment_actioned. There must be one (and only one) instance of action associated with each instance of group_assignment_actioned. This instance specifies what has been done (or needs to be done) to the associated data (e.g. “this data was deleted by Application X”).

Formal propositions:

URG3

The combination of the instances of group (referenced by the attribute assigned_group inherited from the SUPERTYPE group_assignment) and action (referenced by the attribute assigned_action) shall be unique to this instance of group_assignment_actioned. This prevents repetition of assignments.

Informal propositions:

Although this entity could be instanced on its own, it would be of limited semantic value. This entity would normally be instanced with one of its sibling SUBTYPEs (e.g. group_of_physical_data, or managed_data_group).

Notes

New for CIS/2.

The entities group_assignment and action (and its SUBTYPEs) are defined in STEP Part 41.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.2 group_assignment_approved

Entity definition:

A type of group_assignment that is associated with an approval. According to STEP Part 41, an approval is “a confirmation of the quality of the product data”. This provides the level of approval of the data; e.g. “approved for construction”, or “preliminary”.

EXPRESS specification:

```

*)
ENTITY group_assignment_approved
SUBTYPE OF (group_assignment);
    assigned_approval : approval;
UNIQUE
    URG4 : SELF\group_assignment.assigned_group, assigned_approval;
END_ENTITY;
(*

```

Attribute definitions:*assigned_approval*

Declares the instance of approval associated with the group_assignment_approved. There must be one (and only one) instance of approval associated with each instance of group_assignment_approved. This instance specifies what the level of the approval of the associated data.

Formal propositions:*URG4*

The combination of the instances of group (referenced by the attribute assigned_group inherited from the SUPERTYPE group_assignment) and approval (referenced by the attribute assigned_approval) shall be unique to this instance of group_assignment_approved. This prevents repetition of assignments.

Informal propositions:

Although this entity could be instanced on its own, it would be of limited semantic value. This entity would normally be instanced with one of its sibling SUBTYPEs (e.g. group_of_physical_data, or managed_data_group).

Notes

New for CIS/2.

The entities group_assignment and approval are defined in STEP Part 41.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.3 group_of_analysis_data**Entity definition:**

A type of group_assignment that assigns a set of two or more instances of entities representing analysis data to a particular group.

EXPRESS specification:

```

*)
ENTITY group_of_analysis_data
SUBTYPE OF (group_assignment);
    items : SET [2:?] OF select_analysis_item;
END_ENTITY;
(*

```

Attribute definitions:*items*

Declares the set of two or more separate instances of entities representing analysis data that are associated with this `group_of_analysis_data`, and thus assigned to the group. The references are made through the SELECT type `select_analysis_item`. Any member of the SELECT list may be referenced. Any SUBTYPE of any of the members of the SELECT list may be referenced. The references made must all be to different instances. That is, instances may not be repeated in the set. However, the entity type may be repeated. The entity types referenced do not have to be all of the same type. There must be two or more instances of entities that fall under the classification of ‘analysis data’ referenced by each instance of `group_of_analysis_data`.

Formal propositions:

The SET data type prohibits instance repetition in the members of the aggregation.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.4 group_of_design_data**Entity definition:**

A type of `group_assignment` that assigns a set of two or more instances of entities representing design data to a particular group.

EXPRESS specification:

```
*)
ENTITY group_of_design_data
  SUBTYPE OF (group_assignment);
    items : SET [2:?] OF select_design_item;
END_ENTITY;
(*
```

Attribute definitions:*items*

Declares the set of two or more separate instances of entities representing design data that are associated with this `group_of_design_data`, and thus assigned to the group. The references are made through the SELECT type `select_design_item`. Any member of the SELECT list may be referenced. Any SUBTYPE of any of the members of the SELECT list may be referenced. The references made must all be to different instances. That is, instances may not be repeated in the set. However, the entity type may be repeated. The entity types referenced do not have to be all of the same type. There must be two or more instances of entities that fall under the classification of ‘design data’ referenced by each instance of `group_of_design_data`.

Formal propositions:

The SET data type prohibits instance repetition in the members of the aggregation.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity group_assignment is defined in STEP Part 41.

See diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.5 group_of_generic_data**Entity definition:**

A type of group_assignment that assigns a set of one or more instances of entities that represent generic data to a particular group. This collates the type of information that is represented by entities in the STEP Generic Resources; e.g. point, line, curve, etc. In this way, a 'drawing' can 'contain' a set of points and lines (or any other explicit geometry construct.)

EXPRESS specification:

```
*)
ENTITY group_of_generic_data
  SUBTYPE OF (group_assignment);
    items : SET [1:?] OF select_generic_item;
END_ENTITY;
(*
```

Attribute definitions:*items*

Declares the set of one or more separate instances of entities representing generic data that are associated with this group_of_generic_data, and thus assigned to the group. The references are made through the SELECT type select_generic_item. Any member of the SELECT list may be referenced. Any SUBTYPE of any of the members of the SELECT list may be referenced. The references made must all be to different instances. That is, instances may not be repeated in the set. However, the entity type may be repeated. The entity types referenced do not have to be all of the same type. There must be one or more instances of entities that fall under the classification of 'generic data' referenced by each instance of group_of_generic_data. Note, the references in the set must be to entities that are classed as 'generic data' not to any entity.

Formal propositions:

The SET data type prohibits instance repetition in the members of the aggregation.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity group is defined in STEP Part 41. All of the entities that may be referenced by this entity, and hence collected in a group of generic data, are defined in STEP Parts 41, 42, 43, 44, and 45.

See diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.6 group_of_physical_data**Entity definition:**

A type of group_assignment that assigns a set of two or more instances of entities representing physical model data to a particular group.

EXPRESS specification:

```
*)
ENTITY group_of_physical_data
  SUBTYPE OF (group_assignment);
    items : SET [2:?] OF select_physical_item;
END_ENTITY;
(*
```

Attribute definitions:*items*

Declares the set of two or more separate instances of located_item or located_part_joint that are associated with this group_of_physical_data, and thus assigned to the group. The references are made through the SELECT type select_physical_item. Either member of the SELECT list may be referenced. Any SUBTYPE of the entity located_item may be referenced. The references made must all be to different instances. That is, instances may not be repeated in the set. However, the entity type may be repeated. The entity types referenced do not have to be all of the same type. There must be two or more instances of entities that fall under the classification of 'physical model data' referenced by each instance of group_of_physical_data.

Formal propositions:

The SET data type prohibits instance repetition in the members of the aggregation.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity group_assignment is defined in STEP Part 41.

See diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.7 group_of_project_definition_data

Entity definition:

A type of group_assignment that assigns a set of two or more instances of entities representing project definition data to a particular group.

EXPRESS specification:

*)

ENTITY group_of_project_definition_data

SUBTYPE OF (group_assignment);

items : SET [2:?] OF select_project_definition_item;

END_ENTITY;

(*

Attribute definitions:

items

Declares the set of two or more separate instances of entities representing project definition data that are associated with this group_of_project_definition_data, and thus assigned to the group. The references are made through the SELECT type select_project_definition_item. Any member of the SELECT list may be referenced. Any SUBTYPE of any of the members of the SELECT list may be referenced. The references made must all be to different instances. That is, instances may not be repeated in the set. However, the entity type may be repeated. The entity types referenced do not have to be all of the same type. There must be two or more instances of entities that fall under the classification of 'project definition data' referenced by each instance of group_of_project_definition_data.

Formal propositions:

The SET data type prohibits instance repetition in the members of the aggregation.

Informal propositions:

This is an interpreted construct; i.e. a Generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity group_assignment is defined in STEP Part 41.

See diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.8 group_of_structural_data

Entity definition:

A type of group_assignment that assigns a set of one or more instances of entities representing structural data (such as grids or setting out points) to a particular group. For example, if a building grid is contained in a layer of a drawing, an instance of grid_of_building is assigned to an instance of group that represents that layer of the drawing.

EXPRESS specification:

```

*)
ENTITY group_of_structural_data
  SUBTYPE OF (group_assignment);
    items : SET [1:?] OF select_structural_item;
END_ENTITY;
(*)

```

Attribute definitions:*items*

Declares the set of one or more separate instances of entities representing structural data that are associated with this `group_of_structural_data`, and thus assigned to the group. The references are made through the SELECT type `select_structural_item`. Any member of the SELECT list may be referenced. Any SUBTYPE of any of the members of the SELECT list may be referenced. The references made must all be to different instances. That is, instances may not be repeated in the set. However, the entity type may be repeated. The entity types referenced do not have to be all of the same type. There must be two or more instances of entities that fall under the classification of ‘structural data’ referenced by each instance of `group_of_structural_data`.

Formal propositions:

The SET data type prohibits instance repetition in the members of the aggregation.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity `group_assignment` is defined in STEP Part 41.

See diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.9 group_usage**Entity definition:**

A type of `group_relationship` that checks the uniqueness of the relationship declared in the SUPERTYPE. The SUPERTYPE `group_relationship` creates a relationship between two instance of group, giving the relationship a name and a description. This entity (`group_usage`) provides a unique identifier for the relationship, and ensures that the group does not relate to itself, either directly or indirectly.

This entity is used when the dependency of groups is at issue. For example, a Bill of Quantities (group #1) may be related to (and be defined by) a group of assemblies (group #2). The group of assemblies (group #2) may be related to (and be defined by) a group of parts (group #3). This entity checks that the relationships are not cyclic (self-defining) such that the group of parts (group #3) is related to (and defined by) the Bill of Quantities (group #1).

EXPRESS specification:

```

*)
ENTITY group_usage
SUBTYPE OF (group_relationship);
UNIQUE
    URG5 : SELF\group_relationship.name,
        SELF\group_relationship.relate_group,
        SELF\group_relationship.related_group;
WHERE
    WRG6 : acyclic_group_relationship(SELF, [SELF\group_relationship.related_group],
    'STRUCTURAL_FRAME_SCHEMA.
    GROUP_RELATIONSHIP.RELATED_GROUP');
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `group_relationship`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `group_relationship` to ensure that only unique relationships are formed between groups.

Formal propositions:**URG5**

The combination of the values of the attributes `name`, `relate_group`, and `related_group` (inherited from the SUPERTYPE `group_relationship`) shall be unique to this `group_usage`. That is, the inherited `name`, `relate_group`, and `related_group` uniquely identify an instance of `group_usage`. This prevents the association between two groups being repeated with the same name. This does not, on the other hand, prevent two groups from being related twice if the name of the relationship is different for both instances.

WRG6

The graph structure of the group 'nodes' and `group_relationship` 'links' shall be acyclic. Each group shall not be a descendant of itself in the graph structure. In other words, a group shall not be related to itself, either directly, or indirectly through other relationships.

This WHERE rule uses the STEP Part 41 function `acyclic_group_relationship` to determine whether or not the given groups have been self-defined by the associations made in the `group_relationship` inherited by the instance of `group_usage`. The function examines the instance of `group_usage` and the instance of group referenced by the attribute `related_group`. It then calculates the 'nodes' and 'links' of the graph that has this group (as a node) and this `group_relationship` (as a link). The function returns a value of TRUE if the graph is acyclic (i.e. the group referenced by the attribute `related_group` is not used again as a 'related_group' in the same graph). Otherwise, the function returns a value of FALSE (i.e. the group is self-defining).

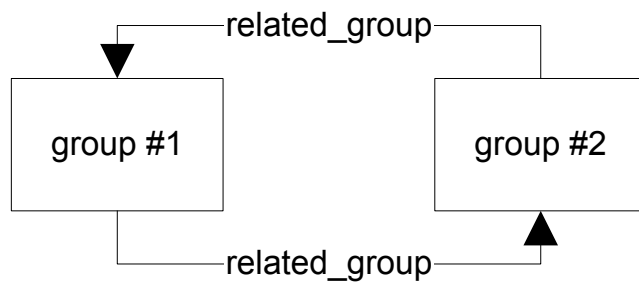


Figure 13.1 *An example of a cyclic group relationship*

Notes

New for CIS/2.

This entity is based upon the entity `product_definition_usage` defined in STEP Part 44.

The entities `group` and `group_relationship`, and the function `acyclic_group_relationship` are all defined in STEP Part 41.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

13.3.10 media_content

Entity definition:

A type of `group_assignment` that collates items of data that belong to a common `media_file`. The `media_file` is a SUBTYPE of `group`.

EXPRESS specification:

```

*)
ENTITY media_content
  SUBTYPE OF (group_assignment);
  WHERE
    WRM26 : 'STRUCTURAL_FRAME_SCHEMA.MEDIA_FILE' IN TYPE OF
      (SELF\group_assignment.assigned_group);
END_ENTITY;
(*
  
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `group_assignment`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `group_assignment`, such that it belongs to a group of the type `media_file`.

Formal propositions:

WRM26

The instance of `group` referenced by the attribute `assigned_group` (inherited from the SUPERTYPE `group_assignment`) shall be of the type `media_file`. In other words, `media_content` describes the content of a `media_file`. (See also definition of `media_file`.)

Informal propositions:

This entity would normally be instantiated with one of its sibling SUBTYPEs such as `managed_data_group`.

Notes

New for CIS/2.

The entity group_assignment is defined in STEP Part 41.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition

13.3.11 media_content_drawing**Entity definition:**

A type of media_content restricted to being part of a drawing file.

EXPRESS specification:

*)

ENTITY media_content_drawing

SUBTYPE OF (media_content);

WHERE

WRM42 : 'STRUCTURAL_FRAME_SCHEMA.MEDIA_FILE_DRAWING' IN TYPE OF
(SELF\group_assignment.assigned_group);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own; it merely constrains the use of its SUPERTYPE.

Formal propositions:

WRM42

The content shall be assigned to a drawing file.

Notes

New for CIS/2 2nd Edition

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

13.3.12 media_file**Entity definition:**

A type of group that describes an electronic file. The file could contain any type of electronic media (e.g. a document, drawing, sound, video, animation, virtual reality walkthroughs, or a web page). This entity allows data to be referenced to external documents or drawings held in electronic form. It also allows a reference to a page on the World Wide Web.

EXPRESS specification:

*)

ENTITY media_file

SUPERTYPE OF (ONEOF(step_file, media_file_drawing, media_file_cnc))

SUBTYPE OF (group);

file_source : label;

file_format : label;

file_date : date_and_time;

media_type : label;

author : LIST [1:?] OF person_and_organization;
 owner : LIST [0:?] OF person_and_organization;
 END_ENTITY;
 (*

Attribute definitions:

file_source

A short text reference for the media_file. This would include the file name with the full path for the electronic file (e.g. C:\temp\mydrawing.dxf). For a web page, this attribute would provide the URL (e.g. <http://www.cis2.org/index.html>).

It should be noted that the backslash (\) is a special character in STEP Part 21. It needs to be encoded as a double backslash (\\). Thus, the file name above would be encoded as (e.g. C:\\temp\\mydrawing.dxf).

file_format

A short computer-readable reference for the format of the electronic file; e.g. 'dxf', 'html', 'jpg', 'gif'. A list of file extensions for defined MIME types may be found on the Duke University web site at <http://www.duke.edu/websrv/file-extensions.html>. Note; the leading dot (.) is dropped when populating this attribute.

file_date

Declares the instance of date_and_time associated with this media_file to provide the date and time that the electronic file was last modified.

media_type

A short human-readable reference for the type of media contained in the electronic file; e.g. drawing, photograph, video.

author

Declares the list of one or more instances of person_and_organization associated with the media file. These instances provide the list of people and organizations that are deemed to have authored the content of the electronic file.

owner

Declares the list of zero, one or more instances of person_and_organization that may be associated with the media file. These instances provide the list of people and organizations that are deemed to own (and hold copyright of) the content of the electronic file. These may or may not correspond to the list of authors.

Notes

New for CIS/2.

The entities date_and_time and person_and_organization are defined in STEP Part 41.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – SUBTYPEs added.

13.3.13 media_file_cnc

Entity definition:

A type of media_file used to represent a CNC file (containing the data for Computer Numerically Controlled machinery).

EXPRESS specification:

```

*)
ENTITY media_file_cnc
SUBTYPE OF (media_file);
    cnc_data_format : label;
DERIVE
    cnc_file_title : label := SELF\group.group_name;
    created_by : person := SELF\media_file.author[1].the_person;
    detail_company : organization := SELF\media_file.author[1].the_organization;
    creation_date : date_and_time := SELF\media_file.file_date;
    cnc_filename : label := SELF\media_file.file_source;
END_ENTITY;
(*

```

Attribute definitions:*cnc_data_format*

A short alphanumerical reference indicating the format of the data contained in the CNC file; e.g. the standard to which the CNC data has been written.

cnc_file_title

Derives the title of the CNC file based on the attribute group_name inherited from the SUPERTYPE group.

created_by

Derives the instance of person designated as the creator of the CNC file based on the attribute author inherited from the SUPERTYPE media_file. It is implied that this person works for the organization referenced by the attribute detail_company.

detail_company

Derives the instance of organization designated as the creator of the CNC file based on the attribute author inherited from the SUPERTYPE media_file. It is implied that the person referenced by the attribute created_by is employed by the detail_company.

creation_date

Derives the instance of date_and_time designated as the creation date of the CNC file based on the attribute file_date inherited from the SUPERTYPE media_file.

cnc_filename

Derives the short text reference for the CNC file based on the attribute file_source inherited from the SUPERTYPE media_file.

Notes

New for CIS/2 2nd Edition

See Diagram 75 of the EXPRESS-G diagrams in Appendix B.

13.3.14 media_file_drawing**Entity definition:**

A type of media_file assigned a set of parameter used to represent an electronic drawing file.

EXPRESS specification:

*)

ENTITY media_file_drawing

SUBTYPE OF (media_file);

drawing_number : OPTIONAL label;

drawing_type : drawing_class;

drawing_size : OPTIONAL label;

current_revision_mark : OPTIONAL label;

current_revision_by : OPTIONAL person_and_organization;

current_revision_date : OPTIONAL date_and_time;

current_revision_note : OPTIONAL text;

DERIVE

drawing_title : label := SELF\group.group_name;

drawn_by : person := SELF\media_file.author[1].the_person;

detail_company : organization := SELF\media_file.author[1].the_organization;

creation_date : date_and_time := SELF\media_file.file_date;

drawing_filename : label := SELF\media_file.file_source;

END_ENTITY;

(*

Attribute definitions:***drawing_number***

An optional short text reference that may be used to identify the drawing file.

drawing_type

An indication of the class of information contained in the drawing represented by the media_file_drawing.

drawing_size

An indication of the size of the (paper) drawing represented by the media_file_drawing; e.g. 'A1', 'A0'.

current_revision_mark

An optional short text reference that may be used to indicate the current status of the drawing file; e.g. 'A', 'B'.

current_revision_by

Declares the instance of person_and_organization that may be used to denote the person and organisation that created the current revision of the drawing file.

current_revision_date

Declares the instance of date_and_time that may be used to denote the date and time that the current revision of the drawing file was created.

current_revision_note

A text description applicable to the current revision of the drawing file.

drawing_title

Derives the title of the drawing file based on the attribute group_name inherited from the SUPERTYPE group.

drawn_by

Derives the instance of person designated as the creator of the drawing file based on the attribute author inherited from the SUPERTYPE media_file. It is implied that this person works for the organization referenced by the attribute detail_company.

detail_company

Derives the instance of organization designated as the creator of the drawing file based on the attribute author inherited from the SUPERTYPE media_file. It is implied that the person referenced by the attribute drawn_by is employed by the detail_company.

creation_date

Derives the instance of date_and_time designated as the creation date of the drawing file based on the attribute file_date inherited from the SUPERTYPE media_file.

drawing_filename

Derives the short text reference for the drawing file based on the attribute file_source inherited from the SUPERTYPE media_file. (Note: this would include the full filename and path with the file extension.)

Notes

New for CIS/2 2nd Edition

See Diagram 75 of the EXPRESS-G diagrams in Appendix B.

13.3.15 project_data_group**Entity definition:**

A type of group_assignment where the data in the group is deemed to belong to a project. A mechanism for grouping together information from the same project.

EXPRESS specification:

```
*)
ENTITY project_data_group
  SUBTYPE OF (group_assignment);
    parent_project : project;
END_ENTITY;
(*
```

Attribute definitions:*parent_project*

Declares the instance of project to which the group of data belongs.

Informal propositions:

Although this entity could be instanced on its own, it would be of limited semantic value. This entity would normally be instanced with one of its sibling SUBTYPES (e.g. group_of_physical_data, or managed_data_group).

Notes

New for CIS/2 2nd Edition

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

13.3.16 step_file

Entity definition:

A type of media_file that represents a STEP Part 21 file. This entity allows a reference to an external STEP Part 21 file. Thus, a STEP Part 21 file may contain several references to other STEP Part 21 files. When used with the data management constructs, these instances represent the origin of the current STEP Part 21 file (and the data contained therein) and trace its history through various versions.

EXPRESS specification:

```

*)
ENTITY step_file
SUBTYPE OF (media_file);
INVERSE
    selected_content : SET [1:?] OF group_assignment FOR assigned_group;
WHERE
    WRS28 : (SELF\media_file.file_format = 'STP') OR (SELF\media_file.file_format = 'stp');
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE media_file. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity media_file.

Formal propositions:

selected_content

There shall be at least one instance of group_assignment associated with this step_file. In other words, the STEP file must have some content. This constrains the relationship between group_assignment and group created by the attribute assigned_group in the INVERSE direction.

WRS28

The attribute file_format (inherited from the SUPERTYPE media_file) shall be assigned the value 'STP' or 'stp'. In other words, a STEP Part 21 file should have the extension STP or stp.

Notes

New for CIS/2.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – WHERE rule changed.

14 LPM/6 GEOMETRY

14.1 Geometry concepts and assumptions

14.1.1 Explicit and implicit geometry

CIS/1 contained no provision for the explicit representation of geometry. All the geometry in CIS/1 was defined implicitly using sets of parameters such as depth, width, length, etc. In order to accommodate CAD systems that specialize in the explicit representation of an object's shape, CIS/2 now incorporates constructs for explicit geometry. As many of these constructs are already defined in STEP Part 42, CIS/2 incorporates STEP Part 42 within LPM/6. The STEP constructs for explicit geometry are used as resources in LPM/6. That is, the STEP entity `shape_representation` is used as the data type of the attribute for an entity wherever the shape of an object requires the use of explicit geometry.

Note: the 2nd Edition of CIS/2 uses the 2nd Edition of the STEP generic resources; many of the geometry constructs have been corrected.

SUPERTYPE-SUBTYPE hierarchies are used in LPM/6 to categorize objects by their geometry. In order to denote the use of implicit or explicit geometry, LPM/6 adopts the convention whereby objects are categorized as either simple or complex. For example:

ENTITY object

 SUPERTYPE OF (ONEOF (object_simple, object_complex));

END_ENTITY;

ENTITY object_simple

 SUBTYPE OF (object);

 length : length_measure;

 depth : length_measure;

 width : length_measure;

END_ENTITY;

ENTITY object_complex

 SUBTYPE OF (object);

 shape : shape_representation; -- or some other explicit geometry construct

END_ENTITY;

14.1.2 Understanding the explicit geometry constructs

The EXPRESS constructs for explicit geometry are defined in STEP Part 42. However, those definitions are rather esoteric. The following definitions (adapted from *CADCAM Glossary*^[4] and other sources^[14, 36, 38]) are intended to add a little clarity to those definitions and aid understanding of the STEP constructs.

Analytics

The generic name for geometric shapes such as lines, planes, spheres, cylinders, etc, that can be modelled mathematically. The analytic representation is only applicable for lines and conic sections, and planar, quadratic, and toroidal surfaces. They cannot represent free-form surfaces – i.e. those surfaces that have no regular geometric properties.

Bézier curves and surfaces

Bézier curves and surfaces were developed by Pierre Bézier for modelling the surfaces of automobiles. Bézier curve blending functions have some important characteristics, key to the determination of the shape and form of the curve. The shape of the curve is predictable and related to the location of the control points. The curve passes through the start and end control points of the control polygon. The curve is tangential to the corresponding edge of the control polygon at the end points. Bézier curves are independent of the coordinate system used to measure the location of the control points. However, the meaning of shape is lost when they are manipulated and they cannot represent analytics exactly. Numerical accuracy can be difficult to achieve with Bézier curves. They generate large amounts of data, since the algorithms are complex. Bézier curves do not provide for local curve control. Moving any control point will change the shape of every part of the curve. The versatility of a Bézier curve is governed by the number of control points used. Too many control points will generate high order polynomial equations which are complex to solve and which become difficult to use because of the lack of localized control.

B-spline curves and surfaces

The key difference between Bézier curves and B-spline curves, lies in the formulation of the blending functions. The control points affect curve shape in a natural way and do it locally on the curve. B-splines also reduce the need to piece together many curves to define a shape. Control points can be added at will, without increasing the degree of the curve which would make the curve more difficult to control. Curves are modified by moving the control points.

B-splines can represent free-form curves and surfaces. They are relatively robust, since they use a general, well-behaved algorithm. B-spline curves can be easily modified locally – the location of the curve depends only on a few neighbouring control points.

As with Bézier, the meaning of the shape is lost when the curve is manipulated. B-splines cannot represent analytics exactly. Numerical accuracy can be difficult to achieve.

Blending functions

The mathematical equations that are attached to each of the control points on the curve or surface that, when summed, define the shape of the curve and, when changed, modify the shape of curve either locally or globally depending upon the curve type.

Boolean operations

Boolean operations are used to construct solid geometry from primitives. They are usually hidden from the user – all the user sees is the end result. Booleans take apart the solid model data structure and rebuild it in a new form. Two primary objects are required to create a resultant new object. The resultant object depends upon the Boolean operation applied; for example:

- Add – the resultant object occupies all the space of the two initial objects (Union).
- Subtract – the resultant object occupies all the space of the first object except for the space inside the second object (Difference).
- Intersect – the resultant object occupies only that space common to both initial objects (Intersection).

Boundary representation (B-rep)

B-rep models represent a part or solid by its external surfaces, or boundaries (the outside of the solid which distinguishes the exterior from the interior). Boolean operations can be performed on B-rep models. Only the names and definitions of the boundaries and their parameters (relative instance, position, orientation, etc) are stored in a boundary representation form.

The boundary representation is a model of the faces of a solid. It keeps track of the description of the faces by distinguishing between geometry and topology. Geometry defines the size and shape of the model and its edges, while topology defines the connectivity of the different elements of geometry of the model. (Connectivity is the blueprint that defines the way that edges join surfaces and faces in model topology.)

Constructive solid geometry (CSG)

CSG models store a solid as a sequence of combinations to primitives (primary building blocks, such as cubes, cylinders, cones, etc), which can be used to produce more complex solids. The sequence of modifications is stored as a 'CSG tree'. The model cannot be fully realized without 'evaluating' the CSG tree each time it is displayed.

CSG representation is a history or family tree structure of Boolean (addition or subtraction) operations of the primitives to make a solid.

Degree of a curve

Curves can have varying degrees; i.e. the highest exponent in the polynomial equation, which can be, for example:

1. for a straight line
2. for a quadratic curve
3. for a cubic curve
4. for a quartic curve.

The order of a curve is one greater than the degree of the polynomial equation required to build it. The higher the order of the curve, the farther it gets from its control points and the smoother it is.

Loops

Particular instances of edges that bind certain kinds of topologies. For example, the edge that binds the top or bottom surface of a closed cylinder to the circumferential surface is a loop. (See also Topology.)

Manifold and non-manifold solids

A manifold solid is one that contains a continuum of volume, whereas a non-manifold solid is one that does not contain a continuum of volume. (Non-manifold solids cannot be physically manufactured; they are generated for the convenience of modelling the geometry and topology of a solid.)

Parameterization

The mapping of a curve in 3D space (x, y, z) into a 1D closed interval between the limits zero and one. This enables any point on the curve to be determined using its parametric value, thus simplifying the representation of the curve and giving it what is called one degree of freedom.

For surfaces, parameterization is the mapping in 3D space (x, y, z), into two 1D closed interval between the limits zero and one. This enables any point on the surface to be

defined by using two parametric values (often represented as u and v values), thus simplifying the representation of the surface giving it two degrees of freedom

Parameterization provides a better technique for controlling the shape of curves and surfaces than do non-parametric forms. Parameterization simplifies mathematical calculations and geometric handling such as rotations and translations.

Primitives

Primary building blocks, such as cubes, cylinders, cones, etc which can be used to produce more complex solids. (See definition of constructive solid geometry.)

Topology

Solid bodies are a combination of both topology and geometry. Geometry is the 'raw material' used to build solid models while topology is the way that raw material is put together.

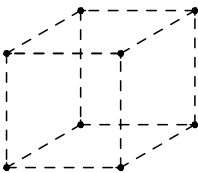
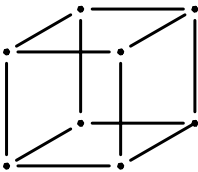
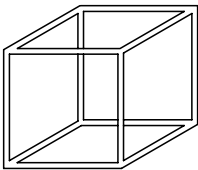
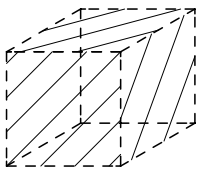
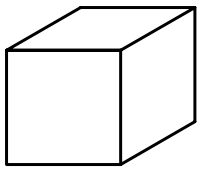
	<u>Topology</u>	<u>Geometry</u>
	8 vertices	points
	12 edges	lines
	6 loops	curves
	6 faces	surfaces
	1 shell	solid

Figure 14.1 *Relating topology to geometry*

Shells, faces, loops, edges and vertices are the building blocks of topology – they bind or limit infinite geometry. Geometry is attached to these topological items.

- **Shells:** A Shell is a collection of consistently oriented faces forming the bound of a single connected, closed volume.
- **Faces:** A Face is a single plane or surface that is bounded externally and also sometimes internally.

- **Loops:** A Loop is a particular instance of edge that bind certain kinds of topologies.
- **Edges:** An Edge is the individual boundary of loops that form a face.
- **Vertices:** A Vertex is the end point of an edge, or a junction between edges (or collapsed edges as in the case of the apex of a cone).

Topology is sometimes described as ‘geometry done on a rubber sheet’; this sheet can be pulled or stretched into different shapes. Topological properties are unaltered by distortions of this kind.

Wireframe modelling

Wireframe geometry is simple geometry constructed from points, lines, arcs and curves.

2D wireframe models can be ambiguous: human interpretation is required to understand them fully. Spatial relationships are hard to visualize using 2D wireframe models.

3D wireframe models use extensions to the basic 2D wireframe primitives. However, because no surfaces are represented in a 3D wireframe model, the surfaces covering the wireframe model are inferred. Without surfaces, it is difficult (if not impossible) to determine what is solid material and what is empty space. Mass and volume properties cannot be automatically calculated.

14.2 Geometry type definitions

No types have been added to this subject area for the 2nd Edition.

The following types have been modified in this subject area for the 2nd Edition:

- cardinal_point_ref

14.2.1 cardinal_point_ref

Type definition:

The integer value of a reference point on the section profile. For a generic section, typical values are shown in Figure 14.2.

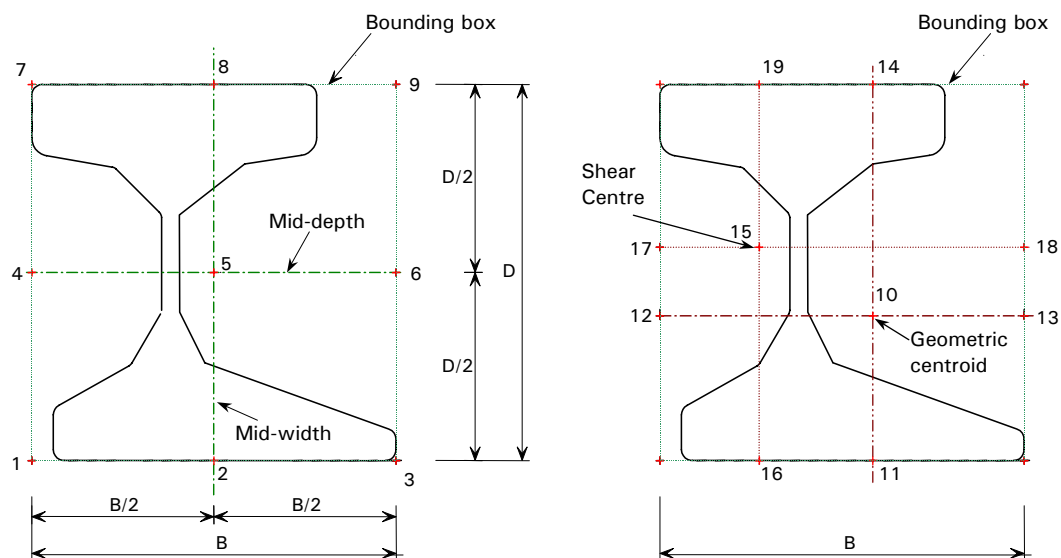


Figure 14.2 Defining cardinal points 1-19 on a generic section profile

The first 9 cardinal points are defined by the bounding box together with lines drawn at mid-height and mid-width of the section. The bounding box is defined by the extreme

corners of the section such that 2 sides of the bounding box lie parallel to the yy and 2 sides lie parallel to the zz-axes of the section. Although the figures show the flanges of the section lying on the bounding box, this will not be the case for irregular section where the extreme top and bottom surfaces of the flanges are not horizontal. In this case, points 1, 2, 3 and 7, 8 and 9 would lie on the true bounding box.

Cardinal points 10-14 are based on the geometric centroid of the section together with lines projected from the geometric centroid to the bounding box.

Cardinal points 15-19 are based on the shear centre of the section together with lines projected from the shear centre to the bounding box.

The first 19 cardinal points can be defined for any section, including fabricated or compound sections. Obviously, for symmetrical sections, several of the points will coincide.

The first 19 cardinal points are numbered as follows:

1. bottom left
2. bottom centre
3. bottom right
4. mid-depth left
5. mid-depth centre
6. mid-depth right
7. top left
8. top centre
9. top right
10. geometric centroid
11. bottom in line with the geometric centroid
12. left in line with the geometric centroid
13. right in line with the geometric centroid
14. top in line with the geometric centroid
15. shear centre
16. bottom in line with the shear centre
17. left in line with the shear centre
18. right in line with the shear centre
19. top in line with the shear centre

With reference to the coordinate system of a part or element whose cross-section is defined by an instance of `section_profile`, the left hand side is where y has its maximum value and the right hand side is where y has its minimum value (looking at the part or element in the x-positive direction). Similarly, the top is where z has its maximum value, while the bottom is where z has its minimum value. These definitions of left, right, top and bottom hold for both ends of the part or element. See Figure 9.1

Cardinal points 21-39 are based on the bounding box defined by the extreme corners and the mid-points of the flanges, and will only exist for flanged sections. (See Figure 14.3.) These points are numbered as follows:

20. *undefined*

- 21. bottom left of bounding box of bottom flange
- 22. bottom centre of bounding box of bottom flange
- 23. bottom right of bounding box of bottom flange
- 24. mid-depth left of bounding box of bottom flange
- 25. mid-depth centre of bounding box of bottom flange
- 26. mid-depth right of bounding box of bottom flange
- 27. top left of bounding box of bottom flange
- 28. top centre of bounding box of bottom flange
- 29. top right of bounding box of bottom flange

30. *undefined*

- 31. bottom left of bounding box of top flange
- 32. bottom centre of bounding box of top flange
- 33. bottom right of bounding box of top flange
- 34. mid-depth left of bounding box of top flange
- 35. mid-depth centre of bounding box of top flange
- 36. mid-depth right of bounding box of top flange
- 37. top left of bounding box of top flange
- 38. top centre of bounding box of top flange
- 39. top right of bounding box of top flange

40. *undefined*

In many cases, the flanges of section profiles will have corner radii. For the purposes of defining cardinal points, these are ignored. Where the flanges are tapered, the extreme corners of the bounding box are defined by projecting the slope of the flange to intersect with a projection of the edge of the flange. Where the flange is thicker on one side of the web than the other, the corners defining the larger bounding box are used. Where cardinal points defined for the flanges coincide with those defined for the whole section, lower values of the cardinal point take precedence.

Cardinal points 41-49 are based on the bounding box defined by the extreme corners and the mid-points of the web, and will only exist for open sections. For the purposes of defining cardinal points, the root radii of the section are ignored. Where cardinal points defined for the web coincide with those defined for the whole section, lower values of the cardinal points take precedence. (See Figure 14.3.) The points are numbered as follows:

- 41. bottom left of bounding box of web
- 42. bottom centre of bounding box of web
- 43. bottom right of bounding box of web
- 44. mid-depth left of bounding box of web
- 45. mid-depth centre of bounding box of web
- 46. mid-depth right of bounding box of web

- 47. top left of bounding box of web
- 48. top centre of bounding box of web
- 49. top right of bounding box of web
- 50. *undefined*

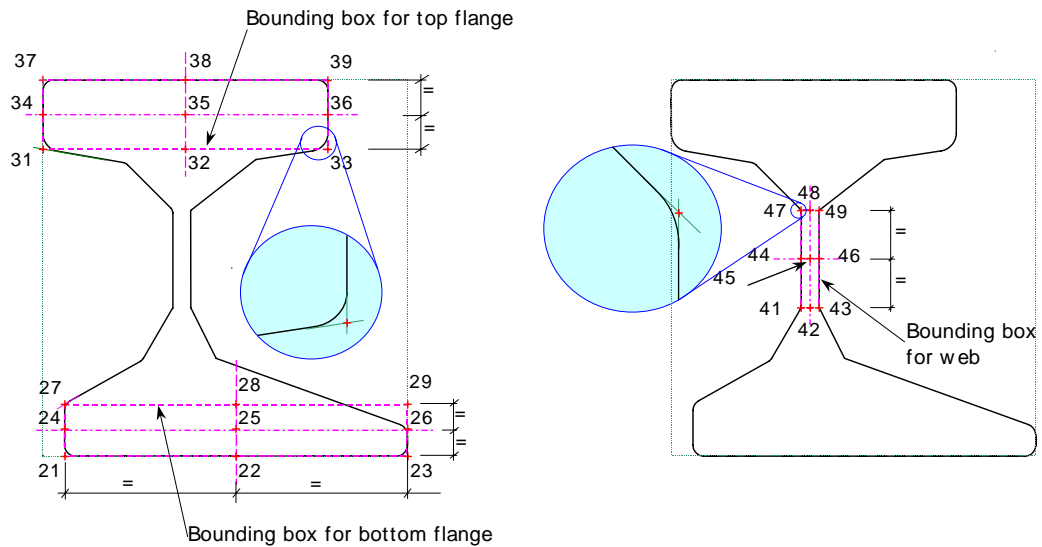


Figure 14.3 Defining cardinal points 21 to 49 on a flanged section profile

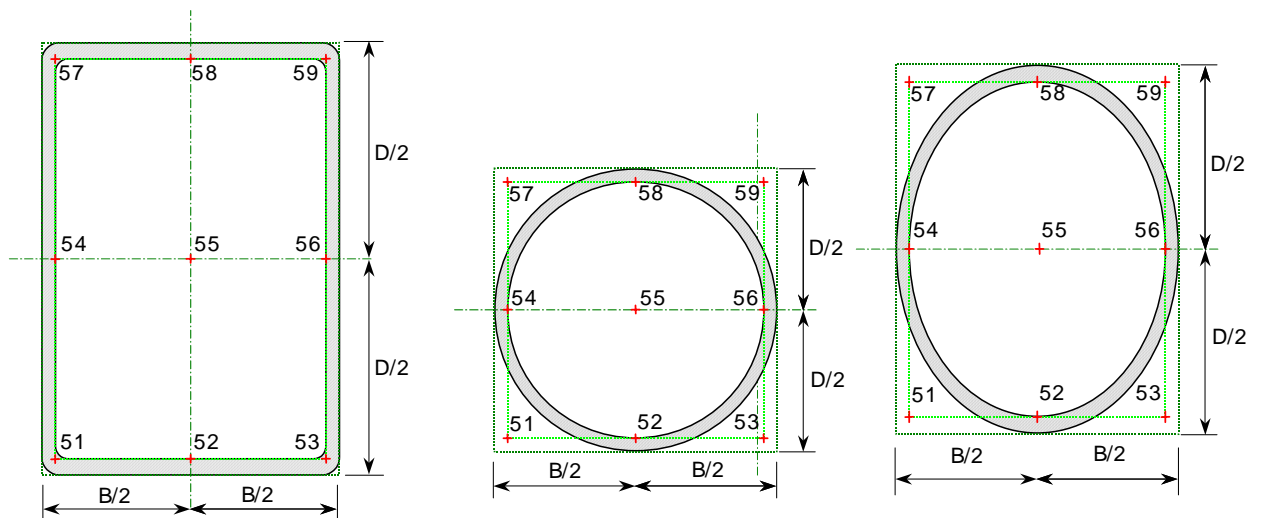


Figure 14.4 Defining cardinal points 51-59 on a box or hollow section profile

Cardinal points 51-59 are based on the internal dimensions of a hollow section. (See Figure 14.4.) These points are numbered as follows:

- 51. bottom left corner of the internal bounding box
- 52. bottom centre of the internal bounding box
- 53. bottom right corner of the internal bounding box
- 54. mid-depth left corner of the internal bounding box
- 55. mid-depth centre of the internal bounding box

- 56. mid-depth right corner of the internal bounding box
- 57. top left corner of the internal bounding box
- 58. top centre of the internal bounding box
- 59. top right corner of the internal bounding box
- 60. *undefined*

Cardinal points above 60 are based on the bounding boxes of compound sections, in such a way that cardinal points 1-9 of the first section become cardinal points 61-69 of the compound section, and cardinal points 1-9 of the second section become cardinal points 71-79 of the compound section; and so on, up to a maximum of four components . (See Figure 14.5.)

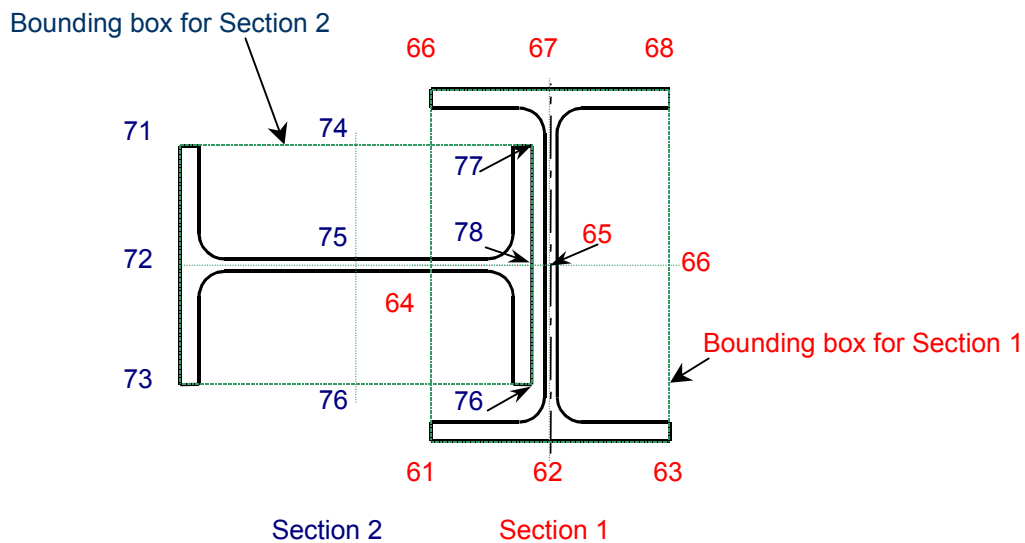


Figure 14.5 *Defining cardinal points for a section_profile_compound*

Cardinal points 20, 30, 40, 50, 60, 70, 80, and 90 are undefined for this edition and should not be used.

EXPRESS specification:

```

*)
TYPE cardinal_point_ref
= INTEGER;
WHERE
    WRTC1 : {0 < SELF < 100};
END_TYPE;
(*)

```

Formal propositions:

WRTC1

The integer value of the cardinal_point_ref shall be greater than 0 and less than 100. That is, only points numbered 1 to 100 are valid cardinal point references.

Notes

New for CIS/2. Modified for 2nd Edition – WHERE rules modified.

14.3 Geometry entity definitions

No entities have been added to this subject area for the 2nd Edition.

The following entities have been modified in this subject area for the 2nd Edition:

- section_profile

14.3.1 section_profile

Entity definition:

A type of structural_frame_item that provides the definition of the two-dimension profile of a cross section. The section_profile may be categorized (through its SUBTYPES) as either simple or complex, corresponding to an implicit or explicit definition (respectively) of the geometry of the section profile. The section_profile may be given a classification in accordance with the appropriate standard or code of practice. It must also be given a cardinal point, which is used to determine the location of the section profile on the part or element.

As a SUBTYPE of structural_frame_item, a section_profile inherits the three attributes item_number, item_name, and item_description. It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a section_profile may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the section_profile to be described in detail
- properties (via the entity item_property_assigned) - this allows extra properties not covered by the entity section_properties to be represented
- item_references (via the entity item_reference_assigned).

A section_profile may also be related to another section_profile (or any other structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

*)

ENTITY section_profile

SUPERTYPE OF (ONEOF(section_profile_simple, section_profile_complex))

SUBTYPE OF (structural_frame_item);

section_classification : OPTIONAL label;

cardinal_point : cardinal_point_ref;

mirrored : LOGICAL;

DERIVE

section_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;

END_ENTITY;

(*

Attribute definitions:***section_classification***

An optional short text reference that may be used to classify the section_profile. This is a function of the local buckling properties of the section. The actual notation used to describe the section classification depend on the standard or code of practice used. Examples of values of section_classification include 'plastic', 'compact', 'semi-compact', 'slender', '1', '2', etc.

cardinal_point

An integer used to identify which point on the section is deemed to be the cardinal point. The cardinal point determines the location of the section profile on the part or element that references this instance of section_profile. The section profile is located such that it sits with its cardinal point on the locating longitudinal axis of the element or part. If adjustments to the described part are to be made (shape, dimensions, orientation, etc.) the cardinal point remains in the same position. (See Figure 14.2, Figure 14.3, Figure 14.4, and Figure 14.5.) It should be noted that section profiles with two axes of symmetry have several cardinal points that coincide.

mirrored

Declares whether the section profile is mirrored (TRUE), not (FALSE), or unknown (UNKNOWN). This attribute is used when asymmetric sections are used in a handed fashion to that defined in the entity definition. For example, an angle section is defined such that its toe appears on the right hand (y-negative) side when viewed from the start (x=0) end. If the angle section is then used in an element or part such that its toe appears on the left hand (y-positive) side when viewed from the start (x=0) end, the section is said to be mirrored (and the attribute is given a value of TRUE).

Declaring a section that is symmetrical about the z-axis to be mirrored has no effect on the definition of the geometry of the section or its local axes. In such a case, a value of UNKNOWN would be more appropriate.

It should be noted that the mirroring acts about the z-axis only – changing a 'right handed' section into a 'left-handed' section. Where the section is used such that its local z-axis points 'down' rather than 'up', this transformation is done by the rotation of the element or the part about its local x-axis (and not by using the mirrored attribute).

The definition of the cardinal points is mirrored when this attribute is set to TRUE. That is 'right' becomes 'left' and 'left' becomes right. This is illustrated in Figure 14.6.

section_ref

A bag of short alphanumerical identifying references for the section profile derived from the attribute item_ref from the SUPERTYPE structural_frame_item. If a standard reference is specified for a section profile it should be represented using the attribute ref in the entity item_reference. A relationship is then made using the entity item_assignment. If the section profile is not assigned an item reference, an empty bag will be returned here.

In most cases, a section profile will be assigned only one reference and that would be taken from a list of standard references from a flavour file. However, there may be times when users wish to add their own references for the section profile. These could be taken from an industry agreed library, or from a proprietary list.

Informal propositions:

The local Coordinate System is based upon the origin as the cardinal point of the section.

As a SUBTYPE of `structural_frame_item`, a `section_profile` (and any of its SUBTYPEs) may be given an `item_reference` through the ANDOR SUPERTYPE construct.

The SUPERTYPE-SUBTYPE relationship is such that a `section_profile` can exist without being instantiated as one of its SUBTYPEs. The SUBTYPEs of `section_profile` will only be required in a physical file where the section profile data is passed by value (explicitly) rather than passed by reference (implicitly) using standard item references. (See `item_reference`.)

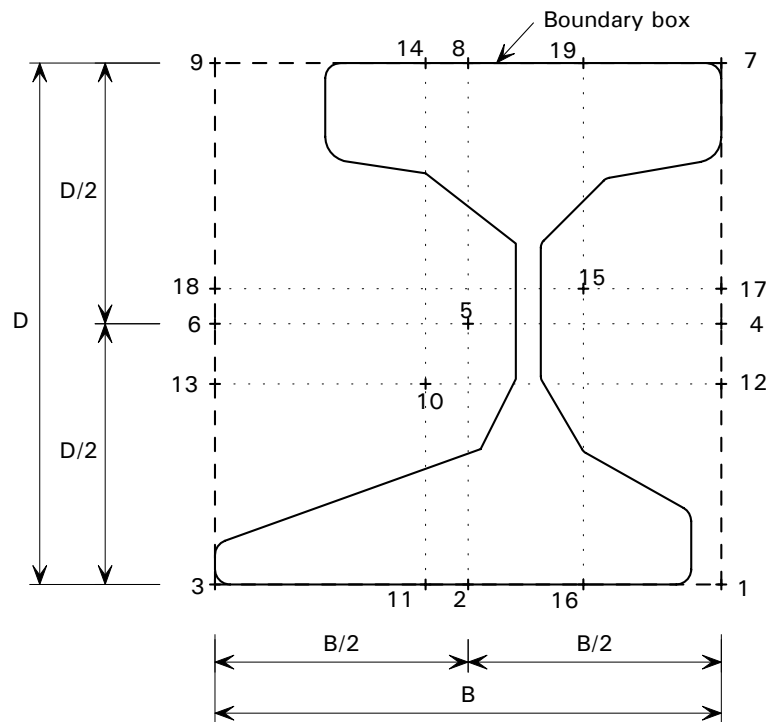


Figure 14.6 *Effect of mirroring on the cardinal point positions*

Notes:

Known as SECTION_PROFILE in CIS/1; SUBTYPEs added.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE clause added.

14.3.2 section_profile_angle

Entity definition:

A type of simple section profile that defines the parameters of an angle section. The `section_profile_angle` must be given a depth and a thickness. It may also be given a width (if an unequal angle). Fillet radii may also be specified for the edges and internal corner of the angle. Where the legs of the angle section are tapered, a value can be given for the slope of the taper. (See Figure 14.7 for an example of a `section_profile_angle` and how its attributes are defined.)

EXPRESS specification:

*)

ENTITY `section_profile_angle`

SUBTYPE OF (`section_profile_simple`);

depth : `positive_length_measure_with_unit`;


```

width : OPTIONAL positive_length_measure_with_unit;
thickness : positive_length_measure_with_unit;
internal_fillet_radius : OPTIONAL positive_length_measure_with_unit;
edge_fillet_radius : OPTIONAL positive_length_measure_with_unit;
leg_slope : OPTIONAL ratio_measure_with_unit;
DERIVE
  width_value : REAL := NVL(width.value_component, depth.value_component);
WHERE
  WRS5 : depth.value_component > thickness.value_component;
END_ENTITY;
(*)

```

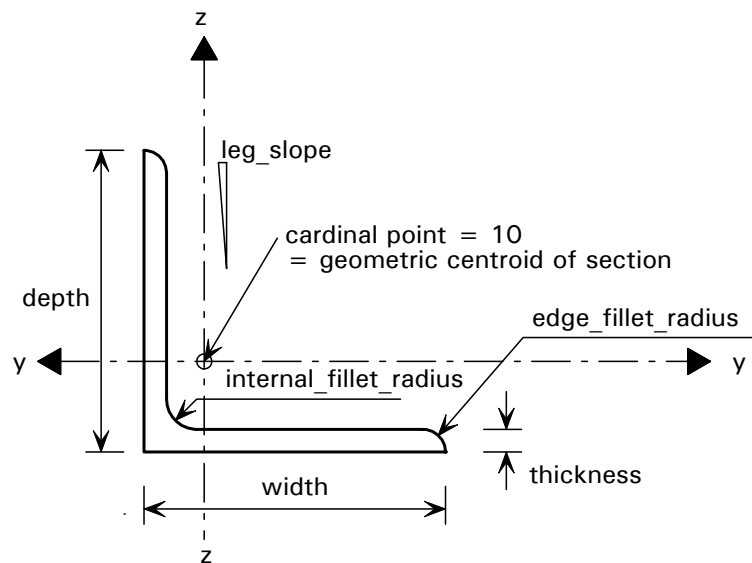


Figure 14.7 *Definition of a section_profile_angle*

Attribute definitions:

depth

Declares the first instance of `positive_length_measure_with_unit` associated with the `section_profile_angle`, which provides the numerical value with an appropriate unit of the overall depth of the angle section. This linear dimension is measured parallel to the local z-axis accordance. For an equal angle section, this parameter provides the length of both legs of the angle.

width

Declares the second instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_angle` to provides the numerical value with an appropriate unit of the overall width of the angle section. This linear dimension is measured parallel to the local y-axis. For an equal angle section, this parameter will be NULL (encoded in the STEP Part 21 file as \$).

thickness

Declares the third instance of `positive_length_measure_with_unit` associated with the `section_profile_angle`, which provides the numerical value with an appropriate unit of the leg thickness of the angle section. This linear dimension is measured in accordance with the appropriate standard or code of practice. It is assumed that the two legs are of equal thickness. Where the legs are tapered, the measurement of leg thickness is taken at a

point specified in the appropriate standard or code of practice (e.g. at midway along the external dimension of the leg).

internal_fillet_radius

Declares the fourth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_angle` to provide the numerical value with an appropriate unit of the internal fillet radius of the angle section. This linear dimension is measured in accordance with the appropriate standard or code of practice.

edge_fillet_radius

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_angle` to provide the numerical value with an appropriate unit of the edge radius at the internal faces at the ends at the legs of the angle section. This linear dimension is measured in accordance with the appropriate standard or code of practice.

leg_slope

Declares the instance of `ratio_measure_with_unit` that may be associated with the `section_profile_angle` to provide the numerical value with an appropriate unit of the slope of the taper of the legs of the angle section. This ratio of the rate of change of leg thickness is measured in accordance with the appropriate standard or code of practice. It is implied that both legs have equal slopes. For unequal angles, it is the angle of the slope that is kept constant rather than the differences in the dimensions at the beginning and end of the legs.

width_value

This attribute derives the real number value of the width of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `width`. If the attribute `width` has been assigned a NULL value, the width is set equal to the real number value associated with instance of `positive_length_measure_with_unit` referenced by the attribute `depth`. That is, if a width is not specified, the angle is assumed to be an equal angle.

Formal propositions:

DERIVE

The derived attribute `width_value` will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS5

The real number value associated with instance of `positive_length_measure_with_unit` referenced by the attribute `depth` shall be greater than the real number value associated with instance of `positive_length_measure_with_unit` referenced by the attribute `thickness`. In other words, the leg depth of an angle must be greater than its thickness. It is implied that both attributes are measured using the same unit.

Informal propositions:

The local axes of the section's coordinate system are defined as shown in Figure 14.7; that is, with one toe in the negative y-direction and the toe of the other (longer) leg in the positive z-direction. The origin of this coordinate system is located at the cardinal point (inherited from the SUPERTYPE `section_profile`). Where the section is used such that one toe points in the positive y-direction, the section is said to be mirrored. (See also the definition of the entity `section_profile`.)

The attributes depth, width, thickness, internal_fillet_radius, and edge_fillet_radius are all measured using the same unit (e.g. mm).

Notes:

Known as ANGLE_SECT in CIS/1.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.3 section_profile_centrelines

Entity definition:

A type of complex section profile that is defined explicitly by the curve of its centreline and a thickness. The section profile is defined as having a constant thickness. This entity makes use of a STEP Part 42 geometric entity to complete its definition.

This entity is likely to be used to represent cold-formed sections such as the purlins and siderails illustrated in Figure 14.8.

EXPRESS specification:

```

*)
ENTITY section_profile_centrelines
  SUBTYPE OF (section_profile_complex);
    centreline : curve;
    thickness : positive_length_measure_with_unit;
END_ENTITY;
(*)
  
```

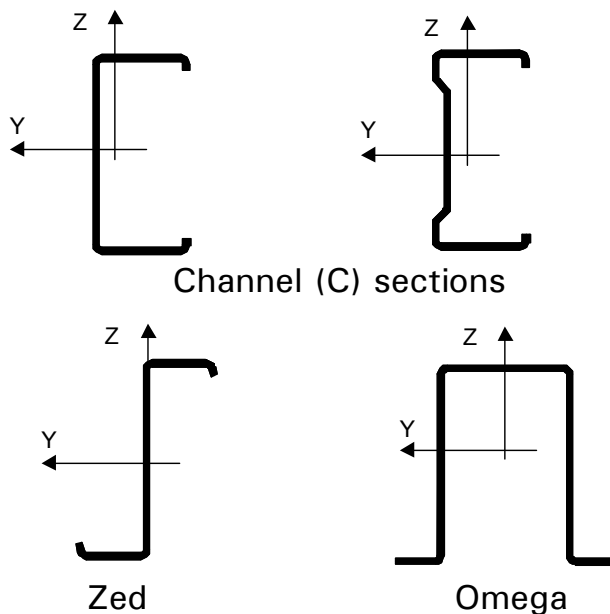


Figure 14.8 *Defining centreline section profiles*

Attribute definitions:

centreline

Declares the instance of curve associated with the section_profile_centrelines, which defines the centreline of the section profile. There must be one (and only one) instance of curve referenced by each instance of section_profile_centrelines.

thickness

Declares the instance of `positive_length_measure_with_unit` associated with the `section_profile_centrelines`, which provide the numerical value with an appropriate unit of the thickness of the section profile. The thickness is measured perpendicular to the centreline and is taken as being constant throughout the defining curve.

Notes:

New for CIS/2.

The entity curve is defined in STEP Part 42.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.4 section_profile_channel**Entity definition:**

A type of simple section profile that defines the parameters of a channel profile. The channel must be given an overall depth, a flange width, and flange and web thicknesses. The channel may also be given values for the radius of the internal root and the slope of the flanges. This section is defined as being symmetrical about its y-y axis such that the top and bottom flanges are of equal size.

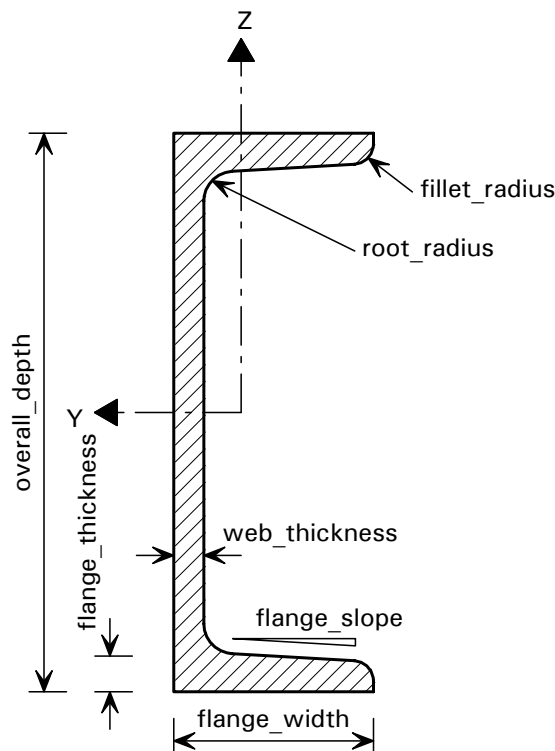


Figure 14.9 *Definition of a section_profile_channel*

EXPRESS specification:

*)

ENTITY `section_profile_channel`

SUBTYPE OF (`section_profile_simple`);

`overall_depth` : `positive_length_measure_with_unit`;

`flange_width` : `positive_length_measure_with_unit`;

```

    flange_thickness : positive_length_measure_with_unit;
    web_thickness : positive_length_measure_with_unit;
    root_radius : OPTIONAL positive_length_measure_with_unit;
    fillet_radius : OPTIONAL positive_length_measure_with_unit;
    flange_slope : OPTIONAL ratio_measure_with_unit;
DERIVE
    overall_depth_value : REAL := overall_depth.value_component;
    flange_width_value : REAL := flange_width.value_component;
    flange_thickness_value : REAL := flange_thickness.value_component;
    web_thickness_value : REAL := web_thickness.value_component;
WHERE
    WRS6 : flange_thickness_value < (overall_depth_value/2);
    WRS7 : web_thickness_value < flange_width_value;
END_ENTITY;
(*)

```

Attribute definitions:

overall_depth

Declares the first instance of `positive_length_measure_with_unit` associated with the `section_profile_channel`, which provides the numerical value with an appropriate unit of the overall depth of the channel section. This linear dimension is measured parallel to the local z-axis.

flange_width

Declares the second instance of `positive_length_measure_with_unit` associated with the `section_profile_channel`, which provides the numerical value with an appropriate unit of the overall width of the flanges of the channel section. This linear dimension is measured parallel to the local y-axis.

flange_thickness

Declares the third instance of `positive_length_measure_with_unit` associated with the `section_profile_channel`, which provides the numerical value with an appropriate unit of the thicknesses of the flanges of the channel section. This linear dimension is measured parallel to the local z-axis. Where the flanges are tapered, the measurement of flange thickness is taken at a point specified in the appropriate standard or code of practice (e.g. midway along the internal flange leg dimension). The section is defined such that the thicknesses of both top and bottom flanges are equal.

web_thickness

Declares the fourth instance of `positive_length_measure_with_unit` associated with the `section_profile_channel`, which provides the numerical value with an appropriate unit of the thicknesses of the web of the channel section. This linear dimension is measured parallel to the local y-axis.

root_radius

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_channel` to provide the numerical value with an appropriate unit of the radius of the internal root of the channel section where the flange meets the web. This linear dimension is measured in accordance with the appropriate standard or code of practice.

fillet_radius

Declares the sixth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_channel` to provide the numerical value with an appropriate unit of the radius of the internal edge of the toe of the channel section. This linear dimension is measured in accordance with the appropriate standard or code of practice.

flange_slope

Declares the instance of `ratio_measure_with_unit` that may be associated with the `section_profile_channel` to provide the numerical value with an appropriate unit of the internal slope of the flanges of the channel section. This linear dimension is measured in accordance with the appropriate standard or code of practice. The section is defined such that the channel has equal top and bottom flanges and each flange has the same slope.

overall_depth_value

This attribute derives the real number value of the depth of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_depth`. This value is used in WHERE rule WRS6.

flange_width_value

This attribute derives the real number value of the flange width of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `flange_width`. This value is used in WHERE rule WRS7.

flange_thickness_value

This attribute derives the real number value of the flange thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `flange_thickness`. This value is used in WHERE rule WRS6.

web_thickness_value

This attribute derives the real number value of the web thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `web_thickness`. This value is used in WHERE rule WRS7.

Formal propositions**DERIVE**

The derived attributes `overall_depth_value`, `flange_width_value`, `flange_thickness_value` and `web_thickness_value` will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS6

The real number value derived by the attribute `flange_thickness_value` shall be less than half the real number value derived by the attribute `overall_depth_value`. In other words, the flange thickness must be less than half the overall depth. It is implied that these two parameters are measured using the same units.

WRS7

The real number value derived by the attribute `web_thickness_value` shall be less than the real number value derived by the attribute `flange_width_value`. In other words, the web thickness must be less than the flange width. It is implied that these two parameters are measured using the same units.

Informal propositions:

The local axes of the section's coordinate system are defined as shown in Figure 14.9; that is, with the toes in the negative y-direction. The origin of this coordinate system is located at the cardinal point (inherited from the SUPERTYPE `section_profile`). Where the section is used such that the toes point in the positive y-direction, the section is said to be mirrored. (See also the definition of the entity `section_profile`.)

The attributes `overall_depth`, `flange_width`, `flange_thickness`, `web_thickness`, and `root_radius` are all measured using the same unit (e.g. mm).

Notes:

New for CIS/2; addressed by `I_TYPE_SECT`.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

14.3.5 section_profile_circle**Entity definition:**

A type of simple section profile that provides the defining parameter of a circular section profile. If this entity is instanced on its own, it represents a solid circular section. If this entity is instanced as the SUBTYPE (`section_profile_circle_hollow`) it represents a hollow circular section.

EXPRESS specification:

*)

ENTITY `section_profile_circle`

SUPERTYPE OF (`section_profile_circle_hollow`)

SUBTYPE OF (`section_profile_simple`);

`external_radius` : `positive_length_measure_with_unit`;

END_ENTITY;

(*

Attribute definitions:***external_radius***

Declares the instance of `positive_length_measure_with_unit` associated with the `section_profile_circle`, which provides the numerical value with an appropriate unit of the external radius of the circular section. (The overall diameter of the circular section is twice this value.)

Notes:

Known as `CIRCLE_SECT` in CIS/1.

An oval section should be represented by a instance of either `section_profile_centreline` or `section_profile_edge_defined`.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

14.3.6 section_profile_circle_hollow

Entity definition:

A type of section_profile_circle which provides the second parameter required to define a circular hollow section.

EXPRESS specification:

```

*)
ENTITY section_profile_circle_hollow
SUBTYPE OF (section_profile_circle);
    wall_thickness : positive_length_measure_with_unit;
DERIVE
    external_radius_value : REAL :=
        (SELF.section_profile_circle.external_radius.value_component);
    wall_thickness_value : REAL := wall_thickness.value_component;
    inside_diameter : REAL := ((external_radius_value - wall_thickness_value)*2);
WHERE
    WRS8 : wall_thickness_value < external_radius_value;
END_ENTITY;
(*

```

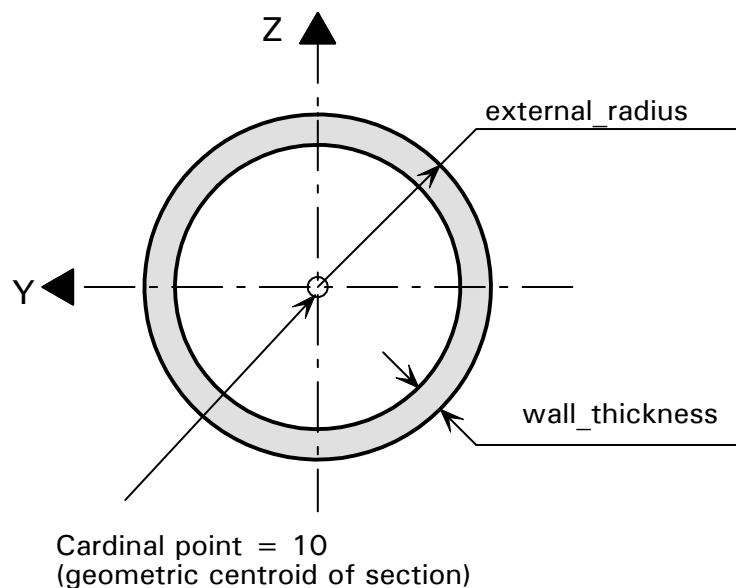


Figure 14.10 Definition of a section_profile_circle_hollow

Attribute definitions:

wall_thickness

Declares the instance of positive_length_measure_with_unit associated with the section_profile_circle_hollow, which provides the numerical value with an appropriate unit of the wall thickness of the circular section. (The external radius of the circular hollow section is inherited from the SUPERTYPE.)

external_radius_value

This attribute derives the real number value of the external radius of the section from the instance of positive_length_measure_with_unit referenced by the attribute external_radius

(inherited from the SUPERTYPE section_profile_circle). This value is used in the WHERE rule WRS8.

wall_thickness_value

This attribute derives the real number value of the wall thickness of the section from the instance of positive_length_measure_with_unit referenced by the attribute wall_thickness. This value is used in the WHERE rule WRS8.

inside_diameter

This attribute derives the real number value of the inside diameter of the section from the values of the derived attributes external_radius_value and wall_thickness_value.

Formal propositions:

DERIVE

The derived attributes overall_depth_value, flange_width_value, flange_thickness_value and web_thickness_value will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS8

The real number value derived by the attribute wall_thickness_value shall be less than the real number value derived by the attribute external_radius_value. That is, the wall thickness must be less than the external radius. In other words, the hollow section must not be solid. It is implied that these two parameters are measured using the same units.

Informal propositions:

The local axes of the section's coordinate system are defined as shown in Figure 14.10. The origin of this coordinate system is located at the cardinal point (inherited from the SUPERTYPE section_profile). Declaring this section to be mirrored would have no effect on the definition of the attributes as the section is symmetrical about the z-axis. (See also the definition of the entity section_profile.)

The attributes overall_depth, flange_width, flange_thickness, web_thickness, and root_radius are all measured using the same unit (e.g. mm).

Notes:

New for CIS/2; addressed by CIRCLE_SECT in CIS/1.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.7 section_profile_complex

Entity definition:

An abstract type of section profile whose geometry is defined explicitly using constructs from STEP Part 42. The complexity arises due to the shape of the section rather than the material of the associated part.

EXPRESS specification:

*)

ENTITY section_profile_complex

ABSTRACT SUPERTYPE OF (ONEOF

(section_profile_compound,

section_profile_edge_defined,

```

    section_profile_centreline,
    section_profile_derived))
SUBTYPE OF (section_profile);
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `section_profile`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `section_profile`, and to collect together the SUBTYPES under a common category.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.8 section_profile_compound**Entity definition:**

A type of complex section profile that is as made up from two or more other section profiles. Each of these section profiles is placed with their local coordinate system related to the cardinal point of the compound section profile. Each of the section profiles may also be rotated with respect to the local coordinate system of the compound section profile. In some cases, the local coordinate system of the first section profile referenced would provide the local coordinate system of the compound section profile to locate and rotate the remaining section profiles. Back-to-back angles and crane rails are simple examples of compound section profiles.

Although composite beams should be represented as assemblies with two or more component parts, this entity could be used to represent the transformed section profile of a composite beam for analysis purposes using an appropriate value for the modular ratio. In this case, the `section_profile_compound` would be referenced by an `element_curve_simple`. Further, the attribute `item_description` inherited from the SUPERTYPE `structural_frame_item` should be set to 'transformed composite section'.

EXPRESS specification:

```

*)
ENTITY section_profile_compound
SUBTYPE OF (section_profile_complex);
    component_sections : LIST [2:?] OF section_profile;
    positions : LIST [2:?] OF point;
    orientations : LIST [2:?] OF orientation_select;
DERIVE
    number_of_sections : INTEGER := SIZEOF (component_sections);
WHERE

```

```

WRS9 : SIZEOF (positions) = number_of_sections;
WRS10 : SIZEOF (orientations) = number_of_sections;
WRS11 : SIZEOF(QUERY(sections <* component_sections | sections := (SELF))) = 0;
END_ENTITY;
(*)

```

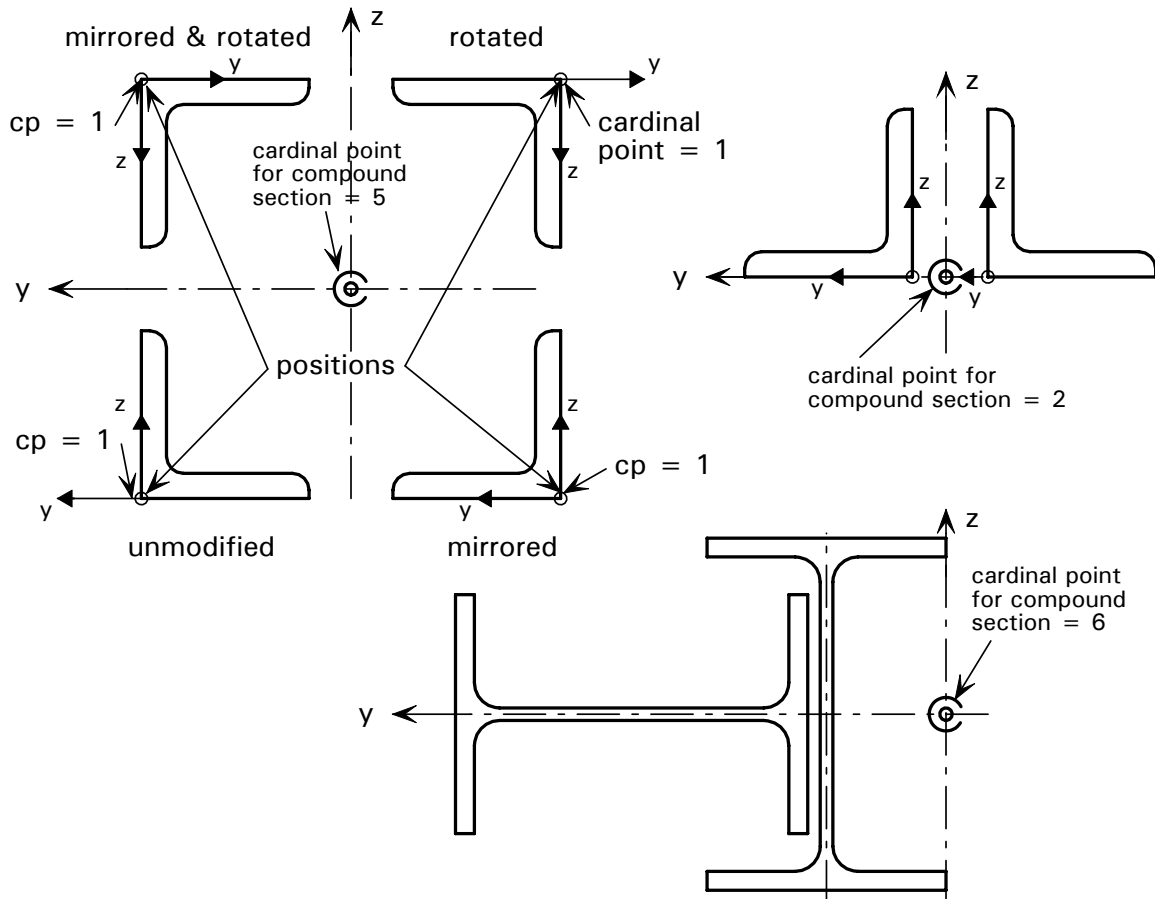


Figure 14.11 *Examples of section_profile_compound*

Attribute definitions:

component_sections

Declares the list of 2 or more instances of `section_profile` associated with the `section_profile_compound`, which contribute to the final section profile. There must be at least two instances of `section_profile` referenced by each instance of `section_profile_compound`. Each `section_profile` is located with its cardinal point at the corresponding point (in the `positions` aggregation), and is rotated by the corresponding angle (in the `orientations` aggregation). The order of the instances in the aggregation is important and instances may be repeated.

positions

Declares the list of 2 or more instances of `point` associated with the `section_profile_compound`, which locate each of the section profiles contributing to the final section profile. The points are defined with respect to the local coordinate system of the compound section profile. The cardinal point of the compound section profile is taken as the origin of its coordinate system. There must be at least two instances of `point` referenced by each instance of `section_profile_compound`. Each component section is

located with its cardinal point at the corresponding point in this aggregation and is rotated by the corresponding angle (in the orientations aggregation).

The order of the instances in the aggregation is important and instances may be repeated. The points are all located with respect to the cardinal point of the compound section profile. This may, but need not, be the centroid of the compound section. If the cardinal point of the first component section provides the (unshifted) cardinal point of the local coordinate system compound section profile, then the first point instance in the aggregation shall be equivalent to (0.0, 0.0, 0.0).

orientations

Declares the list of 2 or more instances of `plane_angle_measure_with_unit` or `direction` associated with this instance of `section_profile_compound`, which provide the numerical values of the rotations of the section profiles contributing to the final section profile about their cardinal points.

The choice of which entity is used is made through the `SELECT` type `orientation_select`. If this attribute references an instance of `plane_angle_measure_with_unit` (via the `SELECT` type) then the value assigned to each angle represents a measure of the rotation of the contributing section profile about its x-axis so that it corresponds to the local axes of the compound section profile. Clockwise rotation from component section to final compound section is taken as positive.

If this attribute references an instance of `direction` (via the `SELECT` type) then the orientation of the section profile is defined by the two or three real number values of the `direction_ratios` attribute (of the entity `direction`). These numbers define the 'orientation vector' for the section profile within the local coordinate system of the compound section. That is, the direction of the re-oriented local z-axis relative to the local coordinate system of the compound section. For example, the normal (unrotated) orientation is given as (0.0, 0.0, 1.0).

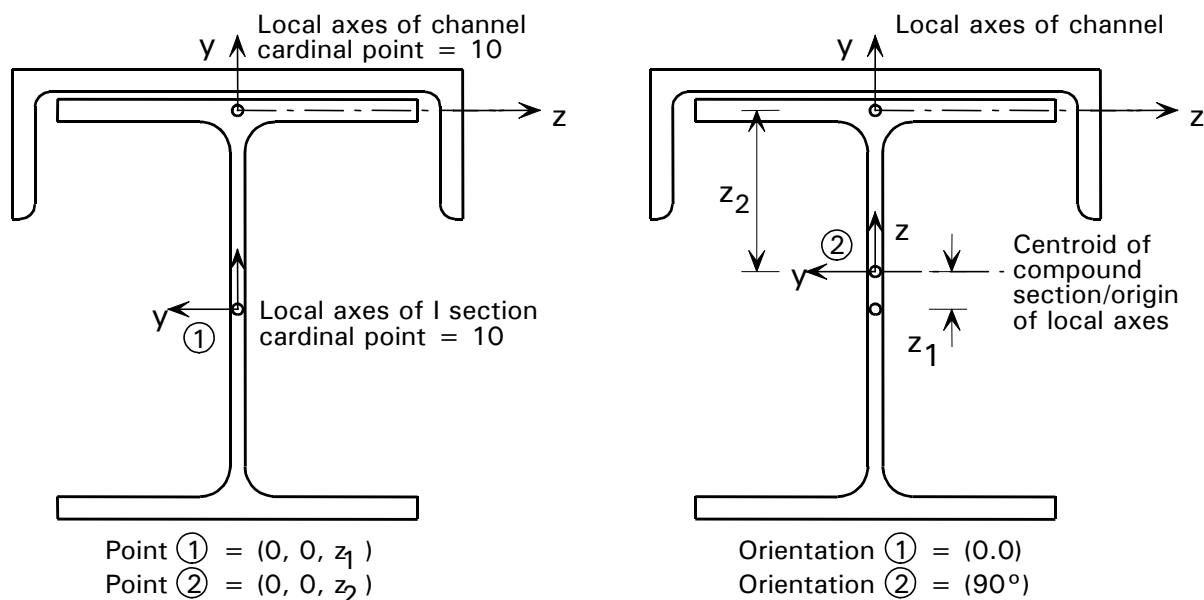


Figure 14.12 *Relocating the components of a section_profile_compound*

There must be at least two instances of `plane_angle_measure_with_unit` or `direction` referenced by each instance of `section_profile_compound`. References may be repeated.

Each component section is located with its cardinal point at the corresponding point (in the positions aggregation) and is rotated by the corresponding angle in this aggregation).

The order of the instances in the aggregation is important and instances may be repeated. If the first component section provides the unrotated local coordinate system of the compound section profile, then the first instance in the aggregation shall provide an angle equivalent to 0.0, or an orientation vector equivalent to [0.0, 0.0, 1.0].

It should be noted that if a direction with three values is used to define the orientation, one of the values assigned to the `direction_ratios` attribute should be zero, while the other two values should be non-zero. As the orientation is defined in the plane of the cross section, a direction assigned three non-zero values would be incorrect. The number of values assigned to the `direction_ratios` attribute is governed by the global rule `compatible_dimension`. This ensures that the count of `direction_ratios` matches the `coordinate_space_dimension` of the `geometric_context` in which the direction is geometrically founded. This should be the same `geometric_context` used for the other `geometric_representation_items` in the model; e.g. the points defining the positions of the section profiles contributing to the final section profile.

number_of_sections

This attribute derives the integer value of the number of sections that define this compound section profile from the size of the list of instances of `section_profile` referenced by the attribute `component_sections`. This value is used in WHERE rules WRS9 and WRS10

Formal propositions:

DERIVE

The derived attribute `number_of_sections` will not appear when the instance is encoded in a STEP Part 21 file.

WRS9

The size of the list of instances of point referenced by the attribute `positions` shall be equal to the integer value derived by the attribute `number_of_sections`. That is, there shall be as many points as there are sections.

WRS10

The size of the list of instances of `plane_angle_measure_with_unit` and/or `direction` referenced by the attribute `orientations` shall be equal to the integer value derived by the attribute `number_of_sections`. That is, there shall be as many orientations as there are sections. If all of the sections are rotated by the same amount, the list should reference the same instance of `plane_angle_measure_with_unit` or `direction` as many times as required. If none of the sections are rotated, the list should reference an instance of `plane_angle_measure_with_unit` or `direction` equivalent to zero as many times as required.

WRS11

The size of the set of instances of `section_profile` that are instance equal to this instance of `section_profile_compound` shall equal zero. This prevents instance recursion while permitting entity type recursion. In other words, a compound section profile cannot be a component of itself, but may be a component of another compound section profile.

Informal propositions:

The LIST data types imply that order is important and that instance repetition is allowed. It is assumed that each instance of each aggregation is populated in the correct sequence. That is, the first `section_profile` instanced is located at the first point instanced, and is rotated by the first `plane_angle_measure_with_unit` or `direction` instanced, etc.

Notes:

New for CIS/2.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.9 section_profile_derived**Entity definition:**

A type of complex section profile that is derived from another section profile. This may apply to castellated sections, 'cellform' sections, tees cut from I or H sections. How the profile is actually derived (i.e. the manufacturing processes) is not described here. The section profile is derived (or cut) from a single profile.

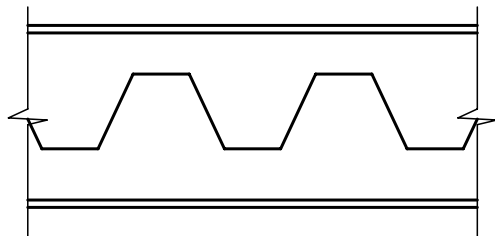
EXPRESS specification:

*)

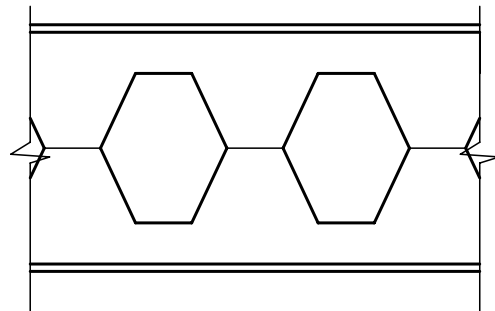
```
ENTITY section_profile_derived
SUBTYPE OF (section_profile_complex);
    original_section : section_profile;
WHERE
    WRS12 : original_section :<>: SELF;
END_ENTITY;
```

(*

Castellated beam

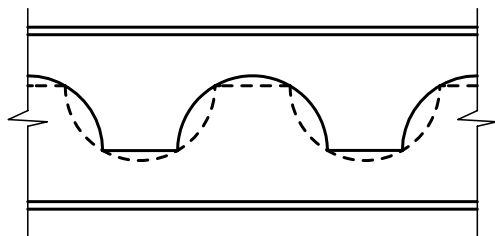


Profile cut in web
of original_section

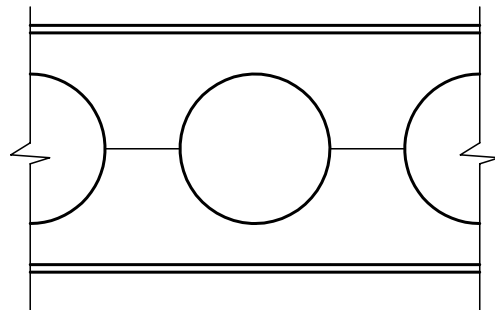


Hexagonal openings in
cellular beam

Cellular beam



Profile cut in web
of original_section



Circular openings in
cellular beam

Figure 14.13 *Examples of section_profile_derived*

Attribute definitions:*original_section*

Declares the instance of `section_profile` associated with this `section_profile_derived`, from which it is derived. There must be one (and only one) instance of `section_profile` referenced by each instance of `section_profile_derived`.

Formal propositions:*WRS12*

The instance of `section_profile` referenced by the attribute `original_section` shall not be instance equal to this instance of `section_profile_derived`. Although entity type recursion is permitted such that a `section_profile_derived` may be derived from another `section_profile_derived`, instance recursion is prohibited so that a `section_profile_derived` cannot be derived from itself.

This entity will normally be instanced in association with the entity `part_prismatic_simple_castellated`.

Notes:

New for CIS/2.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

14.3.10 section_profile_edge_defined**Entity definition:**

A type of complex section profile that is defined by its external edge, and if appropriate, by internal edges, which in turn are defined by a reference to `bounded_curve`. The defining curve is assumed to be closed such that the start and end points of the `bounded_curve` are coincidental. A `section_profile_edge_defined` may also contain some internal features that remove material from the solid cross section defined by other instances of `bounded_curve`. Some extruded sections may need to use this entity for their definition.

EXPRESS specification:

```
*)
ENTITY section_profile_edge_defined
SUBTYPE OF (section_profile_complex);
    external_edge : bounded_curve;
    internal_edges : LIST [0:?] OF bounded_curve;
END_ENTITY;
(*
```

Attribute definitions:*external_edge*

Declares the instance of `bounded_curve` associated with the `section_profile_edge_defined`, which defines the external edge of the complex section profile. There must be one (and only one) instance of `bounded_curve` referenced by each instance of `section_profile_edge_defined`. (See Figure 14.14.)

internal_edges

Declares the list of instances of *bounded_curve* that may be associated with the *section_profile_edge_defined* to define the edges of any internal features that remove material from the complex section profile. If the aggregation is empty, the section profile is solid.

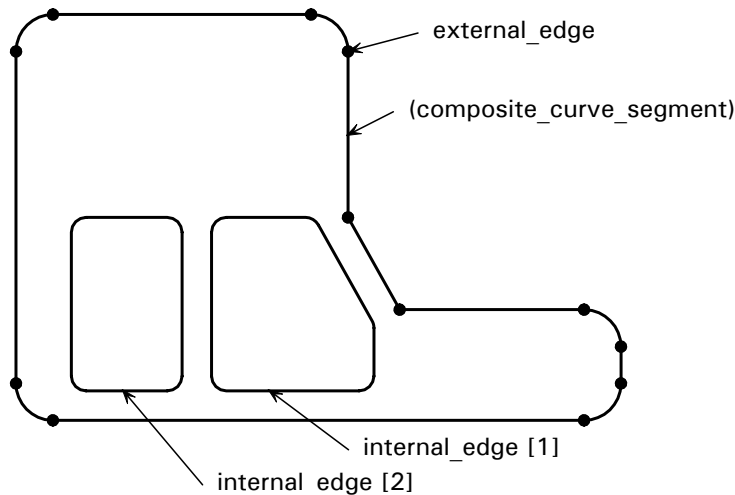


Figure 14.14 *Example of an edge defined section profile*

Notes:

New for CIS/2.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.11 section_profile_i_type**Entity definition:**

A type of simple section profile that provides the defining parameters of a (symmetrical) I section. The *section_profile_i_type* must be given values for its overall depth, width and web thickness. It may also be given an internal depth. If this entity is instanced, it represents a I section that is symmetrical about its major and minor axes; that is both top and bottom flanges are equal and centred on the web. If the SUBTYPE *section_profile_i_type_asymmetric* is instanced, then an asymmetric I section is being represented.

EXPRESS specification:

*)

ENTITY *section_profile_i_type*

SUPERTYPE OF (*section_profile_i_type_asymmetric*)

SUBTYPE OF (*section_profile_simple*);

overall_depth : *positive_length_measure_with_unit*;

overall_width : *positive_length_measure_with_unit*;

web_thickness : *positive_length_measure_with_unit*;

flange_thickness : *positive_length_measure_with_unit*;

internal_depth : OPTIONAL *positive_length_measure_with_unit*;

flange_slope : OPTIONAL *ratio_measure_with_unit*;

root_radius: OPTIONAL *positive_length_measure_with_unit*;


```

    edge_radius : OPTIONAL positive_length_measure_with_unit;
  DERIVE
    overall_depth_value : REAL := overall_depth.value_component;
    overall_width_value : REAL := overall_width.value_component;
    web_thickness_value : REAL := web_thickness.value_component;
    flange_thickness_value : REAL := flange_thickness.value_component;
  WHERE
    WRS13 : flange_thickness_value < (overall_depth_value/2);
    WRS14 : web_thickness_value < overall_width_value;
  END_ENTITY;
  (*

```

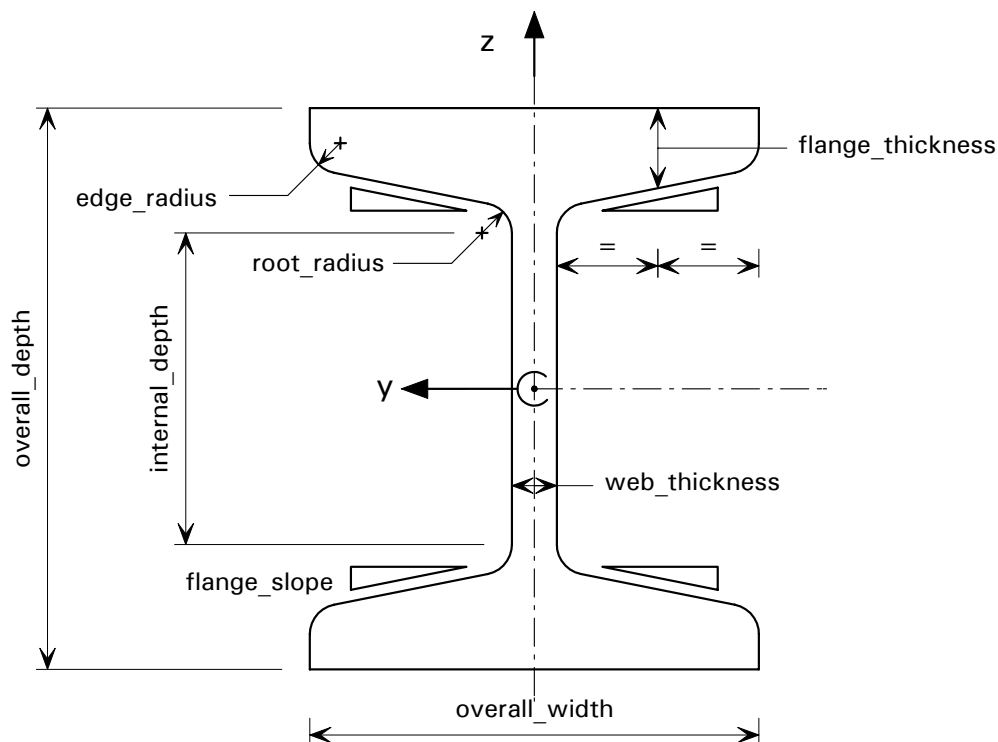


Figure 14.15 Definition of a *section_profile_i_type*

Attribute definitions:

overall_depth

Declares the first instance of *positive_length_measure_with_unit* associated with the *section_profile_i_type*, which provides the numerical value with an appropriate unit of the overall depth of the I section. This linear dimension is measured parallel to the minor (z-z) axis.

overall_width

Declares the second instance of *positive_length_measure_with_unit* associated with the *section_profile_i_type*, which provides the numerical value with an appropriate unit of the overall width of the I section. This linear dimension is measured parallel to the major (y-y) axis.

web_thickness

Declares the third instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type`, which provides the numerical value with an appropriate unit of the web thickness of the I section.

flange_thickness

Declares the fourth instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type`, which provides the numerical value with an appropriate unit of the flange thickness of the I section.

When this entity is instantiated as the SUBTYPE `section_profile_i_type_asymmetric`, this attribute provides the thickness of the top flange. If the flanges are tapered, the flange thickness is measured at a specific position on the flanges given by the appropriate standard or code of practice (e.g. midway along the internal dimension of the flange outstand).

internal_depth

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_i_type` to provide the numerical value with an appropriate unit of the internal depth (i.e. depth between fillets) of the I section.

flange_slope

Declares the first instance of `ratio_measure_with_unit` that may be associated with the `section_profile_i_type` to provide the numerical value with an appropriate unit of the slope (in cross section) of the tapered flange of the I section. This attribute is populated if the flange is tapered such that its inside (web side) slopes. The outside of the flange is defined as being flat. The slope of the flange is taken to be the same on both sides of the web.

For a symmetrical I section, the slope of the top and bottom flanges are equal. When this entity is instantiated as the SUBTYPE `section_profile_i_type_asymmetric`, this attribute provides the slope of the top flange. When the SUBTYPE `section_profile_i_type_rail` is instantiated, this attribute provides the slope of the outer taper of the top flange.

root_radius

Declares the sixth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_i_type` to provide the numerical value with an appropriate unit of the fillet radius of the 'root' between the web and the flange of the I section.

For a symmetrical I section, the root radii at the top and bottom are equal. When this entity is instantiated as the SUBTYPE `section_profile_i_type_asymmetric`, this attribute provides the root radius between the web and the top flange.

edge_radius

Declares the seventh instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_i_type` to provide the numerical value with an appropriate unit of the radius of the inner (web-side) toe edges of the flange of the I section. The outer toe edges are taken as being square. Both toe edges on both sides of the web are assumed to have equal radii.

For a symmetrical I section, the edge radii of the top and bottom flanges are equal. When this entity is instantiated as the SUBTYPE `section_profile_i_type_asymmetric`, this attribute provides the edge radius for the top flange.

overall_depth_value

This attribute derives the real number value of the overall depth of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_depth`. This value is used in WHERE rule WRS13.

overall_width_value

This attribute derives the real number value of the overall width of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_width`. This value is used in WHERE rule WRS14.

web_thickness_value

This attribute derives the real number value of the web thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `web_thickness`. This value is used in WHERE rule WRS14.

flange_thickness_value

This attribute derives the real number value of the flange thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `flange_thickness`. This value is used in WHERE rule WRS13.

Formal propositions:*DERIVE*

The derived attributes `overall_depth_value`, `overall_width_value`, `web_thickness_value` and `flange_thickness_value` will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS13

The real number value derived by the attribute `flange_thickness_value` shall be less than half the real number value derived by the attribute `overall_depth_value`. That is, the I section must be twice as deep as the flanges are thick.

WRS14

The real number value derived by the attribute `web_thickness_value` shall be less than the real number value derived by the attribute `overall_width_value`. That is, the I section must be wider than the web is thick.

Informal propositions:

The origin of the section's local coordinate system is located at the cardinal point (inherited from the SUPERTYPE `section_profile`). Declaring this section to be mirrored may affect the position of the local axes relative to the profile, but this will not affect the definition of the attributes, as the section is symmetrical about its minor axis. (See also the definition of the entity `section_profile`.)

The attributes `overall_depth`, `overall_width`, `web_thickness`, `flange_thickness`, and `internal_depth` are all measured using the same unit (e.g. mm).

Notes:

Known as `I_TYPE_SECT` in CIS/1.

See diagram 67 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.12 section_profile_i_type_asymmetric

Entity definition:

A type of I type section whose top and bottom flanges are of different sizes. The top and bottom flanges are both centred on the web. (See Figure 14.16.)

EXPRESS specification

*)

ENTITY section_profile_i_type_asymmetric

SUPERTYPE OF (section_profile_i_type_rail)

SUBTYPE OF (section_profile_i_type);

top_flange_width : positive_length_measure_with_unit;

bottom_flange_width : positive_length_measure_with_unit;

bottom_flange_thickness : positive_length_measure_with_unit;

bottom_root_radius : OPTIONAL positive_length_measure_with_unit;

bottom_flange_slope : OPTIONAL ratio_measure_with_unit;

bottom_flange_edge_radius : OPTIONAL positive_length_measure_with_unit;

WHERE

WRS43 : (SELF\section_profile_i_type.overall_width :=: top_flange_width) OR

(SELF\section_profile_i_type.overall_width :=: bottom_flange_width);

END_ENTITY;

(*

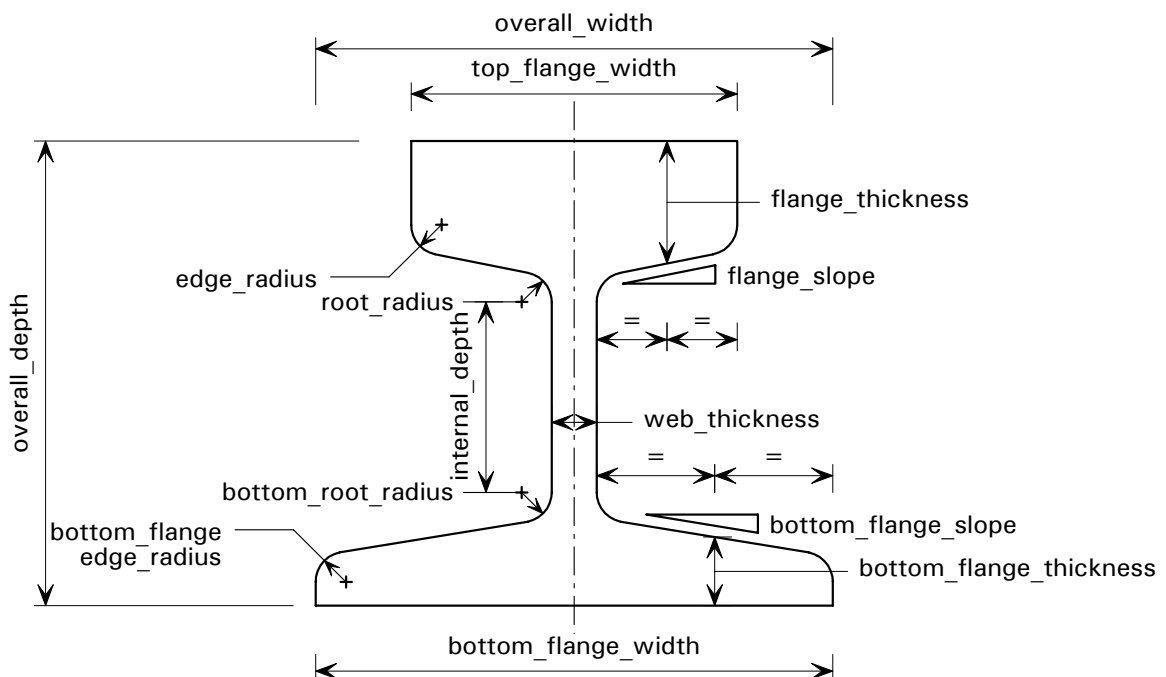


Figure 14.16 Definition of section_profile_i_type_asymmetric

Attribute definitions:

top_flange_width

Declares the first instance of positive_length_measure_with_unit associated with the section_profile_i_type_asymmetric, which provides the numerical value with an appropriate unit of the width of the top flange of the asymmetric I section. This linear dimension is measured parallel to the major (y-y) axis.

bottom_flange_width

Declares the second instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type_asymmetric`, which provides the numerical value with an appropriate unit of the width of the bottom flange of the asymmetric I section. This linear dimension is measured parallel to the major (y-y) axis.

bottom_flange_thickness

Declares the third instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type_asymmetric`, which provides the numerical value with an appropriate unit of the thickness of the bottom flange of the asymmetric I section. (The top flange thickness is provided by the attribute `flange_thickness` inherited from the SUPERTYPE.)

bottom_root_radius

Declares the fourth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_i_type_asymmetric` to provide the numerical value with an appropriate unit of the fillet radius of the ‘root’ between the web and the bottom flange of the asymmetric I section. (The top root radius is provided by the attribute `root_radius` inherited from the SUPERTYPE.)

bottom_flange_slope

Declares the first instance of `ratio_measure_with_unit` that may be associated with the `section_profile_i_type_asymmetric` to provide the numerical value with an appropriate unit of the slope (in cross section) of the tapered bottom flange of the asymmetric I section. This attribute is populated if the bottom flange is tapered such that its topside (web side) slopes. The underside of the bottom flange is taken as being flat. The slope of the flange is assumed to be the same on both sides of the web. (The slope of the top flange is provided by the attribute `flange_slope` inherited from the SUPERTYPE.)

When the SUBTYPE `section_profile_i_type_rail` is instanced, this attribute provides the slope of the outer taper of the bottom flange.

bottom_flange_edge_radius

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_i_type_asymmetric` to provide the numerical value with an appropriate unit of the radius of the inner (web-side) toe edges of the bottom flange of the asymmetric I section. The outer toe edges are taken as being square. Both toe edges on both sides of the web are assumed to have equal radii. (The edge radius of the top flange is provided by the attribute `edge_radius` inherited from the SUPERTYPE.)

Formal propositions:**WRS43**

The instance of `positive_length_measure_with_unit` referenced by the attribute `overall_width` (inherited from the SUPERTYPE `section_profile_i_type`) shall be the same instance referenced by either `top_flange_width` or `bottom_flange_width`. In other words, either one of the attributes `top_flange_width` or `bottom_flange_width` should be equal to that of the attribute `overall_width` inherited from the SUPERTYPE `section_profile_i_type`.

Notes:

New for CIS/2; partially addressed by `I_TYPE_SECT` in CIS/1.

See diagram 67 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.13 section_profile_i_type_rail

Entity definition:

A type of asymmetric I section whose flanges may have two tapers – one inner and one outer taper – and a transition between the two. The rail section may be doubly symmetrical depending upon the values assigned to the attributes this entity inherits from its SUPERTYPE.

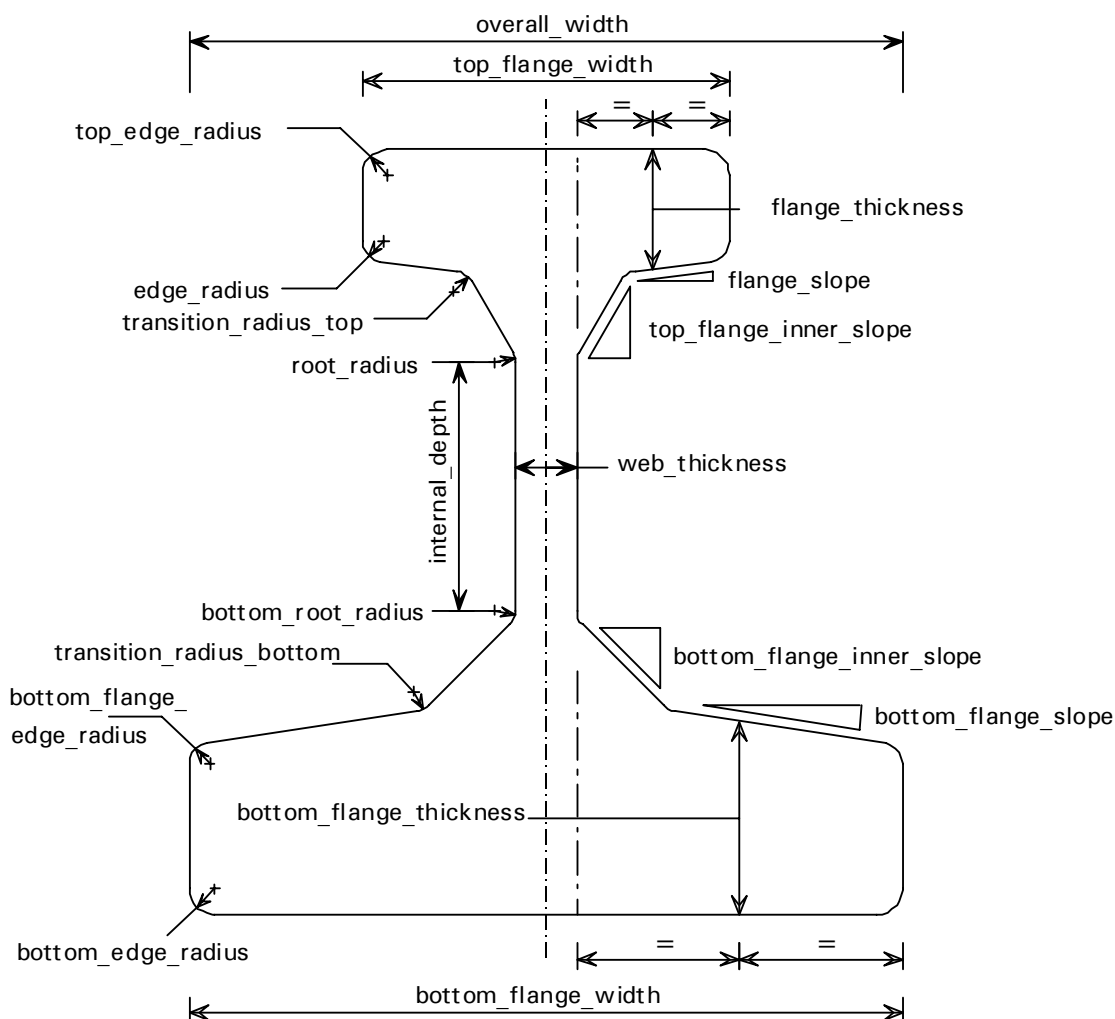


Figure 14.17 Definition of a section_profile_i_type_rail

EXPRESS specification:

*)

ENTITY section_profile_i_type_rail

SUBTYPE OF (section_profile_i_type_asymmetric);

top_edge_radius : positive_length_measure_with_unit;

bottom_edge_radius : positive_length_measure_with_unit;

top_flange_inner_slope : ratio_measure_with_unit;

bottom_flange_inner_slope : ratio_measure_with_unit;

transition_radius_top : OPTIONAL positive_length_measure_with_unit;

transition_radius_bottom : OPTIONAL positive_length_measure_with_unit;

END_ENTITY;

(*)

Attribute definitions:*top_edge_radius*

Declares the first instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type_rail`, which provides the numerical value with an appropriate unit of the outer toe edges of the top flange of the rail section. (The radius for inner toe edges is inherited from the SUPERTYPE). Both toe edges on both sides of the web are assumed to have equal radii.

bottom_edge_radius

Declares the second instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type_rail`, which provides the numerical value with an appropriate unit of the outer toe edges of the bottom flange of the rail section. (The radius for inner toe edges is inherited from the SUPERTYPE). Both toe edges on both sides of the web are assumed to have equal radii.

top_flange_inner_slope

Declares the first instance of `ratio_measure_with_unit` associated with the `section_profile_i_type_rail`, which provides the numerical value with an appropriate unit of the inner (web side) slope (in cross section) of the tapered top flange of the rail section. The topside of the top flange is taken as being flat (i.e. parallel to the y-axis). The slope of the flange is taken as being the same on both sides of the web.

The slope of the outer taper is provided by the attribute `flange_slope` inherited from the SUPERTYPE `section_profile_i_type`.

bottom_flange_inner_slope

Declares the second instance of `ratio_measure_with_unit` associated with the `section_profile_i_type_rail`, which provides the numerical value with an appropriate unit of the inner (web side) slope (in cross section) of the tapered bottom flange of the rail section. The underside of the bottom flange is taken as being flat (i.e. parallel to the y-axis). The slope of the flange is taken as being the same on both sides of the web.

The slope of the outer taper is provided by the attribute `bottom_flange_slope` inherited from the SUPERTYPE `section_profile_i_type_asymmetric`.

transition_radius_top

Declares the third instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type_rail`, which provides the numerical value with an appropriate unit of the radius of the transition curve between the two tapers of the top flange of the rail section. Both transition curves on both sides of the web are assumed to have equal radii.

transition_radius_bottom

Declares the fourth instance of `positive_length_measure_with_unit` associated with the `section_profile_i_type_rail`, which provides the numerical value with an appropriate unit of the radius of the transition curve between the two tapers of the bottom flange of the rail section. Both transition curves on both sides of the web are assumed to have equal radii.

Notes:

New for CIS/2.

See diagram 67 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.14 section_profile_rectangle

Entity definition:

A type of simple section profile that provides the defining parameters of a rectangular section. The `section_profile_rectangle` must be given an overall depth. It may also be given an overall width; if it is not, then the entity represents a square section. It may also be given a value for the corner fillet radius.

If this entity is instanced on its own, it represents a solid rectangular section. The SUBTYPE `section_profile_rectangle_hollow` is instanced to represent a rectangular hollow section.

EXPRESS specification:

```

*)
ENTITY section_profile_rectangle
SUPERTYPE OF (section_profile_rectangle_hollow)
SUBTYPE OF (section_profile_simple);
    overall_depth : positive_length_measure_with_unit;
    overall_width : OPTIONAL positive_length_measure_with_unit;
    external_fillet_radius : OPTIONAL positive_length_measure_with_unit;
DERIVE
    overall_depth_value : REAL := overall_depth.value_component;
    overall_width_value : REAL := NVL(overall_width.value_component,
        overall_depth.value_component);
    external_fillet_value : REAL := NVL(external_fillet_radius.value_component, 0.0);
WHERE
    WRS15 : external_fillet_value < (overall_depth_value/2);
END_ENTITY;
(*

```

Attribute definitions:

overall_depth

Declares the first instance of `positive_length_measure_with_unit` associated with the `section_profile_rectangle`, which provides the numerical value with an appropriate unit of the overall depth of the rectangular section. This linear dimension is measured parallel to the minor (z-z) axis.

overall_width

Declares the second instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_rectangle` to provide the numerical value with an appropriate unit of the overall width of the rectangular section. This linear dimension is measured parallel to the major (y-y) axis. A NULL value would indicate that the entity instance is representing a square section.

external_fillet_radius

Declares the third instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_rectangle` to provide the numerical value with an appropriate unit of the external radius of the fillet (or corner) of the rectangular section. All four corners of the rectangular section profile are assumed to have equal radii. A NULL value would indicate that the entity instance is representing a section with square corners.

overall_depth_value

This attribute derives the real number value of the overall depth of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_depth`. This value is used in the WHERE rule WRS15.

overall_width_value

This attribute derives the real number value of the overall width of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_width`. If the attribute `overall_width` has been assigned a NULL value, this attribute is assigned the value equal to the real number value of the associated with the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_depth`. That is, the instance of `section_profile_rectangle` is assumed to represent a square section, and thus, the width of the section equals the depth.

external_fillet_value

This attribute derives the real number value of the corner fillet radius of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `external_fillet_radius`. This value is used in the WHERE rule WRS15. If the attribute `external_fillet_radius` has been assigned a NULL value, this attribute is assigned the value of zero. That is, the instance of `section_profile_rectangle` is assumed to represent a square-cornered section, and thus, the corner radius of the section equals zero.

Formal propositions:*WRS15*

The real number value derived by the attribute `external_fillet_value` shall be less than half the real number value derived by the attribute `overall_depth_value`. That is, the corner radius must be less than half the overall depth of the section. In other words, the rectangular section must not be round. It is assumed that these two parameters are measured using the same units.

Informal propositions:

The local axes of the section's coordinate system are defined as shown in Figure 14.18 for a rectangular hollow section. The origin of this coordinate system is located at the cardinal point (inherited from the SUPERTYPE `section_profile`). Declaring this section to be mirrored would have no effect on the attribute definitions, as the section is symmetrical. (See also the definition of the entity `section_profile`.)

The attributes `overall_depth`, `overall_width`, and `external_fillet_radius` are all measured using the same unit (e.g. mm).

The derived attributes `overall_depth_value`, `overall_width_value` and `external_fillet_value` will not appear if the entity instance is encoded in a STEP Part 21 file.

Notes:

Known as `RECTANGLE_SECT` in CIS/1.

See diagram 66 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

14.3.15 section_profile_rectangle_hollow

Entity definition:

A type of `section_profile_rectangle` that provides the additional defining parameters of a rectangular hollow section. (See Figure 14.18.)

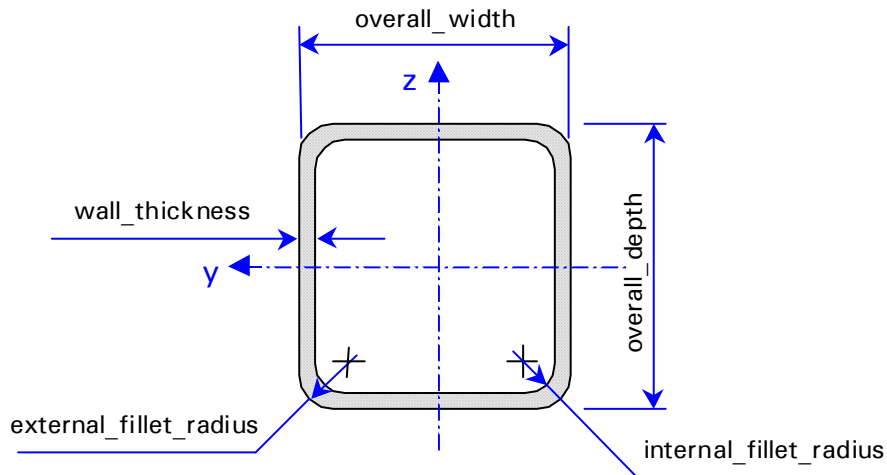


Figure 14.18 Example of a `section_profile_rectangle_hollow`

EXPRESS specification:

```

*)
ENTITY section_profile_rectangle_hollow
SUBTYPE OF (section_profile_rectangle);
    wall_thickness : positive_length_measure_with_unit;
    internal_fillet_radius : OPTIONAL positive_length_measure_with_unit;
DERIVE
    wall_thickness_value : REAL := wall_thickness.value_component;
    internal_radius_value : REAL := NVL(internal_fillet_radius.value_component, 0.0);
WHERE
    WRS16 : wall_thickness_value <
        ((SELF\section_profile_rectangle.overall_depth_value)/2);
    WRS17 : internal_radius_value <
        ((SELF\section_profile_rectangle.overall_depth_value)/2);
END_ENTITY;
(*
  
```

Attribute definitions:

wall_thickness

Declares the first instance of `positive_length_measure_with_unit` associated with the `section_profile_rectangle_hollow`, which provides the numerical value with an appropriate unit of the thickness of the wall of the rectangular hollow section.

internal_fillet_radius

Declares the second instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_rectangle_hollow` to provide the numerical value with an appropriate unit of the internal radius of the fillet (or internal corner) of the rectangular hollow section. All four corners of the rectangular section profile are assumed to have

equal radii. A NULL value would indicate that the entity instance is representing a section with square internal corners.

wall_thickness_value

This attribute derives the real number value of the wall thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `wall_thickness`. This value is used in the WHERE rule WRS16.

internal_radius_value

This attribute derives the real number value of the internal corner radius of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `internal_fillet_radius`. This value is used in the WHERE rule WRS17. If the attribute `internal_fillet_radius` has been assigned a NULL value, this attribute is assigned a value of zero. That is, the instance of `section_profile_rectangle_hollow` is assumed to represent a section whose internal corners are square.

Formal propositions:

DERIVE

The derived attributes `wall_thickness_value` and `internal_radius_value` will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS16

The real number value derived by the attribute `wall_thickness_value` shall be less than half the real number value derived by the attribute `overall_depth_value` (inherited from the SUPERTYPE `section_profile_rectangle`). That is, the wall thickness must be less than half the overall depth of the section. In other words, the hollow section must not be solid. It is assumed that these two parameters are measured using the same units.

WRS17

The real number value derived by the attribute `internal_radius_value` shall be less than half the real number value derived by the attribute `overall_depth_value` (inherited from the SUPERTYPE `section_profile_rectangle`). That is, the internal corner radius must be less than half the overall depth of the section. In other words, the hollow section must not have a circular centre. It is assumed that these two parameters are measured using the same units.

The local axes of the section's coordinate system are defined as shown in Figure 14.18. The origin of this coordinate system is located at the cardinal point (inherited from the SUPERTYPE `section_profile`). Declaring this section to be mirrored would have no effect on the attribute definitions, as the section is symmetrical. (See also the definition of the entity `section_profile`.)

The attributes `wall_thickness` and `internal_fillet_radius` are measured using the same unit (e.g. mm).

Notes:

New for CIS/2; addressed by `RECTANGLE_SECT` in CIS/1.

See diagram 66 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.16 section_profile_simple

Entity definition:

A type of section_profile whose geometry is defined by a number of parameters, rather than through the use of explicit geometry constructs. The simple section profile can be categorized (through its SUBTYPEs) by its shape to represent I, T, channel, angle, circle, and rectangular section profiles.

EXPRESS specification:

```
*)
ENTITY section_profile_simple
ABSTRACT SUPERTYPE OF (ONEOF
    (section_profile_i_type,
    section_profile_t_type,
    section_profile_channel,
    section_profile_angle,
    section_profile_circle,
    section_profile_rectangle))
SUBTYPE OF (section_profile);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE section_profile. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity section_profile, and to collect together the SUBTYPEs under a common category.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Notes:

New for CIS/2; addressed by SECTION_PROFILE in CIS/1.

See diagram 65 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.17 section_profile_t_type

Entity definition:

A type of simple section profile that provides the defining parameters of a T section. Both the flange and the web may be specified as being tapered. The T section is defined such that the flange is at the top of the web (where z has its maximum value in the local coordinate system), has a flat top (i.e. parallel to the y-axis) and is centred on the web. (See Figure 14.19.) The web is defined as having a flat bottom (i.e. parallel to the y-axis), with no fillets.

EXPRESS specification:

*)

ENTITY section_profile_t_type

SUBTYPE OF (section_profile_simple);

overall_depth : positive_length_measure_with_unit;
 flange_width : positive_length_measure_with_unit;
 flange_thickness : positive_length_measure_with_unit;
 web_thickness : positive_length_measure_with_unit;
 root_radius : OPTIONAL positive_length_measure_with_unit;
 flange_slope : OPTIONAL ratio_measure_with_unit;
 web_slope : OPTIONAL ratio_measure_with_unit;
 edge_radius : OPTIONAL positive_length_measure_with_unit;

DERIVE

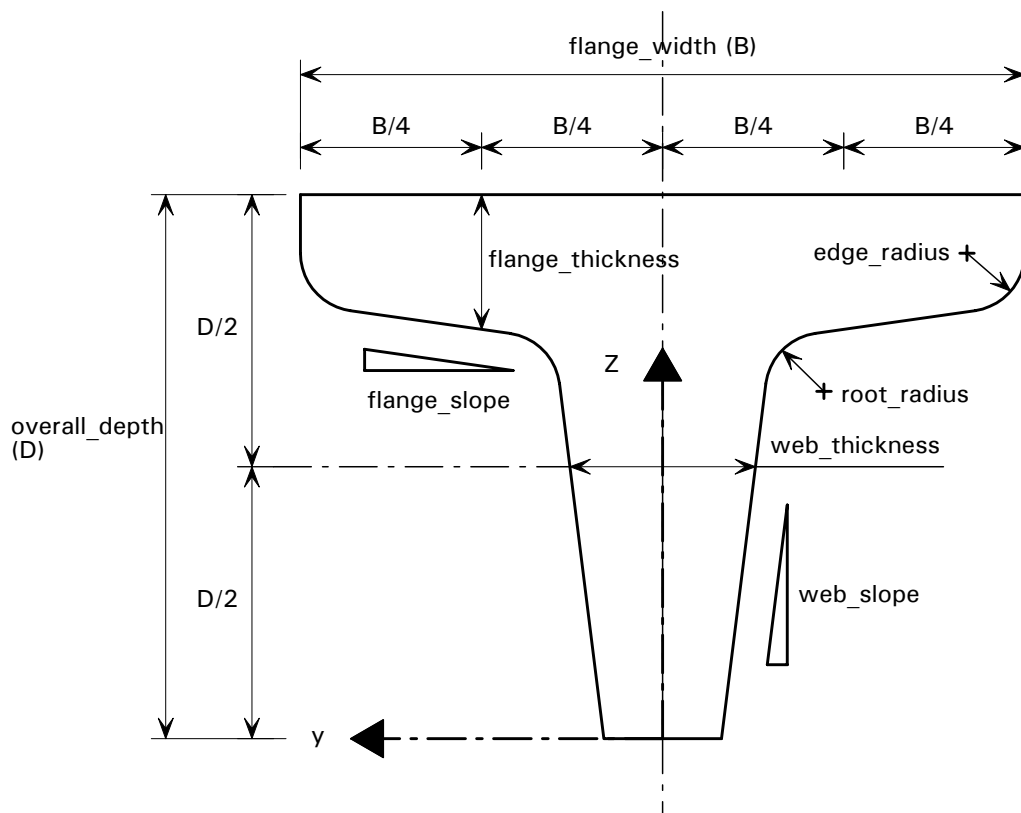
overall_depth_value : REAL := overall_depth.value_component;
 flange_width_value : REAL := flange_width.value_component;
 flange_thickness_value : REAL := flange_thickness.value_component;
 web_thickness_value : REAL := web_thickness.value_component;

WHERE

WRS18 : flange_thickness_value < overall_depth_value;
 WRS19 : web_thickness_value < flange_width_value;

END_ENTITY;

(*)

**Figure 14.19** Definition of a section_profile_t_type

Attribute definitions:***overall_depth***

Declares the first instance of `positive_length_measure_with_unit` associated with the `section_profile_t_type`, which provides the numerical value with an appropriate unit of the overall depth of the T section. This linear dimension is measured parallel to the minor (z-z) axis.

flange_width

Declares the second instance of `positive_length_measure_with_unit` associated with the `section_profile_t_type`, which provides the numerical value with an appropriate unit of the overall flange width of the T section. This linear dimension is measured parallel to the major (y-y) axis.

flange_thickness

Declares the third instance of `positive_length_measure_with_unit` associated with the `section_profile_t_type`, which provides the numerical value with an appropriate unit of the flange thickness of the T section. If the flanges are tapered, the flange thickness is measured at a specific position on the flanges; i.e. midway along the external dimension of the flange outstand. (See Figure 14.19.)

web_thickness

Declares the fourth instance of `positive_length_measure_with_unit` associated with the `section_profile_t_type`, which provides the numerical value with an appropriate unit of the web thickness of the T section. If the web is tapered, the web thickness is measured at a specific position on the web; i.e. at mid-depth of the T section. (See Figure 14.19.)

root_radius

Declares the fifth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_t_type` to provide the numerical value with an appropriate unit of the fillet radius of the 'root' between the web and the flange of the T section. The root radii on either side of the web are defined as being equal.

flange_slope

Declares the first instance of `ratio_measure_with_unit` that may be associated with the `section_profile_t_type` to provide the numerical value with an appropriate unit of the slope (in cross section) of the tapered flange of the T section. This attribute is populated if the flange is tapered such that its underside (web side) slopes. The topside of the flange is defined as being flat (i.e. parallel to the y-axis). The slope of the flange is defined as being the same on both sides of the web.

web_slope

Declares the second instance of `ratio_measure_with_unit` that may be associated with the `section_profile_t_type` to provide the numerical value with an appropriate unit of the slope (in cross section) of the tapered web of the T section. This attribute is populated if the web is tapered symmetrically about its centreline. The slope of the web is given on one side and is defined as being the same on both sides of the web.

edge_radius

Declares the sixth instance of `positive_length_measure_with_unit` that may be associated with the `section_profile_t_type` to provide the numerical value with an appropriate unit of the fillet radius of the 'toe' of the flange of the T section. The edge radii of both toes

are defined as being equal. The top side of the T section is defined as having square edges.

overall_depth_value

This attribute derives the real number value of the overall depth of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `overall_depth`. This value is used in the WHERE rule WRS18.

flange_width_value

This attribute derives the real number value of the overall width of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `flange_width`. This value is used in the WHERE rule WRS19.

flange_thickness_value

This attribute derives the real number value of the flange thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `flange_thickness`. This value is used in the WHERE rule WRS18.

web_thickness_value

This attribute derives the real number value of the web thickness of the section from the instance of `positive_length_measure_with_unit` referenced by the attribute `web_thickness`. This value is used in the WHERE rule WRS19.

Formal propositions:

DERIVE

The derived attributes `overall_depth_value`, `flange_width_value`, `flange_thickness_value`, and `web_thickness_value` will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS18

The real number value derived by the attribute `flange_thickness_value` shall be less than the real number value derived by the attribute `overall_depth_value`. That is, the flange thickness must be less than the overall depth of the section. In other words, the T section must not be ‘all flange’. It is assumed that these two parameters are measured using the same units.

WRS19

The real number value derived by the attribute `web_thickness_value` shall be less than the real number value derived by the attribute `flange_width_value`. That is, the web thickness must be less than the overall width of the section. In other words, the T section must not be ‘all web’. It is assumed that these two parameters are measured using the same units.

Informal propositions:

The local axes of the section’s coordinate system are defined as shown in Figure 14.19. The origin of this coordinate system is located at the cardinal point of the T section (inherited from the SUPERTYPE `section_profile`). Declaring this section to be mirrored would have no effect on the attribute definitions, as the section is symmetrical. (See also the definition of the entity `section_profile`.)

An ‘offset T’ (i.e. one with unequal flange outstands) should not be defined using this entity.

The attributes `overall_depth`, `flange_width`, `flange_thickness` and `web_thickness` are all measured using the same unit (e.g. mm).

Notes:

Known as `T_TYPE_SECT` in CIS/1.

See diagram 66 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

14.3.18 section_properties

Entity definition:

This entity provides a set of defining parameters of the analytical characteristics of the `section_profile` related to the geometry. This entity is instanced to represent the properties of a symmetric section profile, while the SUBTYPE `section_properties_asymmetric` is instanced to represent the additional properties particular to an asymmetric section profile. The `section_properties` must be related to a particular instance of `section_profile`. All the other attributes are optional, although it is expected that one or more of the parameters will be assigned values as required.

Some of the section properties are dependent upon the definition of the analytical axes of the section profile. The analytical axes are based on an origin located at the geometric centroid of the section. The analytical y and z-axes are distinct and separate from the locating axes used to locate the section profile with respect to the part or element referencing the section profile. The locating axes will coincide with the analytical axes only when the cardinal point of the section profile is assigned a value of 10. The amount of the offset in horizontal and vertical directions is given by the second attribute of this entity.

EXPRESS specification:

*)

ENTITY `section_properties`

SUPERTYPE OF (`section_properties_asymmetric`);

```

    profile : section_profile;
    origin_offset : ARRAY [1:2] OF OPTIONAL length_measure_with_unit;
    torsional_constant_lx : OPTIONAL inertia_measure_with_unit;
    inertia_moment_ly : OPTIONAL inertia_measure_with_unit;
    inertia_moment_lz : OPTIONAL inertia_measure_with_unit;
    section_area_Ax : OPTIONAL area_measure_with_unit;
    shear_area_Asy : OPTIONAL area_measure_with_unit;
    shear_area_Asz : OPTIONAL area_measure_with_unit;
    shear_deformation_area_Ay : OPTIONAL area_measure_with_unit;
    shear_deformation_area_Az : OPTIONAL area_measure_with_unit;
    surface_per_length : OPTIONAL positive_length_measure_with_unit;
    radius_of_gyration_ry : OPTIONAL positive_length_measure_with_unit;
    radius_of_gyration_rz : OPTIONAL positive_length_measure_with_unit;
    plastic_modulus_Sy : OPTIONAL modulus_measure_with_unit;
    plastic_modulus_Sz : OPTIONAL modulus_measure_with_unit;
    warping_constant : OPTIONAL derived_measure_with_unit;
    torsional_index : OPTIONAL REAL;
    buckling_parameter : OPTIONAL REAL;
```



```

    nominal_mass : OPTIONAL mass_per_length_measure_with_unit;
    actual_mass : OPTIONAL mass_per_length_measure_with_unit;
DERIVE
    y_offset : REAL := NVL(origin_offset[1].value_component, 0.0);
    z_offset : REAL := NVL(origin_offset[2].value_component, 0.0);
WHERE
    WRS39 : NOT( (profile.cardinal_point = 10) AND (y_offset <> 0.0) );
    WRS40 : NOT( (profile.cardinal_point = 10) AND (z_offset <> 0.0) );
END_ENTITY;
(*)

```

Attribute definitions:

profile

Declares the instance of section_profile associated with the section_properties, for which the properties are defined. There must be one (and only one) instance of section_profile referenced by each instance of section_properties. However, each instance of section_profile may be referenced by zero, one or many instances of section_properties.

origin_offset

Declares the ARRAY of up to two instances of length_measure_with_unit that may be associated with the section_properties to provide the numerical values with appropriate units of the distance from the geometric centroid to the origin of the locating axes for the associated section profile.

The ARRAY data type implies that the order of the members in the aggregation is important. Indeed the ARRAY can be searched by index from 1 to 2. The ARRAY is populated as follows:

1. horizontal distance (i.e. in the local y-axis direction) from the geometric centroid to the cardinal point,
2. vertical distance (i.e. in the local z-axis direction) from the geometric centroid to the cardinal point.

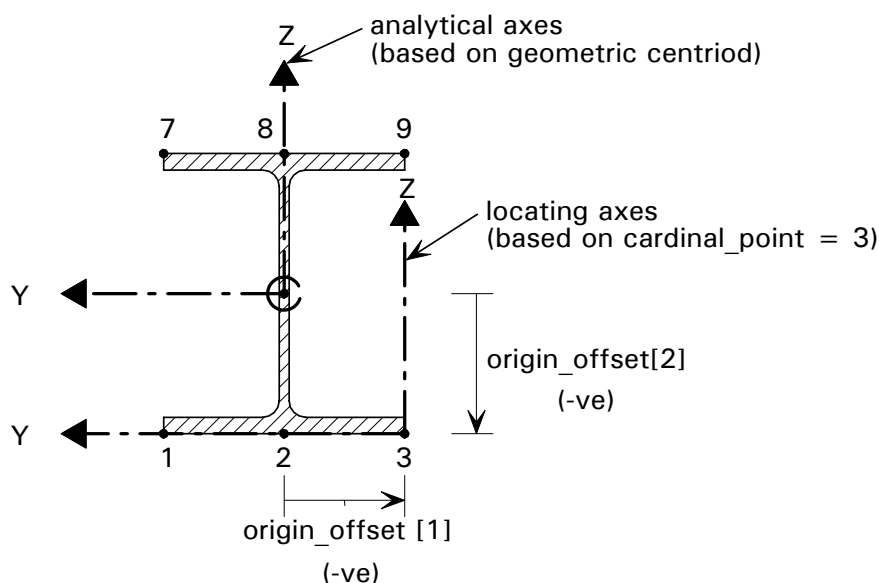


Figure 14.20 *Definition of the origin_offset*

The OPTIONAL keyword allows members of the aggregation to be set to NULL (\$). Thus, there must be 2 members in the aggregation, either member or both members may be set to NULL. Where no values are provided for any of the dimensions, the ARRAY is populated in the physical file as (\$, \$). If either member of the aggregation is set to NULL, the origin offset in that direction is set to zero (0.0). If both are NULL, it is implied that the origin of the analytical axes lies on the cardinal point.

These parameters are functions of the section's geometry and are measured in accordance with the appropriate (national) standard or code of practice. For example, when using EC3 these dimensions would be measured in mm.

torsional_constant_lx

Declares the first instance of inertia_measure_with_unit that may be associated with the section_properties to provide the numerical value with an appropriate unit of the 'polar moment of inertia' (or second moment of area) about the longitudinal axis, of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted I_t and is measured about the analytical x-x axis in cm^4 .

inertia_moment_ly

Declares the second instance of inertia_measure_with_unit that may be associated with the section_properties to provide the numerical value with an appropriate unit of the 'bending moment of inertia' (or second moment of area) about the major axis, of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted I_y and is measured about the analytical y-y axis in cm^4 .

inertia_moment_lz

Declares the third instance of inertia_measure_with_unit that may be associated with the section_properties to provide the numerical value with an appropriate unit of the 'bending moment of inertia' (or second moment of area) about the minor axis, of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted I_z and is measured about the analytical z-z axis in cm^4 .

section_area_Ax

Declares the first instance of area_measure_with_unit that may be associated with the section_properties to provide the numerical value with an appropriate unit of the gross cross sectional area of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted A and is measured in cm^2 .

shear_area_Asy

Declares the second instance of area_measure_with_unit that may be associated with the section_properties to provide the numerical value with an appropriate unit of the effective shear area in the horizontal plane of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted A_v and is measured in the y-y plane in cm^2 . For a welded I section, for example, when the load is applied parallel to the flanges, the effective shear area = the sectional area of the flanges. This parameter is used when calculating the capacity of the section to resist shear force.

shear_area_Asz

Declares the third instance of `area_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the effective shear area in the vertical plane of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted A_v and is measured in the z-z plane in cm². For a welded I section, for example, when the load is applied parallel to the flanges, the effective shear area = the sectional area of the web. This parameter is used when calculating the capacity of the section to resist shear force.

shear_deformation_area_Ay

Declares the third instance of `area_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the effective shear area in the horizontal plane of the associated section profile when shear deformation is important. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted A_v and is measured in the y-y plane in cm². This parameter is used when calculating the capacity of the section to resist shear deformation.

shear_deformation_area_Az

Declares the fourth instance of `area_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the effective shear area in the vertical plane of the associated section profile when shear deformation is important. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted A_v and is measured in the z-z plane in cm². This parameter is used when calculating the capacity of the section to resist shear deformation.

surface_per_length

Declares the first instance of `positive_length_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the gross surface area per unit length of the associated section profile (which resolves to a length measurement). This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is measured in m²/m.

radius_of_gyration_ry

Declares the second instance of `length_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the radius of gyration about the major axis of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted i_y and is measured about the analytical y-y axis in mm.

radius_of_gyration_rz

Declares the third instance of `length_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the radius of gyration about the minor axis of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted i_z and is measured about the analytical z-z axis in mm.

plastic_modulus_Sy

Declares the first instance of `modulus_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the plastic section modulus about the major axis of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted $W_{pl,y}$ and is measured about the analytical y-y axis in cm^3 .

plastic_modulus_Sz

Declares the second instance of `modulus_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the plastic section modulus about the minor axis of the associated section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted $W_{pl,z}$ and is measured about the analytical z-z axis in cm^3 .

warping_constant

Declares an instance of `derived_measure_with_unit` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the warping constant of the associated section profile. This parameter is a function of the geometry used in buckling calculations and is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted I_w and is measured in cm^6 .

torsional_index

The dimensionless numerical value of the torsional index of the associated section profile. This parameter is a function of the geometry used in buckling calculations and is defined in accordance with the declared (national) standard or code of practice.

buckling_parameter

The dimensionless numerical value of the buckling parameter of the associated section profile. This parameter is a function of the geometry used in buckling calculations and is defined in accordance with the declared (national) standard or code of practice.

nominal_mass

Declares the first instance of `mass_per_length_measure` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the specified nominal mass per unit length of the associated section profile. This parameter is a function of the shape and the material associated with the section profile. However, for many purposes it is assumed that the mass density of steel is constant which makes this parameter a function of geometry alone. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is measured in kg/m . This parameter is often used to specify the section profile, but may be used in calculations that derive an approximate mass of a group of structural members.

actual_mass

Declares the second instance of `mass_per_length_measure` that may be associated with the `section_properties` to provide the numerical value with an appropriate unit of the (measured or specified) actual mass per unit length of the associated section profile. This parameter is a function of the shape and the material associated with the section profile.

However, for many purposes it is assumed that the mass density of steel is constant which makes this parameter a function of geometry alone. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is measured in kg/m. This parameter may be used in calculations that derive more accurate values for the mass of a group of structural members than would be derived from the nominal mass.

y_offset

This attribute derives the real number value of the horizontal distance in the direction of the local y-axis from the geometric centroid to the cardinal point of the section profile. The value is derived from the value_component of the instance of length_measure_with_unit referenced by the first member of the aggregation of attribute origin_offset. If no value is provided, (i.e. if the first member of the aggregation is set to NULL) then the y_offset is assigned the value of zero (0.0).

z_offset

This attribute derives the real number value of the vertical distance in the direction of the local z-axis from the geometric centroid to the cardinal point of the section profile. The value is derived from the value_component of the instance of length_measure_with_unit referenced by the second member of the aggregation of attribute origin_offset. If no value is provided, (i.e. if the second member of the aggregation is set to NULL) then the z_offset is assigned the value of zero (0.0).

Formal propositions:

DERIVE

The derived attributes y_offset and z_offset will not appear if the entity instance is encoded in a STEP Part 21 file.

WRS39

The real number value of the derived attribute y_offset shall not be non-zero if the cardinal_point of the instance of section_profile referenced by the attribute profile has been assigned a value of 10. In other words, if the geometric centroid has been assigned as the cardinal point of the section profile, then the horizontal offset shall equal zero.

WRS40

The real number value of the derived attribute z_offset shall not be non-zero if the cardinal_point of the instance of section_profile referenced by the attribute profile has been assigned a value of 10. In other words, if the geometric centroid has been assigned as the cardinal point of the section profile, then the vertical offset shall equal zero.

Informal propositions:

Although all but one of the attributes of this entity are optional, it is expected that one or more of the defining parameters will be assigned values as required.

An instance of section properties will normally only be required to be exchanged where the section_profile is passed explicitly by attribute value rather than being passed implicitly by reference using (predefined) item identifiers.

Notes:

Known as SECTION_PROPERTIES in CIS/1.

See diagram 68 of the EXPRESS-G diagrams in Appendix B.

The relationship with `section_properties` and `section_profile` is such that an instance of `section_properties` cannot exist without a corresponding instance of `section_profile` (i.e. it is existence dependent). However, an instance of `section_profile` can exist without an instance of `section_properties` (i.e. it is existence independent). Further, any instance of `section_profile` can be referenced by any number of instances of `section_profile`. This is different from CIS/1.

The attributes referring to ‘inertia’ in this entity are functions of the section geometry and should not be confused with the inertial force of inertial mass properties (which are related to the material properties) used in dynamic analysis.

Unchanged for 2nd Edition.

14.3.19 `section_properties_asymmetric`

Entity definition:

A type of `section_properties` that provides an additional set of defining parameters of the static characteristics related to the geometry of the asymmetric section profile. For this type of asymmetric section, the shear centre is offset from the neutral axis. The amount of the offset in horizontal and vertical directions is given by the first attribute of this entity. Further, the section profile has major and minor principal axes (defined as *u-u* and *v-v*) that do not lie on the analytical horizontal and vertical axes (defined as *y-y* and *z-z*). Thus, four analytical axes that may be used to define elastic section moduli. Because the values of these elastic section moduli are different on either side of each axis, the last attribute of this entity provides an array of eight section moduli, which must be populated in a given sequence. (See Figure 14.21.)

The section properties are defined for the instance of `section_profile` referenced by the attribute `profile` (inherited from the SUPERTYPE). Some parameters will vary depending upon whether that section is mirrored or unmirrored.

EXPRESS specification:

*)

ENTITY `section_properties_asymmetric`

SUBTYPE OF (`section_properties`);

`neutral_axis_shear_centre` : ARRAY [1:2] OF OPTIONAL `length_measure_with_unit`;

`theta_angle_z_axis_v_axis` : OPTIONAL `plane_angle_measure_with_unit`;

`inertia_moment_lu` : OPTIONAL `inertia_measure_with_unit`;

`inertia_moment_lv` : OPTIONAL `inertia_measure_with_unit`;

`radius_of_gyration_ru` : OPTIONAL `positive_length_measure_with_unit`;

`radius_of_gyration_rv` : OPTIONAL `positive_length_measure_with_unit`;

`section_moduli` : ARRAY [1:8] OF OPTIONAL `modulus_measure_with_unit`;

END_ENTITY;

(*

Attribute definitions:

neutral_axis_shear_centre

Declares the ARRAY of up to two instances of `length_measure_with_unit` that may be associated with the `section_properties_asymmetric` to provide the numerical values with appropriate units of the distance from the elastic neutral axis to the shear centre for the associated asymmetric section profile.

The ARRAY data type implies that the order of the members in the aggregation are important. Indeed the ARRAY can be searched by index from 1 to 2. The ARRAY is populated as follows:

1. horizontal distance (i.e. in the local y-axis direction) from the elastic neutral axis to the shear centre,
2. vertical distance (i.e. in the local z-axis direction) from the elastic neutral axis to the shear centre.

The OPTIONAL keyword allows members of the aggregation to be set to NULL (\$). Thus, there must be 2 members in the aggregation, either member or both members may be set to NULL. Where no values are provided for any of the dimensions, the ARRAY is populated in the physical file as (\$, \$).

These parameters are functions of the section's geometry and are measured in accordance with the appropriate (national) standard or code of practice. For example, when using EC3 these dimensions would be measured in mm.

theta_angle_z_axis_v_axis

Declares the instance of plane_angle_measure_with_unit that may be associated with the section_properties_asymmetric to provide the numerical values with appropriate units of the plane angle of rotation, measured clockwise, from the local analytical positive z-axis to the positive minor principal (v) axis for the associated asymmetric section profile.

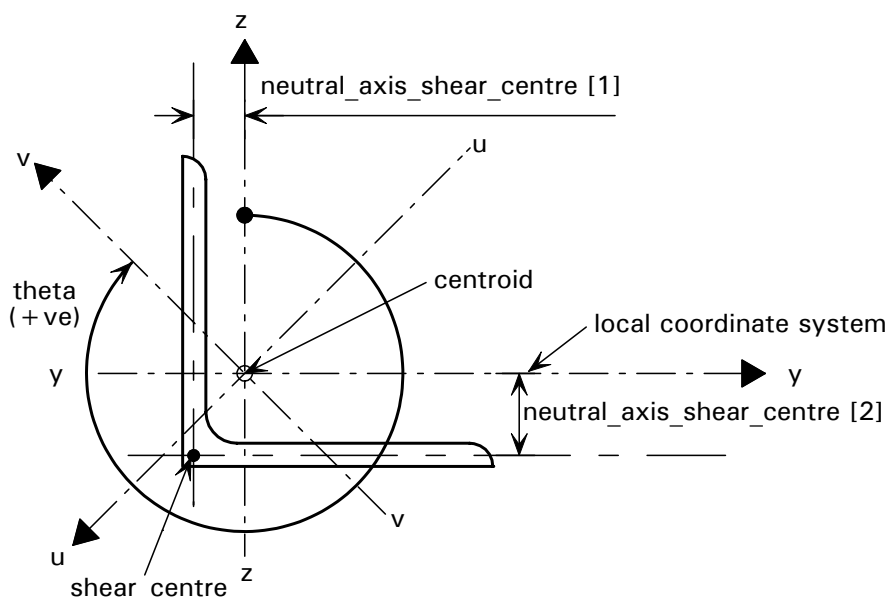


Figure 14.21 Example of the principal axes used in *section_properties_asymmetric* *inertia_moment_lu*

Declares the first instance of inertia_measure_with_unit that may be associated with the section_properties_asymmetric to provide the numerical value with an appropriate unit of the 'bending moment of inertia' (or second moment of area) about the major principal (u) axis, of the associated asymmetric section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted $I_{el,u}$ and is measured about the u-u axis in cm^4 .

inertia_moment_lv

Declares the second instance of *inertia_measure_with_unit* that may be associated with the *section_properties_asymmetric* to provide the numerical value with an appropriate unit of the ‘bending moment of inertia’ (or second moment of area) about the minor principal (v) axis, of the associated asymmetric section profile. This parameter is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted $I_{el,v}$ and is measured about the v-v axis in cm^4 .

radius_of_gyration_ru

Declares the first instance of *positive_length_measure_with_unit* that may be associated with the *section_properties_asymmetric* to provide the numerical values with appropriate unit of the radius of gyration about the major principal (u) axis for the associated asymmetric section profile. This parameter is a function of the section’s geometry and is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted $i_{el,u}$ and is measured about the u-u axis in mm.

radius_of_gyration_rv

Declares the second instance of *positive_length_measure_with_unit* that may be associated with the *section_properties_asymmetric* to provide the numerical values with appropriate unit of the radius of gyration about the minor principal (v) axis for the associated asymmetric section profile. This parameter is a function of the section’s geometry and is measured in accordance with the appropriate (national) standard or code of practice. For example, in EC3, this parameter is denoted $i_{el,v}$ and is measured about the v-v axis in mm.

section_modulii

Declares the array of up to eight instances of *modulus_measure_with_unit* that may be associated with the *section_properties_asymmetric* to provide the numerical values with appropriate units of the elastic section modulii on each side of the four axes for the associated asymmetric section profile. The ARRAY data type implies that the order is important. Indeed the ARRAY can be searched by index from 1 to 8. The ARRAY is populated as follows:

1. elastic section modulus about the y-axis, for the top fibre (z-increasing)
(Denoted $W_{el,y,1}$ in EC3)
2. elastic section modulus about the y-axis, for the bottom fibre (z-decreasing)
(Denoted $W_{el,y,2}$ in EC3)
3. elastic section modulus about the z-axis, for the leftmost fibre (y-decreasing)
(Denoted $W_{el,z,1}$ in EC3)
4. elastic section modulus about the z-axis, for the rightmost fibre (y-increasing)
(Denoted $W_{el,z,2}$ in EC3)
5. elastic section modulus about the u-axis, for the most v-positive fibre (Denoted $W_{el,u,1}$ in EC3)
6. elastic section modulus about the v-axis, for the most u-positive fibre (Denoted $W_{el,v,1}$ in EC3)

7. elastic section modulus about the u-axis, for the most v-negative fibre (Denoted $W_{el.u.2}$ in EC3)
8. elastic section modulus about the v-axis, for the u-negative fibre (Denoted $W_{el.v.2}$ in EC3)

The OPTIONAL keyword allows members of the aggregation to be set to NULL (\$). Thus, there must be 8 members in the aggregation, but some (or even all) may be set to NULL. Where no values are provided for any of the section moduli, the ARRAY is populated in the physical file as (\$, \$, \$, \$, \$, \$, \$, \$).

These parameters are functions of the section's geometry and are measured in accordance with the appropriate (national) standard or code of practice. For example, when using EC3 the section moduli would be expressed in cm^3 . It should be noted that the elastic section modulus is **not** the same as the modulus of elasticity, which is a property of the material.

Informal propositions:

Although most of the attributes of this entity are optional, it is expected that one or more of the defining parameters will be assigned values as required.

Notes:

New for CIS/2; addressed by SECTION_PROPERTIES in CIS/1.

The attribute section_moduli (with its data type defined as an ARRAY [1:8] OF OPTIONAL modulus_measure_with_unit) combines in sequence the following attributes from the CIS/1 entity SECTION_PROPERTIES:

```

section_modulus_Zy_top : OPTIONAL MODULUS_MEASURE;
section_modulus_Zy_bottom : OPTIONAL MODULUS_MEASURE;
section_modulus_Zz_left : OPTIONAL MODULUS_MEASURE;
section_modulus_Zz_right : OPTIONAL MODULUS_MEASURE;
section_modulus_Zu_v_pos : OPTIONAL MODULUS_MEASURE;
section_modulus_Zv_u_pos : OPTIONAL MODULUS_MEASURE;
section_modulus_Zu_v_neg : OPTIONAL MODULUS_MEASURE;
section_modulus_Zv_u_neg : OPTIONAL MODULUS_MEASURE;
```

See diagram 68 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition

14.3.20 shape_representation_with_units

Entity definition:

A type of shape_representation where the context has to be defined such that it provides a set of units to be used with the definition of the geometric constructs.

EXPRESS specification:

*)

ENTITY shape_representation_with_units

SUBTYPE OF (shape_representation);

WHERE

```

WRS20 : ('STRUCTURAL_FRAME_SCHEMA.
GEOMETRIC_REPRESENTATION_CONTEXT' IN TYPE OF
(SELF\representation.context_of_items)) AND ('STRUCTURAL_FRAME_SCHEMA.
GLOBAL_UNIT_ASSIGNED_CONTEXT' IN TYPEOF
(SELF\representation.context_of_items));
```

```

WRS21 : SIZEOF(QUERY(temp <* SELF\representation.items |
('STRUCTURAL_FRAME_SCHEMA.
GEOMETRIC_REPRESENTATION_ITEM') IN TYPE OF(temp))) > 0;
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its STEP Part 41 SUPERTYPE shape_representation. The purpose of this entity is to constrain (or specialize) the SUPERTYPE entity shape_representation when it is used in LPM/6.

Formal propositions:

WRS20

The context for the set of instances of representation_item associated with the shape_representation_with_units shall be of the type geometric_representation_context **and** global_unit_assigned_context. In other words, the items of the representation shall be placed in a geometric context and be provided with a set of global units. It is implied that the set of units associated with the global_unit_assigned_context provide the units for the physical dimensions the associated shape (defined by the geometric constructs).

WRS21

The size of the set of instances of representation_item associated with the shape_representation_with_units that are of the type geometric_representation_item shall be greater than one. In other words, there must be at least one instance of geometric_representation_item referenced by this instance of shape_representation_with_units.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been interpreted for use in LPM/6.

Notes

New for CIS/2.

The entities shape_representation and global_unit_assigned_context are defined in STEP Part 41. The entities geometric_representation_item and geometric_representation_context are defined in STEP Part 42. The entities representation, representation_context and representation_item are defined in STEP Part 43.

See Diagram 12 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition. However, the definition of representation has changed.

15 LPM/6 UNITS & MEASURES

15.1 Units & Measures concepts and assumptions

The use of units and measures and the definition of unit systems have significant implications for implementation. Software developers are recommended to read the Conformance Requirements and the Implementation Guide before writing their translator coding.

15.1.1 Conceptual overview

LPM/6 places no restrictions on the use of particular unit systems, allowing applications to define units as required using the constructs from STEP. LPM/6 follows the general approach, terminology and constructs of STEP.

Definitions:

measure

a value of a real or abstract quantity

unit

a scale used to measure that quantity

dimensionality

the dimensionality of a unit against the fundamental dimensions of length, time, mass etc.

Each measure must have associated with it a unit, and this must be of the required dimensionality. In practice, each unit (and its dimensionality) will be declared once and then used again and again.

In CIS/2, there are two distinct ways in which units are specified for the measurement of both abstract and real quantities; i.e. globally or specifically. In both cases, however, the unit must be specified for each and every measurement that is represented in LPM/6. This, of course, does not mean that the unit has to be declared again and again. It merely has to be stated once, and referred to again and again. The intention of this is to avoid ambiguities that arise from ‘unit system flags’ which imply the units from the unit system declared. A second very important advantage of the CIS/2 approach to units and measures is that it allows qualifiers to be assigned to the measurement. The qualifiers may be precision, qualitative, or uncertainty qualifiers. These constructs have been brought in from STEP Part 45, and are not described in detail here. CIS/2 uses the uncertainty qualifiers to represent tolerances on measurements.

15.1.2 The use of STEP Part 41 Units & Measures

CIS/2 takes full advantage of the EXPRESS constructs documented in clause 1.14 of STEP Part 41 to model the unit systems required by the LPM. This is one area where the CIS follows the ‘STEP interpreted route’ for the LPM. That is, the generic constructs from the STEP measure_schema are specialized for use in LPM/6. For example, STEP Part 41 provides the following constructs:

Extract from EXPRESS schema

```
ENTITY measure_with_unit
  SUPERTYPE OF (ONEOF
    (length_measure_with_unit,
     mass_measure_with_unit,
```

```

    time_measure_with_unit,
    thermodynamic_temperature_measure_with_unit,
    plane_angle_measure_with_unit,
    solid_angle_measure_with_unit,
    area_measure_with_unit,
    volume_measure_with_unit,
    ratio_measure_with_unit,
    force_measure_with_unit,
    frequency_measure_with_unit,
    pressure_measure_with_unit,
    positive_length_measure_with_unit,
    derived_measure_with_unit,
    uncertainty_measure_with_unit));
value_component : measure_value;
unit_component : unit;
WHERE
    WRM25 : valid_units (SELF);
END_ENTITY; -- STEP Part 41 expanded

```

```

ENTITY named_unit
    SUPERTYPE OF ((ONEOF
        (length_unit,
         mass_unit,
         time_unit,
         thermodynamic_temperature_unit,
         plane_angle_unit,
         solid_angle_unit,
         area_unit,
         volume_unit,
         ratio_unit,
         force_unit,
         frequency_unit,
         pressure_unit)) ANDOR (ONEOF
        (si_unit,
         conversion_based_unit,
         context_dependent_unit)));
    dimensions : dimensional_exponents;
END_ENTITY; -- STEP Part 41

```

```

ENTITY dimensional_exponents;
    length_exponent : REAL;
    mass_exponent : REAL;
    time_exponent : REAL;
    electric_current_exponent : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent : REAL;
    luminous_intensity_exponent : REAL;

```

END_ENTITY; -- STEP Part 41

Which are used in LPM/6 in the following way:

```
ENTITY force_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
WRF17 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN TYPE OF
  (SELF\measure_with_unit.unit_component);
  WRF18 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_MEASURE' IN TYPE OF
  (SELF\measure_with_unit.value_component);
END_ENTITY;
```

```
ENTITY force_unit
SUBTYPE OF (named_unit);
WHERE
  WRF24 : (SELF\named_unit.dimensions.length_exponent = 1.0) AND
  (SELF\named_unit.dimensions.mass_exponent = 1.0) AND
  (SELF\named_unit.dimensions.time_exponent = -2.0) AND
  (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
  (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
  (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
  (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;
```

```
ENTITY moment_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
WRM31 : 'STRUCTURAL_FRAME_SCHEMA.MOMENT_UNIT' IN TYPE OF
  (SELF\measure_with_unit.unit_component);
  WRM32 : 'STRUCTURAL_FRAME_SCHEMA.MOMENT_MEASURE' IN TYPE OF
  (SELF\measure_with_unit.value_component);
END_ENTITY;
```

```
ENTITY moment_unit
SUBTYPE OF (derived_unit);
WHERE
  WRM33 : SIZEOF(SELF\derived_unit.elements) = 2;
  WRM34 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN TYPE OF
  (SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
  (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
  WRM35 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
  (SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
  (SELF\derived_unit.elements[2]\derived_unit_element.exponent = 1.0);
END_ENTITY;
```

These constructs are illustrated in diagrams 14-17 of the EXPRESS-G diagrams in Appendix B. Since named_unit is an ANDOR SUPERTYPE, the attribute that has been given a force_measure_with_unit as a data type, may (but need not) be associated with an SI unit of force.

15.1.3 Globally assigned units

LPM/6 uses the normal STEP method for globally assigning units to certain measurements. These measurements include the STEP constructs for geometry and topology, as well as the CIS/2 constructs for materials. That is, geometry, topology and materials are modelled in the LPM as representations. It should be noted that STEP separates the *definition* of a *product* from its *properties* and their *representation*. In other words, in the world of STEP, a product is defined by a number of definitions; each definition is a collection of properties; and each of these properties may be represented using a number of items.

Since the STEP constructs are not described in detail in the ‘short form’, the ‘top level’ of the STEP EXPRESS constructs is shown below.

Extract from EXPRESS schema

ENTITY representation

SUPERTYPE OF (ONEOF

(shape_representation, material_representation) ANDOR
definitional_representation);

name : label;

items : SET [1:?] OF representation_item;

context_of_items : representation_context;

DERIVE

id : identifier := get_id_value (SELF);

description : text := get_description_value (SELF);

WHERE

WRR51: SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;

WRR52: SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;

END_ENTITY; (* STEP Part 43 expanded (modified in 2nd edition) *)

ENTITY representation_context

SUPERTYPE OF (ONEOF

(geometric_representation_context,
parametric_representation_context,
material_property_context) ANDOR
global_unit_assigned_context ANDOR
global_uncertainty_assigned_context);

context_identifier : identifier;

context_type : text;

INVERSE

representations_in_context : SET [1:?] OF representation FOR context_of_items;

END_ENTITY; (* STEP Part 43 expanded (unchanged in 2nd edition) *)

ENTITY representation_item

SUPERTYPE OF (geometric_representation_item ANDOR

topological_representation_item ANDOR

mapped_item ANDOR

material_representation_item);

```

    name : label;
WHERE
    WRR25 : SIZEOF(using_representations(SELF)) > 0;
END_ENTITY; (* STEP Part 43 expanded (unchanged in 2nd edition) *)

```

The constructs above are illustrated in diagram 12 of the EXPRESS-G diagrams in Appendix B.

As can be seen from the constructs above, a representation requires a number of representation items. For the definition of the shape of a product, these items will be the STEP SUBTYPES of the entities `geometric_representation_item` and `topological_representation_item`. For materials, this will be the CIS/2 SUBTYPE `material_representation_item`. Further, a representation requires a `representation_context`.

For example, when defining the shape of a product, the context will be the STEP SUBTYPE `geometric_representation_context`. Where units are assigned globally, the context can also be the STEP SUBTYPE `global_unit_assigned_context`. Where uncertainty or precision qualifiers are assigned to units the context **can also be** the STEP SUBTYPE `global_uncertainty_assigned_context`. (This is made possible through the ANDOR SUPERTYPE constructs.)

Thus, wherever a measurement has not been given a specific unit as part of the attribute's data type, it will be given a globally assigned unit. This applies to all the SUBTYPES of `representation_item` and is used whenever a measurement is required for a geometric, topological or material parameter.

15.1.4 Specific units

Unlike STEP, LPM/6 does not always demand that a particular parameter be declared with the full path from representation to product (via property and definition). Thus, many of the parameters in LPM/6 cannot be given a representation context and therefore the units for their measurements cannot be globally assigned. To overcome this, many attributes in LPM/6 have the data type `measure_with_unit`. This allows the declaration of a numerical value with a reference to a unit. It also allows qualifiers to be assigned to the measurement. For each different measurement (e.g. length, mass, time) an instance of the appropriate unit (e.g. metres, kilograms, seconds) will be declared (normally only once) in the physical file (or other storage mechanism). The unit will then be referred to many times. Although this does add an extra instance in the physical file for each unit used, it enables the LPM to be a very powerful, yet flexible, means of representing units and measures.

LPM/6 defines units in this way, to allow the measurements of physical quantities to be assigned qualifiers. These qualifiers can be applied to an individual measurement and allow for the uncertainty (tolerance) and precision of the measurement.

When a `named_unit` is instanced in a physical file as both a `force_unit` AND an `si_unit`, it is 'externally mapped' and the instance is encoded in alphabetical order of the entity long names. (See the *Implementation Guide* for further details of encoding of EXPRESS entities in a physical file.)

15.2 Units & Measures type definitions

Many of the types used to represent Units and Measures are brought into LPM/6 through schema interfacing from STEP Part 41, and are therefore, not described in this section. (See the earlier section for the Schema Interface declarations.)

No types have been modified or added in this subject area for the 2nd Edition.

15.2.1 derived_measure

Type definition:

An abstract construct that allows the selection of one of the measures in the SELECT list. That is, it allows a measure to provide the numerical value of a measurement of force per length, inertia, linear acceleration, linear stiffness, linear velocity, mass per length, modulus measure, moment measure, rotational acceleration measure, rotational stiffness measure, rotational velocity measure.

EXPRESS specification:

*)

```
TYPE derived_measure
= SELECT
  (force_per_length_measure,
   inertia_measure,
   linear_acceleration_measure,
   linear_stiffness_measure,
   linear_velocity_measure,
   mass_per_length_measure,
   modulus_measure,
   moment_measure,
   rotational_acceleration_measure,
   rotational_stiffness_measure,
   rotational_velocity_measure);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.2 force_measure

Type definition:

A real number value of a measurement of force. (See also the definition for the entity force_measure_with_unit.)

EXPRESS specification:

*)

```
TYPE force_measure
= REAL;
END_TYPE;
(*
```

Notes

Known as FORCE_MEASURE in CIS/1. Unchanged in 2nd Edition.

15.2.3 force_per_length_measure

Type definition:

A real number value of a measurement of force per unit length. (See also the definition for the entity force_per_length_measure_with_unit.)

EXPRESS specification:

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

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.4 frequency_measure

Type definition:

A real number value of a measurement of frequency. (See also the definition for the entity frequency_measure_with_unit.)

EXPRESS specification:

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

Notes

Known as FREQUENCY_MEASURE in CIS/1. Unchanged in 2nd Edition.

15.2.5 inertia_measure

Type definition:

A real number value of a measurement of inertia (or second moment or area). (See also the definition for the entity inertia_measure_with_unit.)

EXPRESS specification:

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

Notes

Known as INERTIA_MEASURE in CIS/1. Unchanged in 2nd Edition.

15.2.6 linear_acceleration_measure

Type definition:

A real number value of a measurement of linear_acceleration. (See also the definition for the entity linear_acceleration_measure_with_unit.)

EXPRESS specification:

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

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.7 linear_stiffness_measure

Type definition:

A real number value of a measurement of the stiffness of a linear spring. (See also the definition for the entity linear_stiffness_measure_with_unit.)

EXPRESS specification:

```
*)
TYPE linear_stiffness_measure
= REAL;
WHERE
    WRTL1 : (SELF >= 0.0);
END_TYPE;
(*
```

Formal propositions:

WRTL1

The real number value shall be greater than zero. That is, stiffness values may only be positive in CIS/2.

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.8 linear_velocity_measure

Type definition:

A real number value of a measurement of linear velocity. (See also the definition for the entity linear_velocity_measure_with_unit.)

EXPRESS specification:

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

Notes

New for CIS/2. Unchanged in 2nd Edition

15.2.9 mass_per_length_measure

Type definition:

A real number value of a measurement of mass per unit length. (See also the definition for the entity mass_per_length_measure_with_unit.)

EXPRESS specification:

```
*)
TYPE mass_per_length_measure
  = REAL;
WHERE
  WRTM1 : (SELF > 0.0);
END_TYPE;
(*
```

Formal propositions:

WRTM1

The real number value shall be greater than zero. That is, a mass per unit length must be positive. (Anti-matter is beyond the scope of CIS/2.)

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.10 modulus_measure

Type definition:

A real number value of a measurement of modulus. (See also the definition for the entity modulus_measure_with_unit.)

EXPRESS specification:

```
*)
TYPE modulus_measure
  = REAL;
WHERE
  WRTM4 : (SELF > 0.0);
END_TYPE;
(*
```

Formal propositions:

WRTM4

The real number value shall be greater than zero. That is, a modulus must be positive.

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.11 moment_measure

Type definition:

A real number value of a measurement of moment. (See also the definition for the entity moment_measure_with_unit.)

EXPRESS specification:

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

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.12 pressure_measure

Type definition:

A real number value of a measurement of pressure. (See also the definition for the entity pressure_measure_with_unit.)

EXPRESS specification:

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

Notes

Known as PRESSURE_MEASURE in CIS/1. Unchanged in 2nd Edition.

15.2.13 rotational_acceleration_measure

Type definition:

A real number value of a measurement of rotational acceleration. (See also the definition for the entity rotational_acceleration_measure_with_unit.)

EXPRESS specification:

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

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.14 rotational_stiffness_measure

Type definition:

A real number value of a measurement of the rotational stiffness of a spring. (See also the definition for the entity rotational_stiffness_measure_with_unit.)

EXPRESS specification:

```
*)
TYPE rotational_stiffness_measure
= REAL;
WHERE
```

```

    WRTR1 : (SELF >= 0.0);
END_TYPE;
(*)

```

Formal propositions:

WRTR1

The real number value shall be greater than zero. That is, a spring stiffness must be positive.

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.2.15 rotational_velocity_measure

Type definition:

A real number value of a measurement of rotational velocity. (See also the definition for the entity rotational_velocity_measure_with_unit.)

EXPRESS specification:

```

*)
TYPE rotational_velocity_measure
= REAL;
END_TYPE;
(*)

```

Notes

New for CIS/2. Unchanged in 2nd Edition.

15.3 Units & Measures entity definitions

No entities have been modified or added in this subject area for the 2nd Edition.

15.3.1 currency_measure_with_unit

Entity definition:

A measure of a quantity of money with an appropriate unit. This entity is referenced by any parameter that requires a measure of price or cost.

EXPRESS specification:

```

*)
ENTITY currency_measure_with_unit
SUPERTYPE OF (ONEOF(currency_rate_with_unit));
    amount : REAL;
    unit : currency_unit;
END_ENTITY;
(*)

```

Attribute definitions:

amount

Provides the real number value of the measure of the price or cost. The value would normally be positive. A negative value indicates a credit or a refund.

unit

Declares the instance of `currency_unit` associated with the `currency_measure_with_unit`. There must be one (and only one) instance of `currency_unit` referenced by each instance of `currency_measure_with_unit`. An instance of `currency_measure_with_unit` can be referenced by any number of instances of `currency_measure_with_unit`.

Informal propositions:

This is not a SUBTYPE of `measure_with_unit`, even though its structure is similar.

Notes:

New for CIS/2.

See Diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.2 currency_rate_with_unit***Entity definition:***

A type of `currency_measure_with_unit` that provides the unit quantity associated with the price or cost. Thus, the instance provides a priced rate per unit quantity (e.g. \$500/t).

EXPRESS specification:

```
*)
ENTITY currency_rate_with_unit
  SUBTYPE OF (currency_measure_with_unit);
    per_quantity : measure_with_unit;
END_ENTITY;
(*
```

Attribute definitions:*per_quantity*

Declares the instance of `measure_with_unit` associated with the `currency_rate_with_unit`, which provides the numerical value with an appropriate unit of the unit quantity; e.g. 1 ton.

Notes

New for CIS/2.

See Diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.3 currency_unit***Entity definition:***

This entity provides the identification and description of a unit of currency. A price may be specified as either a unit rate (e.g. Dollars per ton) using the entity `currency_rate_with_unit` or a total quantity (e.g. Dollars), using the entity `currency_measure_with_unit`.

EXPRESS specification:

```
*)
ENTITY currency_unit;
  name : label;
```

description : OPTIONAL text;
 END_ENTITY;
 (*

Attribute definitions:

name

A short text reference used to identify the unit of currency; e.g. ‘DKr’, ‘DM’ ‘FM’, ‘FFr’, ‘GBP’, ‘EUR’, ‘NKr’, ‘USD’, etc. These are examples (taken from the *Financial Times*) and are not a definitive list of currency units. It is recommended that the term ‘\$’ is not used as a \$ in a physical file represents a NULL value. Further, a ‘£’ sign does not have an ASCII equivalent that may be easily used in a physical file.

description

A text description of the unit of currency; e.g., ‘Danish Krone’, ‘Deutschemark’, ‘Finnish Marks’, ‘French Franks’, ‘Great British Pounds Sterling’, ‘Euros’, ‘Norwegian Krone’, ‘US Dollars’, etc. These are examples (taken from the *Financial Times*) and are not a definitive list of currency units.

Informal propositions:

This is not a SUBTYPE of named_unit, even though its structure is similar.

Notes:

New for CIS/2.

See Diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.4 derived_measure_with_unit

Entity definition:

A type of measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a derived_unit. The derived_measure_with_unit may be categorized (through its SUBTYPES) providing the measure and unit for either a force per length, an inertia, a linear acceleration, a linear stiffness, a linear velocity, a mass per length, a modulus, a moment, a rotational acceleration, a rotational stiffness, or a rotational velocity.

EXPRESS specification:

*)

ENTITY derived_measure_with_unit

SUPERTYPE OF (ONEOF

(force_per_length_measure_with_unit,
 inertia_measure_with_unit,
 linear_acceleration_measure_with_unit,
 linear_stiffness_measure_with_unit,
 linear_velocity_measure_with_unit,
 mass_per_length_measure_with_unit,
 modulus_measure_with_unit,
 moment_measure_with_unit,
 rotational_acceleration_measure_with_unit,
 rotational_stiffness_measure_with_unit,

```

        rotational_velocity_measure_with_unit))
SUBTYPE OF (measure_with_unit);
WHERE
    WRD6 : 'STRUCTURAL_FRAME_SCHEMA.DERIVED_UNIT' IN TYPE OF
        (SELF\measure_with_unit.unit_component);
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `measure_with_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `measure_with_unit`, and to collate the SUBTYPEs under a common category.

Formal propositions:

WRD6

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `derived_unit`.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity `measure_with_unit` is defined in STEP Part 41.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.5 force_measure_with_unit

Entity definition:

A type of `measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `force_unit`. The `value_component` attribute (also inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `force_measure`. (The type `force_measure` resolves to a REAL number.) The `force_unit` itself is a type of `named_unit` which may (but need not) be an `si_unit`. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of (linear) force.

EXPRESS specification:

```

*)
ENTITY force_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRF17 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN TYPE OF
        (SELF\measure_with_unit.unit_component);
    WRF18 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_MEASURE' IN TYPE OF
        (SELF\measure_with_unit.value_component);
END_ENTITY;
(*)

```


Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `measure_with_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `measure_with_unit`.

Formal propositions:**WRF17**

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `force_unit`.

WRF18

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be the type `force_measure`. (The type `force_measure` resolves to a REAL number.)

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 15 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.6 force_per_length_measure_with_unit**Entity definition:**

A type of `derived_measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `force_per_length_unit`. The `force_per_length_unit` is a type of `derived_unit` which has a `force_unit` as one of its components, which may (but need not) be an `si_unit`. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of (linear) force per unit length.

EXPRESS specification:

*)

ENTITY `force_per_length_measure_with_unit`

SUBTYPE OF (`derived_measure_with_unit`);

WHERE

WRF19 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_PER_LENGTH_UNIT' IN
TYPE OF (`SELF`\`measure_with_unit`.`unit_component`);

WRF20 : 'STRUCTURAL_FRAME_SCHEMA.
FORCE_PER_LENGTH_MEASURE' IN TYPE OF
(`SELF`\`measure_with_unit`.`value_component`);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_measure_with_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRF17**

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `force_per_length_unit`.

WRF18

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be the type `force_per_length_measure`. (The type `force_per_length_measure` resolves to a REAL number.)

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 17 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.7 force_per_length_unit**Entity definition:**

A type of `derived_unit` which has 2 unit components; the first being a `force_unit` with an associated exponent of 1.0, the second being a `length_unit` with an associated exponent of -1.0. The force and length units may (but need not) be SI units.

EXPRESS specification:

*)

ENTITY `force_per_length_unit`

SUBTYPE OF (`derived_unit`);

WHERE

WRF21 : `SIZEOF(SELF\derived_unit.elements) = 2;`

WRF22 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN TYPE OF
(`SELF\derived_unit.elements[1]\derived_unit_element.unit`)) AND
(`SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0`);

WRF23 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
(`SELF\derived_unit.elements[2]\derived_unit_element.unit`)) AND
(`SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0`);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRF21**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall equal 2. That is, there shall be two (and only two) instances of `derived_unit_element` referenced by the `force_per_length_unit`.

WRF22

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of force_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 1.0.

WRF23

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to -1.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity derived_unit is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.8 force_unit**Entity definition:**

A type of named_unit that describes a unit of force, where the dimensional exponents of length and mass are set equal to 1.0, the dimensional exponents of time are set equal to -2.0; while the remaining exponents are set equal to zero. This provides a dimensional analysis of the unit equivalent to ML/T^2 .

EXPRESS specification:

*)

ENTITY force_unit

SUBTYPE OF (named_unit);

WHERE

WRF24 : (SELF\named_unit.dimensions.length_exponent = 1.0) AND
 (SELF\named_unit.dimensions.mass_exponent = 1.0) AND
 (SELF\named_unit.dimensions.time_exponent = -2.0) AND
 (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
 (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
 (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
 (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE named_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRF24**

The instance of the entity `dimensional_exponents` referenced by the attribute `dimensions` (inherited from the SUPERTYPE `named_unit`) shall set its exponents of length and mass equal to 1.0, while its exponents of time shall be set equal to -2.0; and its remaining exponents shall be set equal to zero.

When the `force_unit` is instanced with an `si_unit`, the dimensional exponents are derived using the STEP Part 41 function `dimensions_for_si_unit`, and there is no instance of the entity `dimensional_exponents` referenced by the `force_unit` instance. When the instance is encoded in a STEP Part 21 file, the inherited attribute `dimensions` is encoded as “*”.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entities `named_unit` and `dimensional_exponents`, and the function `dimensions_for_si_unit` are all defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.9 frequency_measure_with_unit**Entity definition:**

A type of `measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `frequency_unit`, and the `value_component` attribute (also inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `frequency_measure`. (The type `frequency_measure` resolves to a REAL number.) The `frequency_unit` itself is a type of `named_unit` which may (but need not) be an `si_unit`. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of frequency.

EXPRESS specification:

*)

ENTITY `frequency_measure_with_unit`

SUBTYPE OF (`measure_with_unit`);

WHERE

WRF25 : 'STRUCTURAL_FRAME_SCHEMA.FREQUENCY_UNIT' IN TYPE OF
(SELF\measure_with_unit.unit_component);

WRF26 : 'STRUCTURAL_FRAME_SCHEMA.FREQUENCY_MEASURE' IN TYPE OF
(SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `measure_with_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRF25**

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `frequency_unit`.

WRF26

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be the type `frequency_measure`. (The type `frequency_measure` resolves to a REAL number.)

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity `measure_with_unit` is defined in STEP Part 41.

See diagram 15 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.10 frequency_unit**Entity definition:**

A type of `named_unit` that describes a unit of frequency, where the dimensional exponent of time is set equal to -1.0, while the remaining exponents are set equal to zero. This provides a dimensional analysis of the unit equivalent to T^{-1} .

EXPRESS specification:

*)

ENTITY `frequency_unit`

SUBTYPE OF (`named_unit`);

WHERE

WRF27 : (SELF\`named_unit.dimensions.length_exponent` = 0.0) AND
 (SELF\`named_unit.dimensions.mass_exponent` = 0.0) AND
 (SELF\`named_unit.dimensions.time_exponent` = -1.0) AND
 (SELF\`named_unit.dimensions.electric_current_exponent` = 0.0) AND
 (SELF\`named_unit.dimensions.thermodynamic_temperature_exponent` = 0.0) AND
 (SELF\`named_unit.dimensions.amount_of_substance_exponent` = 0.0) AND
 (SELF\`named_unit.dimensions.luminous_intensity_exponent` = 0.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `named_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRF27**

The instance of the entity `dimensional_exponents` referenced by the attribute `dimensions` (inherited from the SUPERTYPE named `_unit`) shall set its exponent of time equal to -1.0, while its remaining exponents shall be set equal to zero.

When the `frequency_unit` is instantiated with an `si_unit`, the dimensional exponents are derived using the STEP Part 41 function `dimensions_for_si_unit`, and there is no instance of the entity `dimensional_exponents` referenced by the `frequency_unit` instance. When the instance is encoded in a STEP Part 21 file, the inherited attribute `dimensions` is encoded as “*”.

Informal propositions:

This is an interpreted construct; i.e. a Generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entities `named_unit` and `dimensional_exponents`, and the function `dimensions_for_si_unit` are all defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.11 inertia_measure_with_unit**Entity definition:**

A type of derived `measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being an `inertia_unit`. The `inertia_unit` is a type of derived `_unit` which has a `length_unit` as its one component, which may (but need not) be an `si_unit`. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of inertia (or second moment of area).

EXPRESS specification:

*)

ENTITY `inertia_measure_with_unit`

SUBTYPE OF (`derived_measure_with_unit`);

WHERE

WRI1 : 'STRUCTURAL_FRAME_SCHEMA.INERTIA_UNIT' IN TYPE OF
(`SELF\measure_with_unit.unit_component`);

WRI2 : 'STRUCTURAL_FRAME_SCHEMA.INERTIA_MEASURE' IN TYPE OF
(`SELF\measure_with_unit.value_component`);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_measure_with_unit`. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity `measure_with_unit`.

Formal propositions:

WRI1

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of inertia_unit.

WRI2

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of inertia_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.12 inertia_unit

Entity definition:

A type of derived_unit which has 1 unit component; being an length_unit with an associated exponent of 4.0. The length_unit may (but need not) be an si_unit.

EXPRESS specification:

*)

ENTITY inertia_unit

SUBTYPE OF (derived_unit);

WHERE

WRI3 : SIZEOF(SELF\derived_unit.elements) = 1;

WRI4 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
(SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 4.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity derived_unit.

Formal propositions:

WRI3

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall equal 1. That is, there shall be one (and only one) instance of derived_unit_element referenced by the inertia_unit.

WRI4

The one and only member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit,

through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 4.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.13 linear_acceleration_measure_with_unit

Entity definition:

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a linear_acceleration_unit. The linear_acceleration_unit is a type of derived_unit which has length and time components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of linear acceleration (i.e. the rate of change of linear velocity with time).

EXPRESS specification:

*)

ENTITY linear_acceleration_measure_with_unit

SUBTYPE OF (derived_measure_with_unit);

WHERE

WRL3 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_ACCELERATION_UNIT' IN TYPE OF
(SELF\measure_with_unit.unit_component);

WRL4 : 'STRUCTURAL_FRAME_SCHEMA.
LINEAR_ACCELERATION_MEASURE' IN TYPE OF
(SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_measure_with_unit. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity measure_with_unit.

Formal propositions:

WRL3

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of linear_acceleration_unit.

WRL4

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of linear_acceleration_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.14 linear_acceleration_unit**Entity definition:**

A type of derived_unit which has 2 unit components; the first being a length_unit with an associated exponent of 1.0, the second being a time_unit with an associated exponent of -2.0. The length and time units may (but need not) be SI units.

EXPRESS specification:

*)

ENTITY linear_acceleration_unit

SUBTYPE OF (derived_unit);

WHERE

WRL5 : SIZEOF(SELF\derived_unit.elements) = 2;

WRL6 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
(SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);

WRL7 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN TYPE OF
(SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[2]\derived_unit_element.exponent = -2.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRL5**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall equal 2. That is, there shall be two (and only two) instances of derived_unit_element referenced by the linear_acceleration_unit.

WRL6

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 1.0.

WRL7

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of time_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to -2.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity derived_unit is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.15 linear_stiffness_measure_with_unit**Entity definition:**

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a linear_stiffness_unit. The linear_stiffness_unit is a type of derived_unit which has force and length components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of linear stiffness. This could, for example, provide the measurement of the stiffness (the force required for a unit displacement) of a linear spring.

EXPRESS specification:

*)

ENTITY linear_stiffness_measure_with_unit

SUBTYPE OF (derived_measure_with_unit);

WHERE

WRL8 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_STIFFNESS_UNIT' IN TYPE OF
(SELF\measure_with_unit.unit_component);

WRL9 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_STIFFNESS_MEASURE' IN TYPE OF
(SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_measure_with_unit. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity measure_with_unit.

Formal propositions:**WRL8**

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of linear_stiffness_unit.

WRL9

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of linear_stiffness_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.16 linear_stiffness_unit**Entity definition:**

A type of derived_unit which has 2 unit components; the first being a force_unit with an associated exponent of 1.0, the second being a length_unit with an associated exponent of -1.0. The force and length units may (but need not) be SI units.

EXPRESS specification:

*)

ENTITY linear_stiffness_unit

SUBTYPE OF (derived_unit);

WHERE

WRL10 : SIZEOF(SELF\derived_unit.elements) = 2;

WRL11 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN TYPE OF
(SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);

WRL12 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
(SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRL10**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall equal 2. That is, there shall be two (and only two) instances of derived_unit_element referenced by the linear_stiffness_unit.

WRL11

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of force_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 1.0.

WRL12

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to -1.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity derived_unit is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.17 linear_velocity_measure_with_unit**Entity definition:**

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a linear_velocity_unit. The linear_velocity_unit is a type of derived_unit which has length and time components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of linear velocity (i.e. the rate of change of linear displacement with time).

EXPRESS specification:

*)

ENTITY linear_velocity_measure_with_unit

SUBTYPE OF (derived_measure_with_unit);

WHERE

WRL13 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_VELOCITY_UNIT' IN TYPE OF
(SELF\measure_with_unit.unit_component);

WRL14 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_VELOCITY_MEASURE' IN TYPE OF
(SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_measure_with_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity measure_with_unit.

Formal propositions:**WRL13**

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of linear_velocity_unit.

WRL14

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of linear_velocity_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.18 linear_velocity_unit**Entity definition:**

A type of derived_unit which has 2 unit components; the first being a length_unit with an associated exponent of 1.0, the second being a time_unit with an associated exponent of -1.0. The length and time units may (but need not) be SI units.

EXPRESS specification:

*)

```
ENTITY linear_velocity_unit
SUBTYPE OF (derived_unit);
```

```
WHERE
```

```
    WRL15 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRL16 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
              (SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
              (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRL17 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN TYPE OF
              (SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
              (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);
```

```
END_ENTITY;
```

```
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRL15**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall equal 2. That is, there shall be two (and only two) instances of derived_unit_element referenced by the linear_velocity_unit.

WRL16

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 1.0.

WRL17

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of time_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to -1.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity derived_unit is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.19 mass_per_length_measure_with_unit**Entity definition:**

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a mass_per_length_unit. The mass_per_length_unit is a type of derived_unit which has mass and length components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of mass per unit length.

EXPRESS specification:

*)

ENTITY mass_per_length_measure_with_unit

SUBTYPE OF (derived_measure_with_unit);

WHERE

WRM13 : 'STRUCTURAL_FRAME_SCHEMA.MASS_PER_LENGTH_UNIT' IN
TYPE OF (SELF\measure_with_unit.unit_component);

WRM14 : 'STRUCTURAL_FRAME_SCHEMA.
MASS_PER_LENGTH_MEASURE' IN TYPE OF
(SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_measure_with_unit. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity measure_with_unit.

Formal propositions:**WRM11**

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of mass_per_length_unit.

WRM12

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of mass_per_length_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.20 mass_per_length_unit**Entity definition:**

A type of derived_unit which has 2 unit components; the first being a mass_unit with an associated exponent of 1.0, the second being a length_unit with an associated exponent of -1.0. The mass and length units may (but need not) be SI units.

EXPRESS specification:

*)

ENTITY mass_per_length_unit

SUBTYPE OF (derived_unit);

WHERE

WRM15 : SIZEOF(SELF\derived_unit.elements) = 2;

WRM16 : ('STRUCTURAL_FRAME_SCHEMA.MASS_UNIT' IN TYPE OF
(SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);

WRM17 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
(SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRM15**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall equal 2. That is, there shall be two (and only two) instances of derived_unit_element referenced by the mass_per_length_unit.

WRM16

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of mass_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 1.0.

WRM17

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to -1.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity derived_unit is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.21 modulus_measure_with_unit**Entity definition:**

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a modulus_unit. The modulus_unit is a type of derived_unit which has a length component, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of modulus. This could, for example, provide the measurement of the elastic or plastic section modulus of a section profile.

EXPRESS specification:

*)

ENTITY modulus_measure_with_unit

SUBTYPE OF (derived_measure_with_unit);

WHERE

WRM27 : 'STRUCTURAL_FRAME_SCHEMA.MODULUS_UNIT' IN
TYPE OF (SELF\measure_with_unit.unit_component);

WRM28 : 'STRUCTURAL_FRAME_SCHEMA.MODULUS_MEASURE' IN
TYPE OF (SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_measure_with_unit. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity measure_with_unit.

Formal propositions:**WRM27**

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of modulus_unit.

WRM28

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of modulus_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.22 modulus_unit**Entity definition:**

A type of derived_unit which has 1 unit component; that being a length_unit with an associated exponent of 3.0. The length unit may (but need not) be SI units.

EXPRESS specification:

*)

ENTITY modulus_unit

SUBTYPE OF (derived_unit);

WHERE

WRM29 : SIZEOF(SELF\derived_unit.elements) = 1;

WRM30 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN

TYPE OF (SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 3.0);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRM29**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall equal 1. That is, there shall be one (and only one) instance of derived_unit_element referenced by the linear_stiffness_unit.

WRM30

The one (and only) member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE derived_unit) shall reference a type of length_unit, through the attribute unit of the entity derived_unit_element, and the exponent attribute of that entity instance shall be equal to 3.0.

Informal propositions:

This is an interpreted construct; i.e. a Generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity derived_unit is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.23 moment_measure_with_unit

Entity definition:

A type of derived `measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `moment_unit`. The `moment_unit` is a type of derived `unit` which has force and length components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of moment. This could, for example, provide the measurement of a bending moment applied to a finite element.

EXPRESS specification:

```
*)
ENTITY moment_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
WRM31 : 'STRUCTURAL_FRAME_SCHEMA.MOMENT_UNIT' IN
    TYPE OF (SELF\measure_with_unit.unit_component);
WRM32 : 'STRUCTURAL_FRAME_SCHEMA.MOMENT_MEASURE' IN
    TYPE OF (SELF\measure_with_unit.value_component);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_measure_with_unit`. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity `measure_with_unit`.

Formal propositions:

WRM31

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `moment_unit`.

WRM32

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `moment_measure`.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.24 moment_unit

Entity definition:

A type of derived `unit` which has 2 unit components; the first being a `force_unit` with an associated exponent of 1.0, the second being a `length_unit` with an associated exponent of 1.0. The force and length units may (but need not) be SI units.

EXPRESS specification:

```

*)
ENTITY moment_unit
SUBTYPE OF (derived_unit);
WHERE
    WRM33 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRM34 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN
    TYPE OF (SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
    (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRM35 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
    TYPE OF (SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
    (SELF\derived_unit.elements[2]\derived_unit_element.exponent = 1.0);
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRM33**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall equal 2. That is, there shall be two (and only two) instances of `derived_unit_element` referenced by the `linear_stiffness_unit`.

WRM34

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `force_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to 1.0.

WRM35

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `length_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to 1.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity `derived_unit` is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.25 positive_length_measure_with_unit

Entity definition:

A type of `measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `length_unit`, while the `value_component` is specified as being a `positive_length_measure`. The length unit may (but need not) be an SI unit. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of length that is constrained to have a positive value. This could, for example, provide the measurement of the length of a prismatic part.

EXPRESS specification:

```
*)
ENTITY positive_length_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
WRP17 : 'STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
    TYPE OF (SELF\measure_with_unit.unit_component);
WRP18 : 'STRUCTURAL_FRAME_SCHEMA.POSITIVE_LENGTH_MEASURE'
    IN TYPE OF (SELF\measure_with_unit.value_component);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `measure_with_unit`. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity.

Formal propositions:

WRP17

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `length_unit`.

WRP18

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `positive_length_measure`.

Notes

New for CIS/2.

The entities `measure_with_unit` and `named_unit` as well as the type `positive_length_measure` are all defined in STEP Part 41.

See Diagram 15 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.26 pressure_measure_with_unit

Entity definition:

A type of `measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `pressure_unit`, and the `value_component` attribute (also inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `pressure_measure`. (The type `pressure_measure` resolves to a REAL

number.) The `pressure_unit` itself is a type of `named_unit` which may (but need not) be an `si_unit`. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of force per unit area. Examples of these measurements include a pressure applied to a finite element, a resultant stress, or a specified strength of a material.

EXPRESS specification:

*)

ENTITY `pressure_measure_with_unit`

SUBTYPE OF (`measure_with_unit`);

WHERE

WRP19 : 'STRUCTURAL_FRAME_SCHEMA.PRESSURE_UNIT' IN TYPE OF
(SELF\measure_with_unit.unit_component);

WRP20 : 'STRUCTURAL_FRAME_SCHEMA.PRESSURE_MEASURE' IN
TYPE OF (SELF\measure_with_unit.value_component);

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `measure_with_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:

WRP19

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `pressure_unit`.

WRP20

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be the type `pressure_measure`. (The type `pressure_measure` resolves to a REAL number.)

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 15 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.27 `pressure_unit`

Entity definition:

A type of `named_unit` that describes a unit of pressure, stress or strength, where the dimensional exponent of length is set equal to 1.0, the mass exponent is set equal to – 1.0, the time exponent is set equal to –2.0, while the remaining exponents are set equal to zero. This provides a dimensional analysis of the unit equivalent to $ML^{-1}T^{-2}$ (which is derived from MLT^{-2}/L^2).

EXPRESS specification:

```

*)
ENTITY pressure_unit
SUBTYPE OF (named_unit);
WHERE
    WRP21 : (SELF\named_unit.dimensions.length_exponent = -1.0) AND
              (SELF\named_unit.dimensions.mass_exponent = 1.0) AND
              (SELF\named_unit.dimensions.time_exponent = -2.0) AND
              (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
              (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
              (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
              (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE named_unit. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRP21**

The instance of the entity dimensional_exponents referenced by the attribute dimensions (inherited from the SUPERTYPE named_unit) shall set its exponent of length equal to -1.0, its exponent of mass equal to 1.0, its exponent of time equal to -2.0; while its remaining exponents shall be set equal to zero.

When the pressure_unit is instanced with an si_unit, the dimensional exponents are derived using the STEP Part 41 function dimensions_for_si_unit, and there is no instance of the entity dimensional_exponents referenced by the pressure_unit instance. When the instance is encoded in a STEP Part 21 file, the inherited attribute dimensions is encoded as “*”.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entities named_unit, si_unit, and dimensional_exponents, and the function dimensions_for_si_unit are all defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.28 rotational_acceleration_measure_with_unit**Entity definition:**

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a rotational_acceleration_unit. The rotational_acceleration_unit is a type of derived_unit which has angle and time components, which may (but need not) be SI units. This entity is referenced by any

parameter requiring a numerical value with an appropriate unit of a measurement of rotational acceleration (i.e. the rate of change of rotational velocity with time).

EXPRESS specification:

```
*)
ENTITY rotational_acceleration_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
WRR35 : 'STRUCTURAL_FRAME_SCHEMA.
    ROTATIONAL_ACCELERATION_UNIT' IN TYPE OF
    (SELF\measure_with_unit.unit_component);
WRR36 : 'STRUCTURAL_FRAME_SCHEMA.
    ROTATIONAL_ACCELERATION_MEASURE' IN TYPE OF
    (SELF\measure_with_unit.value_component);
END_ENTITY;
(*)
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_measure_with_unit`. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity `measure_with_unit`.

Formal propositions:

WRR35

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `rotational_acceleration_unit`.

WRR36

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `rotational_acceleration_measure`.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.29 rotational_acceleration_unit

Entity definition:

A type of `derived_unit` which has 2 unit components; the first being a `plane_angle_unit` with an associated exponent of 1.0, the second being a `time_unit` with an associated exponent of -2.0. The angle and time units may (but need not) be SI units.

EXPRESS specification:

```
*)
ENTITY rotational_acceleration_unit
SUBTYPE OF (derived_unit);
WHERE
```

```

WRR37 : SIZEOF(SELF\derived_unit.elements) = 2;
WRR38 : ('STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN TYPE OF
(SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
WRR39 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN TYPE OF
(SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[2]\derived_unit_element.exponent = -2.0);
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:

WRR37

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall equal 2. That is, there shall be two (and only two) instances of `derived_unit_element` referenced by the `rotational_acceleration_unit`.

WRR38

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `plane_angle_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to 1.0.

WRR39

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `time_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to -2.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.30 rotational_stiffness_measure_with_unit

Entity definition:

A type of `derived_measure_with_unit` where the `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) is specified as being a `rotational_stiffness_unit`. The `rotational_stiffness_unit` is a type of `derived_unit` which has force, length and angle components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of rotational stiffness. This could, for example, provide the measurement of the stiffness (the moment required for a unit rotation) of a rotational spring or hinge.

EXPRESS specification:

```

*)
ENTITY rotational_stiffness_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
WRR40 : 'STRUCTURAL_FRAME_SCHEMA.
  ROTATIONAL_STIFFNESS_UNIT' IN TYPE OF
  (SELF\measure_with_unit.unit_component);
WRR41 : 'STRUCTURAL_FRAME_SCHEMA.
  ROTATIONAL_STIFFNESS_MEASURE' IN TYPE OF
  (SELF\measure_with_unit.value_component);
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_measure_with_unit`. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity `measure_with_unit`.

Formal propositions:**WRR40**

The `unit_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `rotational_stiffness_unit`.

WRR41

The `value_component` attribute (inherited from the SUPERTYPE `measure_with_unit`) shall be a type of `rotational_stiffness_measure`.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.31 rotational_stiffness_unit**Entity definition:**

A type of `derived_unit` which has 3 unit components; the first being a `force_unit` with an associated exponent of 1.0, the second being a `length_unit` with an associated exponent of 1.0, and the third being a `plane_angle_unit` with an associated exponent of -1.0. The force and angle units may (but need not) be SI units.

EXPRESS specification:

```

*)
ENTITY rotational_stiffness_unit
SUBTYPE OF (derived_unit);
WHERE
  WRR42 : SIZEOF(SELF\derived_unit.elements) = 3;

```

```

WRR43 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN TYPE OF
(SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
WRR44 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN TYPE OF
(SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[2]\derived_unit_element.exponent = 1.0);
WRR45 : ('STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN TYPE OF
(SELF\derived_unit.elements[3]\derived_unit_element.unit)) AND
(SELF\derived_unit.elements[3]\derived_unit_element.exponent = -1.0);
END_ENTITY;
(*)

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:

WRR42

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall equal 3. That is, there shall be three (and only three) instances of `derived_unit_element` referenced by the `rotational_stiffness_unit`.

WRR43

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `force_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to 1.0.

WRR44

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `length_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to 1.0.

WRR45

The third member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `plane_angle_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to -1.0.

Informal propositions:

This is an interpreted construct; i.e. a Generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.32 rotational_velocity_measure_with_unit

Entity definition:

A type of derived_measure_with_unit where the unit_component attribute (inherited from the SUPERTYPE measure_with_unit) is specified as being a rotational_velocity_unit. The rotational_velocity_unit is a type of derived_unit which has angle and time components, which may (but need not) be SI units. This entity is referenced by any parameter requiring a numerical value with an appropriate unit of a measurement of rotational velocity (i.e. the rate of change of angular displacement with time).

EXPRESS specification:

```
*)
ENTITY rotational_velocity_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
WRR46 : 'STRUCTURAL_FRAME_SCHEMA.ROTATIONAL_VELOCITY_UNIT' IN TYPE
      OF (SELF\measure_with_unit.unit_component);
WRR47 : 'STRUCTURAL_FRAME_SCHEMA.
      ROTATIONAL_VELOCITY_MEASURE' IN TYPE OF
      (SELF\measure_with_unit.value_component);
END_ENTITY;
(*
```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE derived_measure_with_unit. The purpose of this entity is to constrain further the use of (or specialize) the SUPERTYPE entity measure_with_unit.

Formal propositions:

WRR46

The unit_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of rotational_velocity_unit.

WRR47

The value_component attribute (inherited from the SUPERTYPE measure_with_unit) shall be a type of rotational_velocity_measure.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

See diagram 16 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

15.3.33 rotational_velocity_unit

Entity definition:

A type of derived_unit which has 2 unit components; the first being a plane_angle_unit with an associated exponent of 1.0, the second being a time_unit with an associated exponent of -1.0. The angle and time units may (but need not) be SI units.

EXPRESS specification:

```

*)
ENTITY rotational_velocity_unit
SUBTYPE OF (derived_unit);
WHERE
    WRR48 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRR49 : ('STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN TYPE OF
    (SELF\derived_unit.elements[1]\derived_unit_element.unit)) AND
    (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRR50 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN TYPE OF
    (SELF\derived_unit.elements[2]\derived_unit_element.unit)) AND
    (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `derived_unit`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity.

Formal propositions:**WRR48**

The size of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall equal 2. That is, there shall be two (and only two) instances of `derived_unit_element` referenced by the `rotational_velocity_unit`.

WRR49

The first member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `plane_angle_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to 1.0.

WRR50

The second member of the aggregation associated with the attribute elements (inherited from the SUPERTYPE `derived_unit`) shall reference a type of `time_unit`, through the attribute unit of the entity `derived_unit_element`, and the exponent attribute of that entity instance shall be equal to -1.0.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes:

New for CIS/2.

The entity `derived_unit` is defined in STEP Part 41.

See diagram 14 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16 LPM/6 ITEM REFERENCES

16.1 Item References concepts and assumptions

CIS/1 contained the idea of ‘international flavours’. These were lists of product references (represented as STRINGS) that allowed information to be exchanged by being ‘passed-by-reference’ rather than being ‘passed by attribute value’. This concept is maintained in CIS/2, although CIS/2 has aimed to make the flavours more easily extensible, and the reference itself more compact². The solution adopted for CIS/2 is to make these references into entities, which can be passed via a STEP Part 21 file (see diagram 21 of the EXPRESS-G diagrams in Appendix B).

To facilitate this, the four classes of product item are defined:

1. **Standard Items** - Internationally recognised generic product items (such as hot-rolled sections) that comply with formal standards that have international visibility.
2. **Proprietary Items** - Manufacturer-specific proprietary product items (such as cold rolled sections) that comply with manufacturer’s catalogues. This class also includes product items that comply with specifications agreed between a consortium of manufacturers.
3. **Library Items** - These are product items that are not explicitly covered in standard or manufacturer’s catalogues. This class also includes product items that have been compiled and placed in a ‘library’ by third party trade associations.
4. **Non-standard Items** - These are product items that cannot be passed-by-reference because an agreed reference does not exist or is not supported by both translators.

In order to exchange product data ‘by reference’, both sending and receiving applications must have access to the information contained in the external source that provided the reference, if they are to process that data correctly.

The use of ‘item references’ has significant implications for implementation. Software developers are strongly advised to read the Conformance Requirements^[9] and the Implementation Guide^[7] before writing their translator coding.

The entity `structural_frame_item` is used as a generic entity such that associations may be made between, for example, a `section_profile` and an `item_reference`. The ‘associative’ nature of many of the entities in this section resolves a number of many-to-many relationships. This allows, for example, a fastener to be assigned several prices, while at the same time, allowing the price to be assigned to many fasteners.

Not all of the SUBTYPES of `structural_frame_item` are described in this section. Many, such as material, part, assembly, and fastener, have their own subject area.

16.2 Item References type definitions

There were no types specifically defined for this subject area in CIS/2. The following types have been added for the 2nd Edition.

- `boolean_value`
- `measure_select`

² The product references in CIS/1 often exceeded 32 characters, with a great deal of repetition.

16.2.1 boolean_value

Type definition

A named type resolving to the simple BOOLEAN type. Valid values are TRUE or FALSE. This type is used in the select type measure_select.

EXPRESS specification:

```
*)
TYPE boolean_value
  = BOOLEAN;
END_TYPE;
(*
```

Notes

New for CIS/2 2nd Edition.

16.2.2 measure_select

Type definition

A select type, offering a choice of the type of measure that may be used to specify the value of an item_property. This type is used in the entity item_property.

EXPRESS specification:

```
*)
TYPE measure_select
  = SELECT (measure_with_unit, measure_value, boolean_value);
END_TYPE;
(*
```

Notes

New for CIS/2 2nd Edition.

16.3 Item References entity definitions

The following entities have been added to this subject area for the 2nd Edition:

- item_cost_code
- item_cost_code_assigned
- item_cost_code_with_source
- item_property_with_source

The following entities have been modified in this subject area for the 2nd Edition:

- item_property
- item_ref_source_library
- item_ref_source_proprietary
- item_ref_source_standard
- structural_frame_item

16.3.1 document_standard

Entity definition:

A type of `document_with_class` that constrains the `documents_type` to be either a “Standard Specification” or a “Code of Practice”. This entity also derives the set of relevant clauses contained in the standard document by examining where the document has been quoted as a reference source. This entity represents an identifiable reference to a ‘Standard Specification’ or ‘Code of Practice’. The ‘standard document’ may be published by a national or international standards institute or an industrially based trade association.

EXPRESS specification:

```
*)
ENTITY document_standard
SUBTYPE OF (document_with_class);
DERIVE
    clauses : SET [1:?] OF document_usage_constraint := bag_to_set (USEDIN (SELF,
        'STRUCTURAL_FRAME_SCHEMA.
        DOCUMENT_USAGE_CONSTRAINT.SOURCE'));
INVERSE
    relevant_clauses : SET [1:?] OF document_usage_constraint FOR source;
WHERE
    WRD11 : ((SELF\document.kind.product_data_type = 'Standard Specification') OR
        (SELF\document.kind.product_data_type = 'Code of Practice'));
END_ENTITY;
(*
```

Attribute definitions:

clauses

This attribute derives the set of one or more instances of `document_usage_constraint` that use this instance of `document_standard` as their source. It is implied that this ‘standard document’ contains the clauses (chapters or subsections) that are relevant to the product data.

Formal propositions:

relevant_clauses

The number of instances of `document_usage_constraint` that reference this instance of `document_standard` (via the attribute `source`) shall be greater than zero. This constrains the relationship from `document_usage_constraint` to `document` for this SUBTYPE, such that an instance of `document_standard` cannot exist without being referenced by an instance of `document_usage_constraint`. That is, `document_standard` is existence dependent on `document_usage_constraint`. In other words, a ‘Standard Specification’ or ‘Code of Practice’ cannot be captured without having at least one clause quoted.

WRD11

The attribute `product_data_type` of the entity `document_type` referenced by the attribute `kind` (inherited from the SUPERTYPE `document`) shall be assigned the value ‘Standard Specification’ or ‘Code of Practice’. Note, the rule is case sensitive, the value must be given exactly as shown.

In other words, the `document_standard` represents an identifiable reference to a ‘Standard Specification’ or ‘Code of Practice’.

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Notes

New for CIS/2.

The entities `document`, `document_type`, `document_usage_constraint`, and `document_with_class` are all defined in STEP Part 41.

See Diagram 73 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.2 document_usage**Entity definition:**

A type of `document_relationship` that provides that checks the uniqueness of the relationship declared in the SUPERTYPE. The SUPERTYPE `document_relationship` creates a relationship between two instance of `document`, giving the relationship a name and a description. This entity (`document_usage`) provides a unique identifier for the relationship, and ensures that the document does not relate to itself, either directly or indirectly.

This entity is used when the dependency of documents is at issue. For example, a Standard Specification (document #1) may be related to (and be defined by) a group of other reference documents (documents #2, #3, #4). The reference documents may be related to (and be defined by) other reference documents. This entity checks that the relationships are not cyclic (self-defining) such that reference documents (documents #2, #3, #4) are is related to (and defined by) the Standard Specification (document #1).

EXPRESS specification:

*)

ENTITY `document_usage`

SUBTYPE OF (`document_relationship`);

UNIQUE

URD2 : SELF\document_relationship.name,
SELF\document_relationship.relateing_document,
SELF\document_relationship.related_document;

WHERE

WRD12 : acyclic_document_relationship(SELF,
[SELF\document_relationship.related_document],
'STRUCTURAL_FRAME_SCHEMA.
DOCUMENT_RELATIONSHIP.RELATED_DOCUMENT');

END_ENTITY;

(*

Attribute definitions:

This entity has no attributes of its own. However, it does inherit several attributes from its SUPERTYPE `document_relationship`. The purpose of this entity is to constrain the use of (or specialize) the SUPERTYPE entity `document_relationship` to ensure that only unique relationships are formed between documents.

Formal propositions:**URD12**

The combination of the values of the attributes name, relating_document, and related_document (inherited from the SUPERTYPE document_relationship) shall be unique to this document_usage. That is, the inherited name, relating_document, and related_document uniquely identify an instance of document_usage. This prevents the association between two documents being repeated with the same name. This does not, on the other hand, prevent two documents from being related twice if the name of the relationship is different for both instances.

WRD12

The graph structure of the document 'nodes' and document_relationship 'links' shall be acyclic. Each document shall not be a descendant of itself in the graph structure. In other words, a document shall not be related to itself, either directly, or indirectly through other relationships.

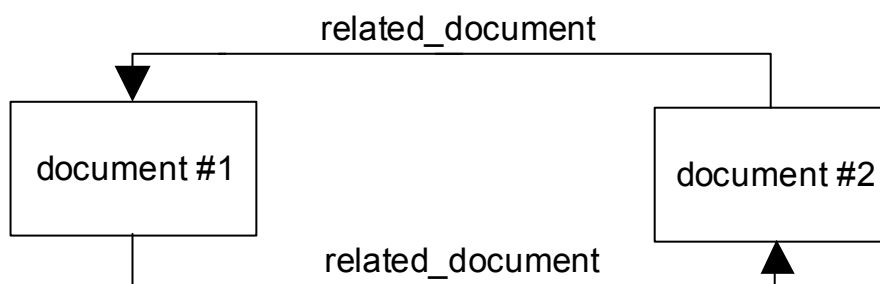


Figure 16.1 *An example of a cyclic document relationship*

This WHERE rule uses the STEP Part 41 function `acyclic_document_relationship` to determine whether or not the given documents have been self-defined by the associations made in the `document_relationship` inherited by the instance of `document_usage`. The function examines the instance of `document_usage` and the instance of `document` referenced by the attribute `related_document`. It then calculates the 'nodes' and 'links' of the graph that has this document (as a node) and this `document_relationship` (as a link). The function returns a value of TRUE if the graph is acyclic (i.e. the document referenced by the attribute `related_document` is not used again as a 'related_document' in the same graph). Otherwise, the function returns a value of FALSE (i.e. the document is self-defining).

Notes

New for CIS/2.

This entity is based upon the entity `product_definition_usage` defined in STEP Part 44.

The entities `document` and `document_relationship`, and the function `acyclic_document_relationship` are all defined in STEP Part 41.

See Diagram 73 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.3 flavour

Entity definition:

A type of group_assignment that brings together a set of two or more item references. The group_assignment entity is taken from STEP Part 41. An instance of flavour need not represent the complete flavour; i.e. the complete set of item references defined for one region. It may simply be an extract from that set. The complete flavour (containing possibly thousands of item references) is represented by the instance of group that this entity references through the attribute assigned_group (inherited from its SUPERTYPE group_assignment). This entity may be thought of as a ‘flavour list’ rather than the ‘flavour’ itself.

EXPRESS specification:

```
*)
ENTITY flavour
SUBTYPE OF (group_assignment);
    items : SET [2:?] OF item_reference;
END_ENTITY;
(*
```

Attribute definitions:

items

Declares the set of two or more separate instances of item_reference associated with the flavour, which are assigned to the group. The attribute assigned_group is inherited from the SUPERTYPE group_assignment. There must be at least two instances of item_reference referenced by each instance of flavour. An instance of item_reference can be referenced by any number of instances of flavour. That is, the same identifier can be placed in two different flavours. This allows for the definition of sub-flavours. The item_references can be of any type (standard, proprietary, library, or uncategorized).

Informal propositions:

This is an interpreted construct; i.e. a generic entity from STEP Part 41 has been specialized for use in LPM/6.

Although an item_reference can be of any type, the flavour would normally be instantiated such that all the instances of item_reference are of the same type (standard, proprietary, or library).

As a SUBTYPE of group_assignment, this entity may be instantiated with one of its sibling SUBTYPES such as group_assignment_approved or group_assignment_actioned. These would provide the flavour with a level of approval or instruct the receiving application as to what to do with the flavour. It may also be instantiated with its sibling SUBTYPE media_content. In this case, the flavour list and the associated item references are the contents of a media file. That media file could be a STEP Part 21 file. (These complex entity instances would be encoded in a STEP Part 21 file using external mapping).

Notes:

New for CIS/2; partially addressed by CIS_FILE in CIS/1.

The entities group and group_assignment are defined in STEP Part 41.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.4 item_cost_code

Entity definition:

A mechanism for classifying structural_frame_items by their relative cost. The cost code defined here may be used in estimating and tendering.

EXPRESS specification:

```
*)
ENTITY item_cost_code
SUPERTYPE OF (item_cost_code_with_source);
    cost_code : label;
    description : OPTIONAL text;
END_ENTITY;
(*
```

Attribute definitions:

cost_code

Short alphanumerical reference that when assigned to a structural_frame_item, may be used to classify that structural_frame_item in terms of a codified costing. The cost code would normally be taken from a pre-defined list and be particular to a company.

description

An optional text description of the cost code.

Notes

New for CIS/2 2nd Edition.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

16.3.5 item_cost_code_assigned

Entity definition:

This entity provides a unique association between an instance of item_code_code and an instance of structural_frame_item. The association provides a codified cost for a structural_frame_item. This is effectively an intersection entity that resolves the many-to-many relationship between the entities item_property and structural_frame_item.

EXPRESS specification:

```
*)
ENTITY item_cost_code_assigned;
    code : item_cost_code;
    costed_item : structural_frame_item;
UNIQUE
    URI8 : code, costed_item;
END_ENTITY;
(*
```

Attribute definitions:

code

Declares the instance of item_cost_code associated with the item_property_assigned, which provides the codified costing that is being assigned to the structural_frame_item.

costed_item

Declares the instance of `structural_frame_item` (or one of its SUBTYPES) associated with the `item_cost_code_assigned`. It is implied that this `structural_frame_item` has the cost code that is being assigned.

Formal propositions:*URI8*

The combination of instances of `item_cost_code` (referenced by the attribute code) and `structural_frame_item` (referenced by the attribute `costed_item`) shall be unique to this instance of `item_cost_code_assigned`. This prevents repetition of the same association. This does not prevent the `structural_frame_item` having many cost codes. Neither does it prevent the `item_cost_code` being possessed by (or a characteristic of) many instances of `structural_frame_item`.

Notes

New for CIS/2 2nd Edition.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

16.3.6 item_cost_code_with_source**Entity definition:**

A type of `item_code_code` associated with an instance of `item_ref_source`. It is implied that this source provides a formalised definition of the cost code used.

EXPRESS specification:

*)

ENTITY `item_cost_code_with_source`

SUBTYPE OF (`item_cost_code`);

`source` : `item_ref_source`;

UNIQUE

 URI9 : SELF\item_cost_code.cost_code, `source`;

END_ENTITY;

(*

Attribute definitions:*source*

Declares the instance of `item_ref_source` associated with the cost code. It is implied that this source provides the definitions of the cost codes.

Formal propositions:*URI8*

The combination of the value of the attribute `cost_code` (inherited from the SUPERTYPE `item_cost_code`) shall be unique to the instance of `item_ref_source` (referenced by the attribute `source`). In other words, the cost code shall be unique within the source. This prevents repetition of the same association. This does not prevent an `item_ref_source` being the source for many instances of `item_cost_code_with_source`.

Notes

New for CIS/2 2nd Edition.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

16.3.7 item_property

Entity definition:

This entity allows a generic property to be defined. The property must be given a name and a value (through the associated instance of `measure_with_unit`). The property may also be given a description.

EXPRESS specification:

```
*)
ENTITY item_property
  SUPERTYPE OF (item_property_with_source);
  property_name : label;
  property_description : OPTIONAL text;
  property_value : measure_select;
END_ENTITY;
(*
```

Attribute definitions:

property_name

A short alphanumeric reference for the property that may be used to identify the instance of `item_property`.

property_description

A human readable free text description of the `item_property`.

property_value

Provides the logical, numerical, or descriptive value of the `item_property`. The selection of either a `measure_value`, a `boolean_value` or an instance of `measure_with_unit` is made through the select type `measure_select`. To represent a logical or ‘flag’ type property, this attribute should be populated as a `boolean_value`. Where a property without units is represented, this attribute should be populated as a `measure_value`. Where a physical property is represented, this attribute should reference an instance of `measure_with_unit`. The instance referenced will represent the (numerical) value with an appropriate unit for the property.

Informal propositions:

This may be thought of as a name-value pair construct. It is intended to be used with the entity `item_property_assigned`, to provide a name and a value of a generic property of a `structural_frame_item`.

This is a very flexible and unconstrained construct. Its use should be agreed with other software vendors implementing CIS/2 translators.

Notes

New for CIS/2.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

16.3.8 item_property_assigned

Entity definition:

This entity provides a unique association between an instance of item_property and an instance of structural_frame_item. The association provides a name and a value of a generic property of a structural_frame_item. This is effectively an intersection entity that resolves the many-to-many relationship between the entities item_property and structural_frame_item.

EXPRESS specification:

```
*)
ENTITY item_property_assigned;
    property : item_property;
    item : structural_frame_item;
UNIQUE
    URI1 : property, item;
END_ENTITY;
(*
```

Attribute definitions:

property

Declares the instance of item_property associated with the item_property_assigned, which provides the name and value of the property that is being assigned to the structural_frame_item.

item

Declares the instance of structural_frame_item (or one of its SUBTYPEs) associated with the item_property_assigned. It is implied that this structural_frame_item possesses the property that is being assigned.

Formal propositions:

URI1

The combination of instances of item_property (referenced by the attribute property) and structural_frame_item (referenced by the attribute item) shall be unique to this instance of item_property_assigned. This prevents repetition of the same association. This does not prevent the structural_frame_item possessing many properties. Neither does it prevent the item_property being possessed by (or a characteristic of) many instances of structural_frame_item.

Notes

New for CIS/2.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.9 item_property_with_source

Entity definition:

A type of item_property associated with an instance of item_ref_source. It is implied that this source provides a formalised definition of the item property.

EXPRESS specification:

```

*)
ENTITY item_property_with_source
SUBTYPE OF (item_property);
    source : item_ref_source;
UNIQUE
    URI7 : SELF\item_property.property_name, source;
END_ENTITY;
(*

```

Attribute definitions:*source*

Declares the instance of `item_ref_source` associated with the cost code. It is implied that this source provides the definitions of the cost codes.

Formal propositions:*URI7*

The combination of the value of the attribute `property_name` (inherited from the SUPERTYPE `item_property`) shall be unique to the instance of `item_ref_source` (referenced by the attribute `source`). In other words, the property name shall be unique within the source. This prevents repetition of the same association. This does not prevent an `item_ref_source` being the source for many instances of `item_property_with_source`.

Notes

New for CIS/2 2nd Edition.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

16.3.10 item_ref_source**Entity definition:**

An abstract SUPERTYPE entity that categorizes (though its SUBTYPEs) the source for the item reference as either a standard source, a proprietary source, or library source. (See definitions of the SUBTYPEs.)

This is also used to define the source for the entities `item_property` and `item_cost_code`.

EXPRESS specification:

```

*)
ENTITY item_ref_source
ABSTRACT SUPERTYPE OF (ONEOF
    (item_ref_source_standard,
    item_ref_source_proprietary,
    item_ref_source_library));
END_ENTITY;
(*

```

Attribute definitions:

This entity has no attributes of its own. The purpose of this entity is to collate the SUBTYPEs under a common category.

Formal propositions:**ABSTRACT SUPERTYPE**

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPES.

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition, however, the scope of its application has increased.

16.3.11 item_ref_source_documented**Entity definition:**

This entity provides a unique association between an instance of `item_ref_source` and an instance of `document_usage_constraint`. The association provides a document reference for the source of an `item_reference`. For example, the dimensions and properties of a series of hot rolled section may be specified in a clause of national standard. Similarly, a range of cold-formed purlin may be specified in a particular section of a manufacturer's catalogue structural `frame_item`. This is effectively an intersection entity that resolves the many-to-many relationship between the entities `item_ref_source` and `document_usage_constraint`.

EXPRESS specification:

*)

```
ENTITY item_ref_source_documented;
    documented_item_source : item_ref_source;
    document_reference : document_usage_constraint;
UNIQUE
    URI2 : documented_item_source, document_reference;
END_ENTITY;
(*)
```

Attribute definitions:**documented_item_source**

Declares the instance of `item_ref_source` associated with the `item_ref_source_documented`. It is implied that this `item_ref_source` is specified or described in the clause of the document referenced by the `document_usage_constraint`.

document_reference

Declares the instance of `document_usage_constraint` associated with the `item_ref_source_documented`, which provides the document reference for the source of the collection of item references. It is implied that this `document_usage_constraint` provides an identifiable reference to a clause or subsection of an identifiable document.

Formal propositions:**URI2**

The combination of instances of `item_ref_source` (referenced by the attribute `documented_item_source`) and `document_usage_constraint` (referenced by the attribute `document_reference`) shall be unique to this instance of `item_ref_source_documented`. This prevents repetition of the same association. This does not prevent the

item_ref_source possessing many document references. Neither does it prevent the document references being possessed by (or a characteristic of) many instances of item_ref_source.

Notes

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.12 item_ref_source_library

Entity definition:

A type of item_ref_source that provide the defining parameters for a ‘library’ that acts as the source of the item reference information. The ‘library’ must be given an owner, a name and a date. It may also be given a version. This entity may be referenced by any number of instances of item_reference_library.

This entity provides the source of information for products that are not explicitly covered in standard or manufacturer’s catalogues. This includes product information that has been compiled and placed in a ‘library’ by a third party trade association. This could include items covered by ISO 13584 – compatible ‘Parts Libraries’.

EXPRESS specification:

*)

ENTITY item_ref_source_library

SUBTYPE OF (item_ref_source);

library_owner : person_and_organization;

library_name : label;

date_of_library : calendar_date;

version_of_library : OPTIONAL label;

INVERSE

library_items : SET [0:?] OF item_reference_library FOR source;

END_ENTITY;

(*

Attribute definitions:

library_owner

Declares the instance of person_and_organization associated with the item_ref_source_library, which provides the defining parameters of a person and organization. It is implied that this person and organization acts as the owner of ‘library’. The same person and organization may own several libraries.

library_name

A short text reference that is used to identify the library that acts as the source of the item references.

date_of_library

Declares the instance of calendar_date associated with the item_ref_source_library, which provides the date when this version of the ‘library’ was created. If the library has not been given a version, this date provides the date when the library was created.

version_of_library

An optional short text reference that is used to identify the version of the library that acts as the source of the item references. A library can have many versions.

Formal propositions:*library_items*

The set of instances of *item_reference_library* that reference this instance of *item_ref_source_library* (via the attribute *source*) shall be greater than or equal to zero.

This INVERSE clause has been relaxed for the 2nd Edition, due to the use of item_ref_source in entities other than item_reference.

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition

16.3.13 item_ref_source_proprietary**Entity definition:**

A type of *item_ref_source* that provide the defining parameters for a manufacturer's catalogue that acts as the source of the item reference information for a range of proprietary products. The manufacturer's catalogue must be given an organization that is assumed to be the manufacturer. The range of the proprietary products must be given a name and a date. It may also be given a version. This entity may be referenced by any number of instances of *item_reference_proprietary*.

This entity provides the source of information for manufacturer-specific proprietary products (such as cold rolled sections), which comply with manufacturer's catalogues. This includes products that comply with specifications agreed between a consortium of manufacturers.

EXPRESS specification:

*)

ENTITY *item_ref_source_proprietary*

SUBTYPE OF (*item_ref_source*);

manufacturers_name : organization;

manufacturers_range : label;

year_of_range : year_number;

version_of_range : OPTIONAL label;

INVERSE

proprietary_items : SET [0:?] OF *item_reference_proprietary* FOR *source*;

END_ENTITY;

(*

Attribute definitions:*manufacturers_name*

Declares the instance of organization associated with the *item_ref_source_proprietary*, which provides the defining parameters of an organization. It is implied that this organization acts as the manufacturer of the range of the proprietary products. The same manufacturer may make several different ranges of products, each with its own

catalogue. Thus, an instance of organization may be referenced by several instances of `item_ref_source_proprietary`.

manufacturers_range

A short text reference which is used to identify the range of proprietary products with which the `item_ref_source_proprietary` is concerned.

year_of_range

Declares the instance of `year_number` associated with the `item_ref_source_proprietary`, which provides the integer value of the year in which this version of the range of proprietary products (and its associated ‘catalogue’) was created. If the range has not been given a version, this date provides the date when the range was created

version_of_range

An optional short text reference that is used to identify the version of the range of proprietary products and the associated ‘catalogue’ that acts as the source of the item references. A range can have many versions.

Formal propositions:

proprietary_items

The set of instances of `item_reference_proprietary` that reference this instance of `item_ref_source_proprietary` (via the attribute `source`) shall be greater than or equal to zero.

This INVERSE clause has been relaxed for the 2nd Edition, due to the use of `item_ref_source` in entities other than `item_reference`.

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

16.3.14 `item_ref_source_standard`

Entity definition:

A type of `item_ref_source` that provide the defining parameters for a national or international standard (or code of practice) that acts as the source of the item reference information for a range of standard products. The ‘standard’ must be given a label that identifies the standardization organization that publishes the ‘standard’. The ‘standard’ (containing the list of standard products) must be given a name and a date. It may also be given a version. This entity may be referenced by any number of instances of `item_reference_standard`.

This entity provides the source of information for internationally recognised generic product items (such as hot-rolled sections) which comply with formal standards that have international visibility.

EXPRESS specification:

*)

ENTITY `item_ref_source_standard`

SUBTYPE OF (`item_ref_source`);

`standardization_organization` : label;

`name_of_standard` : label;

```

    year_of_standard : year_number;
    version_of_standard : OPTIONAL label;
INVERSE
    standard_items : SET [0:?] OF item_reference_standard FOR source;
END_ENTITY;
(*)

```

Attribute definitions:

standardization_organization

A short text reference that identifies the body or standardization organization that published the national or international standard (or code of practice); e.g. ‘BSI’, ‘ASTM’, ‘CEN’, etc.

name_of_standard

A short text reference that identifies standard (or code of practice); e.g. ‘BS4_Part1’, ‘ASTM_A6M’, ‘EURONORM_19’, etc.

year_of_standard

Provides the integer value of the year the standard was published; e.g. ‘1980’, ‘1994’, ‘1957’, etc.

version_of_standard

An optional short text reference that is used to identify the version of the standard (or code of practice). In any year, a standard may have more than one version.

Formal propositions:

standard_items

The set of instances of item_reference_standard that reference this instance of item_ref_source_standard (via the attribute source) shall be greater than or equal to zero.

This INVERSE clause has been relaxed for the 2nd Edition, due to the use of item_ref_source in entities other than item_reference.

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

16.3.15 item_reference

Entity definition:

This entity provides the item reference for a particular standard, proprietary, or library item. The item_reference is categorized through its SUBTYPES, which describe where the item reference is fully defined (i.e. the source). Where the source of the item reference is unknown or where the item reference is complete and unambiguous within itself, the item_reference will be instanced on its own. The CIS/1 identifiers may be used in this way.

To map from one item_reference to another (e.g. from CIS/1 identifiers to CIS/2 item_references) the group_usage construct should be used with one flavour containing the CIS/1 references and a second flavour containing CIS/2 references. If a ‘one-to-one’ mapping is required, then each of the two flavours should contain only one

item_reference; the first instance containing the CIS/1 identifier and the second containing CIS/2 item_reference.

EXPRESS specification:

```
*)
ENTITY item_reference
SUPERTYPE OF (ONEOF
    (item_reference_standard,
    item_reference_proprietary,
    item_reference_library));
    ref : identifier;
WHERE
    WRI7 : LENGTH(ref) > 0;
END_ENTITY;
(*
```

Attribute definitions:

ref

A short text string that acts as the item reference for the product.

Formal propositions:

WRI7

The length of the string value assigned to the attribute ref shall be greater than zero. That is, the item reference cannot be defined as an empty string.

Informal propositions:

This entity would normally be instanced as one of its SUBTYPEs. If it is instanced on its own, then the source of the item reference is not captured.

Notes:

New for CIS/2.

See diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.16 item_reference_assigned

Entity definition:

This entity provides a unique association between an instance of item_reference and an instance of structural_frame_item. The association provides a reference (or identifier) for the structural_frame_item, so that it may be passed by reference, rather than by attribute value.

This is effectively an intersection entity, which resolves the many-to-many relationship between the entities item_reference and structural_frame_item. The item reference does **not** uniquely identify the instance of structural_frame_item, merely the type of product that it is representing. A product may have many references, and a reference may be used by many products.

EXPRESS specification:

```

*)
ENTITY item_reference_assigned;
    assigned_reference : item_reference;
    assigned_to_item : structural_frame_item;
UNIQUE
    URI3 : assigned_reference, assigned_to_item;
END_ENTITY;
(*

```

Attribute definitions:*assigned_reference*

Declares the instance of *item_reference* associated with the *item_reference_assigned*. It is implied that this *item_reference* is assigned to the associated *structural_frame_item* (or one of its SUBTYPES).

assigned_to_item

Declares the instance of *structural_frame_item* associated with the *item_reference_assigned*. It is implied that this *structural_frame_item* may be referenced by the associated *item_reference*.

Formal propositions:*URI3*

The combination of instances of *item_reference* (referenced by the attribute *assigned_reference*) and *structural_frame_item* (referenced by the attribute *assigned_to_item*) shall be unique to this instance of *item_reference_assigned*. This prevents repetition of the same association. This does not prevent the product (represented by the *structural_frame_item*) possessing many references. Neither does it prevent the *item_reference* being possessed by many instances of *structural_frame_item*.

Notes

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.17 item_reference_library**Entity definition:**

A type of *item_reference* which provides the item reference for a particular ‘library’ item, whose information is defined in the ‘library’ identified through the associated source.

This entity provides the reference for a particular product item that is not explicitly covered in standard or manufacturer’s catalogue. This could include product information that has been placed in a ‘library’ by a third party trade association.

EXPRESS specification:

```

*)
ENTITY item_reference_library
SUBTYPE OF (item_reference);
    source : item_ref_source_library;

```

UNIQUE

URI4 : SELF\item_reference.ref, source;
 END_ENTITY;
 (*

Attribute definitions:**source**

Declares the instance of item_ref_source_library, associated with the item_reference_library, which provides the source of the item reference information. There must be one (and only one) instance of item_ref_source_library referenced by each instance of item_reference_library. An item_ref_source_library can be referenced by any number of instances of item_reference_library.

(In other words, library items must be sourced from a library.)

Formal propositions:**URI4**

The combination of ref (inherited from the SUPERTYPE entity item_reference) and the instance of item_ref_source_library (referenced by the attribute source) shall be unique to this instance of item_reference_library. In other words, the reference must be unique within the source. This prevents repetition of the same association. This does not prevent the same reference from being used by (or contained in) several sources (although this is not recommended).

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.18 item_reference_proprietary**Entity definition:**

A type of item_reference which provides the item reference for a particular ‘proprietary’ item, whose information is defined in the ‘catalogue’ identified through the associated source.

This entity provides the reference for a particular manufacturer-specific proprietary product (such as a cold rolled section) which complies with a manufacturer’s catalogue. This includes products that comply with specifications agreed between a consortium of manufacturers.

EXPRESS specification:

*)
 ENTITY item_reference_proprietary
 SUBTYPE OF (item_reference);
 source : item_ref_source_proprietary;
 UNIQUE
 URI5 : SELF\item_reference.ref, source;
 END_ENTITY;
 (*

Attribute definitions:**source**

Declares the instance of `item_ref_source_proprietary`, associated with the `item_reference_proprietary`, which provides the source of the item reference information. There must be one (and only one) instance of `item_ref_source_proprietary` referenced by each instance of `item_reference_proprietary`. An `item_ref_source_proprietary` can be referenced by any number of instances of `item_reference_proprietary`.

(In other word, proprietary items must be sourced from a manufacturer's catalogue.)

Formal propositions:**URI5**

The combination of `ref` (inherited from the SUPERTYPE entity `item_reference`) and the instance of `item_ref_source_proprietary` (referenced by the attribute `source`) shall be unique to this instance of `item_reference_proprietary`. In other words, the reference must be unique within the source. This prevents repetition of the same association. This does not prevent the same reference from being used by (or contained in) several sources (although this is not recommended).

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.19 item_reference_standard**Entity definition:**

A type of `item_reference` which provides the item reference for a particular 'standard' item, whose information is defined in the national or international standard or code of practice identified through the associated source.

This entity provides the reference for a particular internationally recognised generic product (such as a hot-rolled section) which complies with a formal standard.

EXPRESS specification:

```

*)
ENTITY item_reference_standard
SUBTYPE OF (item_reference);
    source : item_ref_source_standard;
UNIQUE
    URI6 : SELF\item_reference.ref, source;
END_ENTITY;
(*)

```

Attribute definitions:**source**

Declares the instance of `item_ref_source_standard`, associated with the `item_reference_standard`, which provides the source of the item reference information. There must be one (and only one) instance of `item_ref_source_standard` referenced by each instance of `item_reference_standard`. An `item_ref_source_standard` can be referenced by any number of instances of `item_reference_standard`.

(In other word, standard items must be sourced from a national or international standard or code of practice.)

Formal propositions:

URI6

The combination of ref (inherited from the SUPERTYPE entity item_reference) and the instance of item_ref_source_standard (referenced by the attribute source) shall be unique to this instance of item_reference_standard. In other words, the reference must be unique within the source. This prevents repetition of the same association. This does not prevent the same reference from being used by (or contained in) several sources (although this is not recommended).

Notes:

New for CIS/2.

See Diagram 21 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.20 structural_frame_item

Entity definition:

This entity provides the SUPERTYPE construct by which many (generic) items may be defined. The item must be given a number and a name, and may be given a description.

This is a high level generic entity – from which many more specific SUPERTYPE-SUBTYPE hierarchies are derived.

By using the entity item_reference_assigned, the structural_frame_item may be given an item reference so that it can be passed implicitly by reference rather than explicitly by its attribute values.

EXPRESS specification:

*)

```
ENTITY structural_frame_item
SUPERTYPE OF (ONEOF
    (building,
    building_complex,
    feature,
    joint_system,
    located_item,
    material,
    project,
    project_plan,
    project_plan_item,
    section_profile,
    site,
    structural_frame_process,
    structural_frame_product,
    structure));
item_number : INTEGER;
item_name : label;
item_description : OPTIONAL text;
```

DERIVE

```

item_ref : BAG OF identifier := get_item_ref(SELF);
cost_code : BAG OF label := get_item_cost_code(SELF);
object_id : globally_unique_id := get_instance_id(SELF);

```

WHERE

```

WRS49 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.
MANAGED_DATA_ITEM.DATA_ITEM')) <= 1;

```

END_ENTITY;

(*)

Attribute definitions:*item_number*

A numerical reference that is used to identify the structural_frame_item.

item_name

A short text reference that is used to identify the structural_frame_item.

item_description

An optional text description of the structural_frame_item.

item_ref

Derives the bag of standard references (e.g. from a flavour file) for the structural_frame_item if any references have been provided. The value is derived from the attribute ref in the entity item_reference using the function get_item_ref. The association between the instances of structural_frame_item and item_reference is made by the entity item_reference_assigned.

If a standard reference has not been provided, the function returns an empty bag. This should not be regarded as an error. However, some (non-conforming³) receiving applications may only be able to import data 'by reference', and the lack of reference would result in loss of information.

It is possible that more than one reference has been assigned to the structural_frame_item. Although this is not wrong, it may give rise to ambiguity in the receiving application.

cost_code

Derives the bag of standard cost codes (e.g. from a company's database or MIS system) for the structural_frame_item if any cost codes have been provided. The value is derived from the attribute cost_code in the entity item_cost_code using the function get_item_cost_code. The association between the instances of structural_frame_item and item_cost_code is made by the entity item_cost_code_assigned.

If a cost code has not been provided, the function returns an empty bag. This should not be regarded as an error. It is possible that more than one cost code has been assigned to the structural_frame_item. Although this is not wrong, it may give rise to ambiguity in the receiving application.

³ To improve the efficiency of data exchange, product items should be 'passed-by-reference' rather than being 'passed-by-attribute-value', as discussed in Volume 5 of the CIS/2 documentation^[9].

object_id

Derives the `globally_unique_id` for the `structural_frame_item` if one has been provided. The value is derived from the attribute `instance_id` in the entity `managed_data_item` using the function `get_instance_id`. The association between the instances of `structural_frame_item` and `managed_data_item` is made via the select type `select_data_item` referenced by the attribute `data_item`.

If a `globally_unique_id` has not been assigned to the `structural_frame_item` then the sending application does not have a DMC translator, and the function returns a value of 'UNMANAGED'. If more than one `globally_unique_id` has been assigned to the `structural_frame_item` (due to a DMC translator not working correctly), the function returns an indeterminate value (?). This should be regarded as an error.

Formal propositions:*WR49*

The size of the bag of instances of `managed_data_item` that reference this instance of `structural_frame_item` (via the attribute `data_item`) shall be less than or equal to one. In other words, a `structural_frame_item` can have one unique identifier or none at all.

Notes:

New for CIS/2.

See Diagram 20 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition – DERIVE and WHERE clauses added.

16.3.21 structural_frame_item_approved**Entity definition:**

This entity provides a unique association between an instance of `structural_frame_item` and an instance of approval. The association provides the approval level for the product data represented by the instance of `structural_frame_item`. This is effectively an intersection entity which resolves the many-to-many relationship between the entities `structural_frame_item` and approval.

According to Clause 4.5.3.2 of STEP Part 41, an approval is “a confirmation of the quality of the product data that it is related to.” (Note the emphasis on the product **data**, rather than the product.)

EXPRESS specification:

*)

ENTITY `structural_frame_item_approved`;

`approved_item` : `structural_frame_item`;

`assigned_approval` : approval;

UNIQUE

 URS1 : `approved_item`, `assigned_approval`;

END_ENTITY;

(*

Attribute definitions:*approved_item*

Declares the instance of structural_frame_item (or one of its SUBTYPES) associated with the structural_frame_item_approved. It is implied that this structural_frame_item possesses the approval level that is being assigned.

assigned_approval

Declares the instance of approval associated with the structural_frame_item_approved. It is implied that this approval applies to the associated instance of structural_frame_item.

Formal propositions:*URS1*

The combination of instances of structural_frame_item (referenced by the attribute approved_item) and approval (referenced by the attribute assigned_approval) shall be unique to this instance of structural_frame_item_approved. This prevents repetition of the same association. This does not prevent the structural_frame_item possessing many approvals. Neither does it prevent the approvals being possessed by many instances of structural_frame_item.

Notes

New for CIS/2.

The entity approval is defined in STEP Part 41.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.22 structural_frame_item_certified**Entity definition:**

This entity provides a unique association between an instance of structural_frame_item and an instance of certification. The association provides the certification for the product data represented by the instance of structural_frame_item. This is effectively an intersection entity which resolves the many-to-many relationship between the entities structural_frame_item and certification.

According to Clause 4.4.1 of STEP Part 41, certification “assures and validates product data”. The entity certification represents “documentation which asserts facts.” For example, a material certificate states the chemical composition of one or more physical pieces of material. “The presence of the material certificate removes the need to test the composition of the material; it allows the specified material composition to be accepted as fact without further investigation.”

EXPRESS specification:

*)

```
ENTITY structural_frame_item_certified;
    certified_item : structural_frame_item;
    assigned_certification : certification;
UNIQUE
    URS2 : certified_item, assigned_certification;
END_ENTITY;
(*)
```

Attribute definitions:*certified_item*

Declares the instance of structural_frame_item (or one of its SUBTYPES) associated with the structural_frame_item_certified. It is implied that this structural_frame_item possesses the certification that is being assigned.

assigned_certification

Declares the instance of certification associated with the structural_frame_item_approved. It is implied that this certification applies to the associated instance of structural_frame_item.

Formal propositions:*URS2*

The combination of instances of structural_frame_item (referenced by the attribute certified_item) and certification (referenced by the attribute assigned_certification) shall be unique to this instance of structural_frame_item_certified. This prevents repetition of the same association. This does not prevent the structural_frame_item possessing many certifications. Neither does it prevent a certification being possessed by many instances of structural_frame_item.

Notes

New for CIS/2.

The entity certification is defined in STEP Part 41.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.23 structural_frame_item_documented**Entity definition:**

This entity provides a unique association between an instance of structural_frame_item and an instance of document_usage_constraint. The association provides the document reference for the product data represented by the instance of structural_frame_item. This is effectively an intersection entity that resolves the many-to-many relationship between the entities structural_frame_item and certification.

Clause 4.2.3.4 of STEP Part 41 defines the entity document_usage_constraint as identifying “a specific subject area or aspect from within a document and gives the relevant information or text which applies”. It can be used to define the clauses of a document that are relevant. Note, that it includes the text of the referenced clause, not just the reference to the clause.

EXPRESS specification:

*)

ENTITY structural_frame_item_documented;

documented_item : structural_frame_item;

document_reference : document_usage_constraint;

UNIQUE

URS3 : documented_item, document_reference;

END_ENTITY;

(*

Attribute definitions:*documented_item*

Declares the instance of `structural_frame_item` (or one of its SUBTYPES) associated with the `structural_frame_item_documented`. It is implied that an aspect of this `structural_frame_item` is described by the document reference that is being assigned.

document_reference

Declares the instance of `document_usage_constraint` associated with the `structural_frame_item_documented`. It is implied that this document reference contains a description of an aspect of the associated instance of `structural_frame_item`.

Formal propositions:*URS3*

The combination of instances of `structural_frame_item` (referenced by the attribute `documented_item`) and `document_usage_constraint` (referenced by the attribute `document_reference`) shall be unique to this instance of `structural_frame_item_documented`. This prevents repetition of the same association. This does not prevent the `structural_frame_item` possessing many document references. Neither does it prevent a document reference being possessed by many instances of `structural_frame_item`.

Notes

New for CIS/2.

The entity `document_usage_constraint` is defined in STEP Part 41.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.24 structural_frame_item_priced**Entity definition:**

This entity provides a unique association between an instance of `structural_frame_item` and an instance of `currency_measure_with_unit`. The association provides the cost or price for the product represented by the instance of `structural_frame_item`. This is effectively an intersection entity that resolves the many-to-many relationship between the entities `structural_frame_item` and `currency_measure_with_unit`.

The `assigned_price` is expected to have a real value rather than a codified value. To assign a cost code to a `structural_frame_item`, use the entity `item_cost_code_assigned`.

EXPRESS specification:

*)

```
ENTITY structural_frame_item_priced;
    priced_item : structural_frame_item;
    assigned_price: currency_measure_with_unit;
    price_description : text;
```

UNIQUE

```
    URS4 : priced_item, assigned_price;
```

```
END_ENTITY;
```

(*

Attribute definitions:*priced_item*

Declares the instance of `structural_frame_item` (or one of its SUBTYPES) associated with the `structural_frame_item_priced`. It is implied the cost or price of this `structural_frame_item` is described by the associated instance of `currency_measure_with_unit`.

assigned_price

Declares the instance of `currency_measure_with_unit` that may be associated with the `structural_frame_item_priced` to provide the price (or cost) of the `structural_frame_item`.

price_description

A free text description of the price associated with the `structural_frame_item`; e.g. “cost price”, “purchase price”.

Formal propositions:*URS4*

The combination of instances of `structural_frame_item` (referenced by the attribute `priced_item`) and `currency_measure_with_unit` (referenced by the attribute `assigned_price`) shall be unique to this instance of `structural_frame_item_priced`. This prevents repetition of the same association. This does not prevent the `structural_frame_item` having many prices. Neither does it prevent a price being assigned to many instances of `structural_frame_item`.

Notes:

New for CIS/2.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

16.3.25 structural_frame_item_relationship**Entity definition:**

This entity provides the mechanism that allows two separate instances of `structural_frame_item` to be related to each other. The relationship must be given a name and may also be given a description.

EXPRESS specification:

*)

```
ENTITY structural_frame_item_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    related_item : structural_frame_item;
    relating_item : structural_frame_item;
```

WHERE

```
    WRS29 : related_item :<>: relating_item;
```

```
END_ENTITY;
```

(*)

Attribute definitions:*relationship_name*

A short text reference used to identify the structural_frame_item_relationship.

relationship_description

An optional text description of the structural_frame_item_relationship.

related_item

Declares the first instance of structural_frame_item associated with the structural_frame_item_relationship, which refers to the item for which a relationship is being defined. There must be one (and one only) instance of structural_frame_item referenced by each instance of structural_frame_item_relationship, which acts as the related process. The same item may be involved in any number of relationships and thus an instance of structural_frame_item may be referenced by any number of instances of structural_frame_item_relationship.

relating_item

Declares the second instance of structural_frame_item associated with the structural_frame_item_relationship, which refers to the item which has a relationship with the related_item. There must be one (and one only) instance of structural_frame_item referenced by each instance of structural_frame_item_relationship, which acts as the relating item. The same item may be involved in any number of relationships and thus an instance of structural_frame_item may be referenced by any number of instances of structural_frame_item_relationship.

Formal propositions:**WRS29**

The instance of structural_frame_item referenced by the attribute related_item shall not be instance equal to the instance of structural_frame_item referenced by the attribute relating_item. In other words, an instance of structural_frame_item cannot be related to itself.

Notes

New for CIS/2.

See Diagram 74 of the EXPRESS-G diagrams in Appendix B.

16.4 Item References function definitions

Item References is one of only two subject areas that have EXPRESS functions defined within them. All other functions are imported from STEP Parts 41, 42, 43, and 44 via schema interfacing.

The following functions have been added to this subject area for the 2nd Edition of CIS/2:

- get_instance_id
- get_item_cost_code
- get_item_ref

16.4.1 get_instance_id

Function definition:

This function calculates the number of times an entity instance is used by all the instances of managed_data_item in the information base, and then builds a bag containing those instances.

If the bag is empty, the instance of the entity under test has not been referenced by any instances of managed_data_item (the data is unmanaged) and the function returns a value of 'UNMANAGED'.

If the bag contains only one instance, the entity under test has been referenced by only one instance of managed_data_item (the data is managed) and the function returns the value of the instance_id from that managed_data_item.

If the bag contains more than one instance, the DMC translator is not working correctly and an indeterminate value (?) is returned.

This function is called by the entity structural_frame_item, to retrieve the object id (globally_unique_id) of the structural_frame_item instance.

EXPRESS specification:

```
*)
FUNCTION get_instance_id
  (obj : select_data_item) : globally_unique_id;

  LOCAL
    id_bag : BAG OF managed_data_item :=
      (USEDIN (obj,
        'STRUCTURAL_FRAME_SCHEMA.' +
        'MANAGED_DATA_ITEM.' +
        'DATA_ITEM'));
    n : INTEGER;
  END_LOCAL;

  n := SIZEOF(id_bag);

  CASE n OF
    0 : RETURN ('UNMANAGED');
    1 : RETURN (id_bag[1].instance_id);
    OTHERWISE : RETURN (?);
  END_CASE;

END_FUNCTION;
(*
```

Argument definitions:

obj

The instance of the entity under test. This must be one of the items in the select list of the type select_data_item.

Notes

New for CIS/2 2nd Edition.

16.4.2 get_item_cost_code**Function definition:**

This function calculates the number of times an instance of the entity `structural_frame_item` is used by all the instances of `item_cost_code_assigned` in the information base, and then builds a bag containing those instances.

If the bag is empty, the instance of the entity under test has not been referenced by any instances of `item_cost_code_assigned` (the `structural_frame_item` has not been assigned a cost code) and the function returns an empty bag.

If the bag contains one or more instances, the entity under test has been referenced by one or more instances of `item_cost_code_assigned` (the `structural_frame_item` has been assigned at least one cost code) and the function returns the bag populated with the values of the cost code from the related instances of `item_cost_code`.

This function is called by the entity `structural_frame_item` to retrieve the cost code (or cost codes for the `structural_frame_item` instance).

EXPRESS specification:

*)

FUNCTION `get_item_cost_code`

(`item` : `structural_frame_item`) : BAG OF label;

LOCAL

`i` : INTEGER;

`cost_codes` : BAG OF label := [];

`item_assignment` : BAG OF `item_cost_code_assigned` :=

(USEDIN (`item`,
'STRUCTURAL_FRAME_SCHEMA.' +
'ITEM_COST_CODE_ASSIGNED.' +
'COSTED_ITEM'));

END_LOCAL;

IF SIZEOF (`item_assignment`) > 0 THEN

REPEAT `i` := 1 to HIINDEX (`item_assignment`);

`cost_codes` := `cost_codes` + `item_assignment`[`i`].`code` \| `item_cost_code`.`cost_code`;

END_REPEAT;

END_IF;

RETURN (`cost_codes`);

END_FUNCTION;

(*

Argument definitions:

item

The instance of the `structural_frame_item` under test.

Notes

New for CIS/2 2nd Edition.

16.4.3 get_item_ref**Function definition:**

This function calculates the number of times an instance of the entity `structural_frame_item` is used by all the instances of `item_reference_assigned` in the information base, and then builds a bag containing those instances.

If the bag is empty, the instance of the entity under test has not been referenced by any instances of `item_reference_assigned` (the `structural_frame_item` has not been assigned a standard reference from a flavour file) and the function returns an empty bag.

If the bag contains one or more instances, the entity under test has been referenced by one or more instances of `item_reference_assigned` (the `structural_frame_item` has been assigned at least one standard reference) and the function returns the bag populated with the values of the reference from the related instances of `item_reference`.

This function is called by the entity `structural_frame_item` to retrieve the standard reference (or references) from a flavour file for the `structural_frame_item` instance.

EXPRESS specification:

*)

FUNCTION `get_item_ref`

(`item` : `structural_frame_item`) : BAG OF identifier;

LOCAL

`i` : INTEGER;

`refs` : BAG OF identifier := [];

`item_assignment` : BAG OF `item_reference_assigned` :=

(USEDIN (`item`,
'STRUCTURAL_FRAME_SCHEMA.' +
'ITEM_REFERENCE_ASSIGNED.' +
'ASSIGNED_TO_ITEM'));

END_LOCAL;

IF SIZEOF (`item_assignment`) > 0 THEN

REPEAT `i` := 1 to HIINDEX (`item_assignment`);

`refs` := `refs` + `item_assignment`[`i`].`assigned_reference`\`item_reference.ref`;

END_REPEAT;

END_IF;

RETURN (`refs`);

END_FUNCTION;

(*

Argument definitions:

item

The instance of the `structural_frame_item` under test.

Notes

New for CIS/2 2nd Edition.

17 LPM/6 PRODUCT DEFINITION

17.1 Product Definition concepts and assumptions

17.1.1 What is a product?

The Product Definition subject area covers the definition of generic products. The products are the result of (or are produced by) the processes described in the Process Definition subject area.

The use of the word product underlines the fact that the CIS is one of the ‘new generation’ of data exchange standards that are intended to enable software to share engineering information, rather than simply graphical representations. STEP Part 1 defines the term product as “a thing or substance produced by a natural or artificial process”. In STEP Part 41 the entity product is defined as “... the identification and description, in an application context of a physically realizable object that is produced by a process”.

17.1.2 Product references

As `structural_frame_product` is a SUBTYPE of a `structural_frame_item` it may also be instanced as a `structural_frame_item_with_reference`. In this case, the information relating to that product may be passed implicitly by reference rather than explicitly by the attribute values. Thus, the more detailed information (some of which may be contained in the SUBTYPES of `structural_frame_product`) does not need to be exchanged, as that information is contained in the external source. For example, the physical characteristics of a proprietary part may be found in a manufacturer’s catalogue and so do not need to be repeated during the data exchange. If they are to process that data correctly, both sending and receiving applications involved in the data exchange must have access to the information contained in the external source that provided the reference.

The primary distinction between a `structural_frame_product` and its SUPERTYPE `structural_frame_item` is that a `structural_frame_product` may be given a material. In other words, a product is an item with a material.

Not all of the SUBTYPES of `structural_frame_product` are described in this section. Many, such as part, assembly, and fastener, have their own subject area.

The use of ‘item references’ has significant implications for implementation. Software developers are recommended to read the ‘Conformance Requirements’^[9] and the ‘Implementation Guide’^[7] before writing their translator coding.

17.2 Product Definition type definitions

No types have been modified or added in this subject area for the 2nd Edition.

17.2.1 coating_purpose

Type definition:

Classifies the purpose of the applied coating as providing either corrosion or fire protection, or being simply aesthetic. The purpose of the coating may be classed as undefined.

EXPRESS specification:

```

*)
TYPE coating_purpose
= ENUMERATION OF
    (corrosion_protection,
     fire_protection,
     aesthetic,
     undefined);
END_TYPE;
(*

```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

17.3 Product Definition entity definitions

No entities have been modified or added in this subject area for the 2nd Edition.

17.3.1 coating

Entity definition:

A coating is a product that is applied to the surface of a part or assembly. Examples of coatings include paint and fire protection. A coating may be applied through an association with `surface_treatment_coat` (a SUBTYPE of `structural_frame_process`). For example, a coating may be brushed, sprayed, dipped or electroplated onto a product.

As a SUBTYPE of `structural_frame_product`, a coating inherits the three attributes `item_number`, `item_name`, and `item_description` (from `structural_frame_item`). It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a coating may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`) – this allows the appropriate national standard or code of practice used in the definition of the coating to be described in detail
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A coating may also be related to another coating (or any other `structural_frame_item`) via the entity `structural_frame_item_relationship`.

As a SUBTYPE of `structural_frame_product`, a coating may also be assigned a material (via the entity `structural_frame_product_with_material`).

EXPRESS specification:

```

*)
ENTITY coating
SUBTYPE OF (structural_frame_product);

```

```

    primary_purpose : coating_purpose;
END_ENTITY;
(*)

```

Attribute definitions:

primary_purpose

Declares the classification of the applied coating by its primary purpose. The coating may be classed as providing either corrosion or fire protection, or being simply aesthetic. The purpose of the coating may be classed as undefined.

Notes:

New for CIS/2.

See Diagram 20 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

17.3.2 structural_frame_product

Entity definition:

A type of structural_frame_item that provides the identification and description, in an application context of a physically realizable object that is produced by a process. Through the ANDOR SUPERTYPE construct, any of the SUBTYPEs of structural_frame_product may be given a material.

As a SUBTYPE of structural_frame_item, a structural_frame_product inherits the three attributes item_number, item_name, and item_description. It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a structural_frame_product may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the product to be described in detail
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A structural_frame_product may also be related to an other structural_frame_product (or any other structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

*)

```

ENTITY structural_frame_product
SUPERTYPE OF (ONEOF
    (assembly,
    coating,
    part,
    fastener,
    fastener_mechanism,

```

```

        weld_mechanism,
        chemical_mechanism) ANDOR
        structural_frame_product_with_material)
SUBTYPE OF (structural_frame_item);
    life_cycle_stage : OPTIONAL label;
END_ENTITY;
(*)

```

Attribute definitions:*life_cycle_stage*

A free text description of the stage in the life-cycle at which the product information is defined; e.g. “as planned”, “as designed”, “as built”, etc.

Formal propositions:**ANDOR SUPERTYPE**

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instanced with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

Notes:

New for CIS/2.

See Diagram 20 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

17.3.3 structural_frame_product_with_material**Entity definition:**

A type of structural_frame_product which is associated with a material. It may also be given a nominal and actual mass.

As a SUBTYPE of structural_frame_product, an structural_frame_product_with_material inherits the three attributes item_number, item_name, and item_description (from structural_frame_item). It also inherits the many implicit relationships that exist because other entities use structural_frame_item as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a structural_frame_product_with_material may have:

- approvals (via the entity structural_frame_item_approved)
- prices (via the entity structural_frame_item_priced)
- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented) – this allows the appropriate national standard or code of practice used in the definition of the structural_frame_product_with_material to be described in detail
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

An structural_frame_product_with_material may also be related to another structural_frame_product_with_material (or any other structural_frame_item) via the entity structural_frame_item_relationship.

As a SUBTYPE of `structural_frame_product`, this entity is used to assign a material to a product (and any SUBTYPE of `structural_frame_product`).

EXPRESS specification:

*)

ENTITY `structural_frame_product_with_material`

SUBTYPE OF (`structural_frame_product`);

`material_definition` : `material`;

`nominal_mass` : OPTIONAL `mass_measure_with_unit`;

`actual_mass` : OPTIONAL `mass_measure_with_unit`;

END_ENTITY;

(*

Attribute definitions:

material_definition

Declares the instance of `material` associated with the `structural_frame_product_with_material`, which provides the specification of the material of the `structural_frame_product`. There must be one (and only one) instance of `material` referenced by each instance of `structural_frame_product_with_material`. A `material` may be referenced by any number of instances of `structural_frame_product_with_material`.

nominal_mass

Declares the first instance of `mass_measure_with_unit` that may be associated with the `structural_frame_product_with_material` to provide the nominal mass of the `structural_frame_product`. This is an approximate value that may be used to estimate the total mass of a structure and its constituent assemblies and parts.

actual_mass

Declares the second instance of `mass_measure_with_unit` that may be associated with the `structural_frame_product_with_material` to provide the actual mass of the `structural_frame_product`. This is a measured or calculated value that may be used to calculate the total mass of a structure and its constituent assemblies and parts.

Notes:

New for CIS/2.

See Diagram 20 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18 LPM/6 PROCESS DEFINITION

18.1 Process Definition concepts and assumptions

The Process Definition subject area provides constructs to represent the generic processes that are involved in the construction of a steel framed building. The processes produce the products defined in the Product Definition subject area.

18.2 Process Definition type definitions

The following types have been added to this subject area for the 2nd Edition of CIS/2:

- brazing_type
- product_item_select
- soldering_type
- welding_type_arc
- welding_type_beam
- welding_type_gas
- welding_type_other
- welding_type_pressure
- welding_type_resistance
- welding_type_stud

The following types have been modified in this subject area for the 2nd Edition of CIS/2:

- assembly_component_select
- welding_type

18.2.1 assembly_component_select

Type definition:

Selects the component of the assembly from one of the ‘physical model’ entities. This is a selection from a located_assembly, a located_part, a located_feature, or a located_joint_system. That is, the assembly process involves one or more of these items.

EXPRESS specification:

*)

TYPE assembly_component_select

= SELECT

(located_assembly,
located_part,
located_feature,
located_joint_system);

END_TYPE;

(*

Notes:

New for CIS/2. Modified for 2nd Edition.

18.2.2 bending_method

Type definition:

Classifies the method of bending as either, hot, cold, or undefined.

EXPRESS specification:

```
*)  
TYPE bending_method  
= ENUMERATION OF  
    (hot_bend, cold_bend, undefined);  
END_TYPE;  
(*
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

18.2.3 brazing_type

Type definition:

Classifies the method of brazing in accordance with the American Welding Society's Welding Handbook^[1].

EXPRESS specification:

```
*)  
TYPE brazing_type  
= ENUMERATION OF  
    (diffusion_brazing,  
    dip_brazing,  
    furnace_brazing,  
    induction_brazing,  
    infrared_brazing,  
    resistance_brazing,  
    torch_brazing);  
END_TYPE;  
(*
```

Notes:

New for 2nd Edition.

18.2.4 cleaning_method

Type definition:

Classifies the method of cleaning as being either a chemical wash, a blast clean, or undefined.

EXPRESS specification:

```
*)  
TYPE cleaning_method  
= ENUMERATION OF
```

```
(chemical_wash, blast_clean, undefined);
END_TYPE;
(*)
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

18.2.5 coating_method

Type definition:

Classifies the method of application of the coating as either sprayed, brushed, dipped, electroplated, or undefined.

EXPRESS specification:

```
*)
TYPE coating_method
= ENUMERATION OF
    (sprayed, brushed, dipped, electroplated, undefined);
END_TYPE;
(*)
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

18.2.6 cutting_type

Type definition:

Classifies the method used to make a cut as being either sawn, flame cut, sheared, punched, drilled, laser cut, or cut by abrasion. The method used to make a cut may be classed as undefined.

EXPRESS specification:

```
*)
TYPE cutting_type
= ENUMERATION OF
    (sawn, flame_cut, sheared, punched, drilled, laser, abrasion, undefined);
END_TYPE;
(*)
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

18.2.7 product_item_select

Type definition:

Selects the product or the component of the assembly from one of the ‘physical model’ entities. This is a selection from a structural_frame_product (or its subtypes) a located_assembly, a located_part, a located_feature, or a located_joint_system. That is, a process involves one or more of these items.

EXPRESS specification:

```
*)
TYPE product_item_select
= SELECT
```

```
(structural_frame_product, assembly_component_select);
END_TYPE;
(*
```

Notes:

New for CIS/2 2nd Edition.

18.2.8 soldering_type

Type definition:

Classifies the method of soldering in accordance with the American Welding Society's Welding Handbook^[1].

EXPRESS specification:

```
*)
TYPE soldering_type
= ENUMERATION OF
    (dip_soldering,
     furnace_soldering,
     induction_soldering,
     infrared_soldering,
     iron_soldering,
     resistance_soldering,
     torch_soldering,
     wave_soldering);
END_TYPE;
(*
```

Notes

New for CIS/2 2nd Edition.

18.2.9 welding_type

Type definition:

Classifies the method used to make a (generic) welding process as being made by one of the methods stated in the enumeration list (e.g. fusion, friction, flash, laser, forge). The method used to make a weld may be classed as undefined.

EXPRESS specification:

```
*)
TYPE welding_type
= ENUMERATION OF
    (fusion_weld,
     friction_weld,
     flash_weld,
     laser_weld,
     forge_weld,
     undefined);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged for 2nd Edition.

18.2.10 welding_type_arc**Type definition:**

Classifies the method used to make an arc welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.1.

Table 18.1: Mapping the enumeration items of welding_type_arc

TYPE	ISO 4063 Equivalent		AWS Equivalent	BS 499 Equivalent
generic_arc_welding	1	Arc Welding	Arc welding	Arc Welding
metal_arc_welding	101	Metal arc welding		
manual_metal_arc_welding	111	Manual Metal arc welding	SMAW Shielded Metal Arc	Manual metal-arc welding
gravity_arc_welding	112	Gravity arc welding with covered electrode		
self_shielded_arc_welding	114	Self-shielded tubular-cored arc welding		
submerged_arc_welding	12	Submerged arc welding	SAW Submerged Arc	Submerged-arc welding
gas_shielded_metal_arc_welding	13	Gas-shielded metal arc welding	GMAW Gas Metal Arc	
metal_inert_gas_welding	131	Metal inert gas welding	GMAW Gas Metal Arc	metal inert gas welding
metal_active_gas_welding	135	Metal active gas welding	GMAW Gas Metal Arc	metal active-gas welding
tubular_inert_gas_welding	136	Tubular cored metal arc welding with active gas shield	FCAW Flux Cored Arc	
tubular_active_gas_welding	137	Tubular cored metal arc welding with inert gas shield	FCAW Flux Cored Arc	
tungsten_inert_gas_welding	141	Tungsten inert gas welding	GTAW Gas Tungsten Arc	Tungston inert-gas welding
atomic_hydrogen_welding	149	Atomic-hydrogen welding (obsolete)		atomic-hydrogen welding
plasma_arc_welding	15	Plasma arc welding	PAW Plasma Arc	arc plasma welding
carbon_arc_welding	181	Carbon-arc welding (obsolete)	CAW Carbon Arc	Carbon-arc welding
magnetically_impelled_arc_butt_welding	185	Magnetically impelled arc butt welding		magnetically-impelled arc butt welding

EXPRESS specification:

*)

TYPE welding_type_arc

= ENUMERATION OF

(generic_arc_welding,
 metal_arc_welding,
 manual_metal_arc_welding,
 gravity_arc_welding,
 self_shielded_arc_welding,
 submerged_arc_welding,
 gas_shielded_metal_arc_welding,
 metal_inert_gas_welding,
 metal_active_gas_welding,
 tubular_inert_gas_welding,
 tubular_active_gas_welding,
 tungsten_inert_gas_welding,
 atomic_hydrogen_welding,
 plasma_arc_welding,
 carbon_arc_welding,
 magnetically_impelled_arc_buttwelding);

END_TYPE;

(*)

NotesNew for CIS/2 2nd Edition.**18.2.11 welding_type_beam****Type definition:**

Classifies the method used to make a beam welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.2.

Table 18.2: Mapping the enumeration items of welding_type_beam

TYPE	ISO4063 Equivalent	AWS Equivalent	BS 499 Equivalent
	5 Beam welding		
electron_beam_welding	51 Electron beam welding	EBW Electron beam	Electron beam welding
laser_beam_welding	52 Laser welding	LBW Laser beam	Laser welding
gas_laser_welding	522 Gas laser welding		

EXPRESS specification:

*)

TYPE welding_type_beam

= ENUMERATION OF

(electron_beam_welding,
 laser_beam_welding,
 gas_laser_welding);

END_TYPE;
(*

Notes

New for CIS/2 2nd Edition.

18.2.12 welding_type_gas

Type definition:

Classifies the method used to make a gas welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.3.

Table 18.3: Mapping the enumeration items of welding_type_gas

TYPE	ISO4063 Equivalent	AWS Equivalent	BS 499 Equivalent
generic_gas_welding	3 Gas Welding	Oxyfuel gas	
oxyacetylene_welding	311 Oxy-acetylene welding	OAW Oxyacetylene	Gas welding
oxyhydrogen_welding	312 Oxy-hydrogen welding	OHW Oxyhydrogen	Gas welding
oxypropane_welding	313 Oxy-propane welding		Gas welding

EXPRESS specification:

*)
TYPE welding_type_gas
= ENUMERATION OF
(generic_gas_welding,
oxyacetylene_welding,
oxyhydrogen_welding,
oxypropane_welding);
END_TYPE;
(*

Notes

New for CIS/2 2nd Edition.

18.2.13 welding_type_other

Type definition:

Classifies the method used to make a welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.4.

EXPRESS specification:

*)
TYPE welding_type_other
= ENUMERATION OF
(aluminothermic_welding,
electroslag_welding,
electrogas_welding,
induction_welding,


```

induction_butt_welding,
induction_seam_welding,
infrared_welding,
percussion_welding);
END_TYPE;

```

(*

Table 18.4: *Mapping the enumeration items of welding_type_other*

TYPE	ISO4063 Equivalent		AWS Equivalent		BS 499 Equivalent
	7	Other welding processes			
aluminothermic welding	71	Aluminothermic welding	TW	Thermit	Aluminothermic welding
electroslag_welding	72	Electroslag welding	ESW	Electroslag	Electro-slag welding
electrogas_welding	73	Electogas welding			Electrogas welding
induction_welding	74	Induction welding	IW	Induction	
induction_butt_welding	741	Induction butt welding			
induction_seam_welding	742	Induction seam welding			
infrared_welding	753	Infrared welding			
percussion_welding	77	Percussion welding	PEW	Percussion	Percussion welding

Notes

New for CIS/2 2nd Edition.

18.2.14 welding_type_pressure

Type definition:

Classifies the method used to make a pressure welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.5.

EXPRESS specification:

*)

```
TYPE welding_type_pressure
```

```
= ENUMERATION OF
```

```

(generic_pressure_welding,
ultrasonic_welding,
friction_welding,
forge_welding,
explosive_welding,
diffusion_welding,
oxyfuel_gas_pressure_welding,
cold_pressure_welding,
hot_pressure_welding,
roll_welding,
high_frequency_pressure_welding);

```

```
END_TYPE;
```

(*)

Table 18.5: Mapping the enumeration items of welding_type_pressure

TYPE	ISO4063 Equivalent	AWS Equivalent	BS 499 Equivalent
generic_pressure_welding	4 Welding with pressure	Solid State Welding	
ultrasonic_welding	41 Ultrasonic welding	USW Ultrasonic	Ultrasonic welding
friction_welding	42 Friction welding	FRW Friction	Friction welding
forge_welding	43 Forge Welding (obsolete)	FOW Forge	Forge welding
explosive_welding	441 Explosive welding	EXW Explosion	Explosive welding
diffusion_welding	45 Diffusion welding	DFW Diffusion	Diffusion welding
oxyfuel_gas_pressure_welding	47 Oxy-fuel gas pressure welding	PGW Pressure gas	Oxy-acetylene pressure welding
cold_pressure_welding	48 Cold pressure welding	CW Cold	Cold welding
hot_pressure_welding		HPW Hot pressure	
roll_welding		ROW Roll	
high_frequency_pressure_welding			HF pressure welding

NotesNew for CIS/2 2nd Edition.**18.2.15 welding_type_resistance****Type definition:**

Classifies the method used to make a resistance welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.6.

Table 18.6: Mapping the enumeration items of welding_type_resistance

TYPE	ISO4063 Equivalent	AWS Equivalent	BS 499 Equivalent
generic_resistance_welding	2 Resistance Welding		
spot_welding	21 Spot welding	RSW Resistance-spot	Spot welding
	211 Indirect spot welding	Indirect	
	212 Direct spot welding		
seam_welding	22 Seam welding	RSEW Resistance-seam	Seam welding
projection_welding	23 Projection welding	RPW Projection	Projection welding
flash_welding	24 Flash welding	FW Flash	Flash Welding
resistance_butt_welding	25 Resistance butt welding	UW Upset	Resistance butt Welding
high_frequency_resistance_welding	291 High Frequency resistance welding	HFRW High frequency resistance	HF resistance welding

EXPRESS specification:

```

*)
TYPE welding_type_resistance
= ENUMERATION OF
    (generic_resistance_welding,
    spot_welding,
    seam_welding,
    projection_welding,
    flash_welding,
    resistance_butt_welding,
    high_frequency_resistance_welding);
END_TYPE;
(*

```

Notes

New for CIS/2 2nd Edition.

18.2.16 welding_type_stud**Type definition:**

Classifies the method used to make a stud welding process as being made by one of the methods stated in the enumeration list. A mapping of the enumeration items onto ISO 4063^[34], AWS Welding Handbook^[1], and BS 499^[2] is shown in Table 18.7.

Table 18.7: Mapping the enumeration items of welding_type_stud

TYPE	ISO4063 Equivalent		AWS Equivalent	BS 499 Equivalent
generic_stud_welding	78	Stud welding	SW Stud Arc	
resistance_stud_welding	782	Resistance stud welding		
drawn_arc_stud_welding	783	Drawn arc stud welding with ceramic ferrule or shielding gas		
	784	Short cycle drawn arc stud welding		
	785	Capacitor discharge drawn arc stud welding		
	786	Capacitor discharge stud welding with tip ignition		
	787	Drawn arc stud welding with ceramic fusible collar		
friction_stud_welding	788	Friction stud welding		

EXPRESS specification:

```
*)
TYPE welding_type_stud
= ENUMERATION OF
    (generic_stud_welding,
     resistance_stud_welding,
     drawn_arc_stud_welding,
     friction_stud_welding);
END_TYPE;
(*
```

Notes

New for CIS/2 2nd Edition.

18.3 Process Definition entity definitions

The following have been added to this subject area for the 2nd Edition of CIS/2:

- braze
- solder
- weld_arc
- weld_beam
- weld_gas
- weld_other
- weld_pressure
- weld_resistance
- weld_stud

The following entities have been modified in this subject area for the 2nd Edition:

- dispatch
- procure
- project_process_item
- weld

18.3.1 assemble

Entity definition:

A type of structural_frame_process that produces a single assembly_manufacturing by bringing together a number of components using a number of processes.

EXPRESS specification:

```
*)
ENTITY assemble
SUBTYPE OF (structural_frame_process);
    resulting_assembly : located_assembly;
    components : SET [1:?] OF assembly_component_select;
```

```

    required_processes : SET [1:?] OF structural_frame_process;
WHERE
    WRA29 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN TYPE
        OF (resulting_assembly.descriptive_assembly);
    WRA30 : SIZEOF(QUERY(component <* components | component :=:
        resulting_assembly)) = 0;
    WRA31 : SIZEOF(QUERY(process <* required_processes | process :=: (SELF))) = 0;
END_ENTITY;
(*)

```

Attribute definitions:

resulting_assembly

Declares the instance of *located_assembly* associated with the *assemble*, which is the product of the *structural_frame_process*. There must be one (and only one) instance of *located_assembly* referenced by each instance of *assemble*. An instance of *located_assembly* may be referenced by any number of instances of *assemble*. (That is a sub-assembly may be involved in a number of higher level assemblies requiring a number of processes.)

components

Declares the set of one or more instances of *located_assembly_child*, *located_part*, *located_feature*, or *located_joint_system* associated with the *assemble*. This is effectively the list of components that are required to make up the *assembly_manufacturing* (referenced by the *located_assembly*). Instance recursion is prohibited such that an instance of *assembly_manufacturing* cannot be a component of itself. Entity type recursion is allowed such that an instance of *assembly_manufacturing_child* may be a component of another separate and distinct instance of *assembly_manufacturing_child*.

required_processes

Declares the set of one or more instances of *structural_frame_process* associated with the *assemble*. This is effectively the list of processes (cutting, drilling, welding, etc) that are required to make up the *assembly* (defined in the *assembly_manufacturing*). Instance recursion is prohibited such that an instance of *assemble* cannot require itself as a process. Entity type recursion is allowed such that an instance of *assemble* may require another separate and distinct instance of *assemble* to complete the process.

Formal propositions:

WRA29

The instance of *assembly* referenced by the attribute *descriptive_assembly* (of the entity *located_assembly* referenced by the attribute *resulting_assembly*) shall be of the type *assembly_manufacturing*. In other words, a *assembly* process produces an *assembly_manufacturing*.

WRA30

The number of instances referenced by the attribute *components* that are instance equal to the *located_assembly* referenced by the attribute *resulting_assembly* shall equal zero. That is, the *located_assembly* resulting from this *assemble* process cannot be a member of the aggregation referenced by the attribute *resulting_assembly*. This prevents instance recursion. In other words, an *assembly* cannot be a component of itself.

WRA31

The number of instances referenced by the attribute `required_processes` that are instance equal to this assemble shall equal zero. This prevents instance recursion. In other words, an assemble process cannot require itself as one of its processes.

Notes:

New for CIS/2.

See diagram 59 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

18.3.2 bend**Entity definition:**

A type of `structural_frame_process` that specifies a method of bending that would, for example, be used to bend a part.

EXPRESS specification:

```
*)
ENTITY bend
SUBTYPE OF (structural_frame_process);
    method : bending_method;
END_ENTITY;
(*
```

Attribute definitions:**method**

Declares the classification of the method of bending used to form the bend. The bend may be declared as either a hot or a cold bend. It may be declared as being undefined.

Notes:

New for CIS/2.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

18.3.3 braze**Entity definition:**

A type of `structural_frame_process` that specifies a method of brazing that would, for example, be used to join two parts. Brazing is a welding process which produces coalescence of materials by heating them to a suitable temperature and by using a filler metal, having a liquidus above 450°C and below the solidus of the base materials. The filler metal is distributed between the closely fitted surfaces of the joint by capillary attraction.

EXPRESS specification:

```
*)
ENTITY braze
SUBTYPE OF (structural_frame_process);
    braze_type : brazing_type;
END_ENTITY;
```

(*

Attribute definitions:

braze_type

Declares the classification of the method of brazing used to form the join as described in the definition of the TYPE brazing_type.

Notes:

New for CIS/2 2nd Edition

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.4 cut

Entity definition:

A type of structural_frame_process that classifies the method of a cut.

EXPRESS specification:

```
*)
ENTITY cut
SUBTYPE OF (structural_frame_process);
    cutting_method : cutting_type;
END_ENTITY;
(*
```

Attribute definitions:

cutting_method

Declares the class of the method used to make a cut as being either sawn, flame cut, sheared, punched, drilled, laser cut, or cut by abrasion. The method used to make a cut may be declared as being undefined

Notes:

New for CIS/2.

See diagram 44 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

18.3.5 dispatch

Entity definition:

A type of structural_frame_process representing a dispatch and delivery process. Using this entity, a list of products is assigned dispatch and delivery addresses and dispatch and delivery dates. For example, a batch of steel components may be collected and dispatched from a fabrication shop and delivered to a construction site. Either the dispatch or delivery addresses could represent marshalling yards.

As a SUBTYPE of structural_frame_process, a dispatch inherits an item_number, an item_name, an item_description, and a place_of_process. The organizational_address referenced by the attribute place_of_process may (but need not) be the same as either the dispatch or delivery address. For a dispatch, the place_of_process indicates where the order was given.

No details are given here of the people providing the transport. To include these details, the dispatch must be the subject of a `project_process_item`, which in turn is part of a `project_plan`.

EXPRESS specification:

*)

ENTITY dispatch

SUBTYPE OF (structural_frame_process);

transported_products : LIST [1:?] OF product_item_select;

dispatch_address : organizational_address;

delivery_address : organizational_address;

dispatch_date : calendar_date;

delivery_date : calendar_date;

END_ENTITY;

(*

Attribute definitions:

transported_products

Declares the list of one or more products associated with this dispatch. These are the products to be dispatched from the address given in the attribute `dispatch_address` and delivered to the address given in the attribute `delivery_address`.

The items in the list are selected from instances of `structural_frame_product`, `located_part`, `located_feature`, `located_joint_system`, or `located_assembly` via the select type `product_item_select`.

dispatch_address

Declares the instance of `organizational_address` associated with this dispatch, which defines the address of the organization from which the list of products is to be (or has been) dispatched.

delivery_address

Declares the instance of `organizational_address` associated with this dispatch, which defines the address of the organization to which the list of products is to be (or has been) delivered.

dispatch_date

Declares the instance of `calendar_date` associated with this dispatch, which defines the date when the list of products is to be (or has been) dispatched.

delivery_date

Declares the instance of `calendar_date` associated with this dispatch, which defines the date when the list of products is to be (or has been) delivered.

Notes:

New for CIS/2.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

18.3.6 move

Entity definition:

A type of `structural_frame_process` which describes the relocation of a product. This relocation could occur in the fabrication shop, or on the construction site. It is **not** intended to represent the transportation of items from the fabrication shop to the construction site. The move must be given initial and final locations with reference to two placements. It may also be a path from the initial to final locations using a reference to a curve. The entities `placement` and `curve` are geometric entities taken from (and defined in) STEP Part 42.

EXPRESS specification:

```
*)
ENTITY move
SUBTYPE OF (structural_frame_process);
    initial_location : placement;
    final_location : placement;
    path : OPTIONAL curve;
END_ENTITY;
(*
```

Attribute definitions:

initial_location

Declares the first instance of `placement` associated with the move, which defines the initial location of the associated product. It is assumed that, before the move takes place, the origin of the local coordinate system of the product is located at the Cartesian point referenced by this placement.

final_location

Declares the second instance of `placement` associated with the move, which defines the final location of the associated product. It is assumed that, after the move takes place, the origin of the local coordinate system of the product is located at the Cartesian point referenced by this placement.

path

Declares an instance of `curve` that may be associated with the move to define the path of the move. It is assumed that, during the move, the origin of the local coordinate system of the product is located somewhere on this path.

Informal propositions:

It is assumed that if a path is defined, both the initial and final locations (defined by the two instances of `placement`) lie on that curve.

Notes:

New for CIS/2.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.7 procure

Entity definition:

A type of structural_frame_process whereby a set of one or more products is obtained. This entity allows a procurement transaction to be recorded. The transaction involves one purchaser and one vendor who supplies the product(s). The sales contract must also be recorded, as well as the delivery date(s).

As a SUBTYPE of the entity structural_frame_item, an instance of this entity may be referenced by an instance of structural_frame_item_priced. The price agreed between the parties for the procurement transaction is captured by the associated instance of currency_measure_with_unit.

EXPRESS specification:

*)

ENTITY procure

SUBTYPE OF (structural_frame_process);

vendors : person_and_organization;

purchaser : person_and_organization;

purchased_products : LIST [1:?] OF product_item_select;

sales_contract : contract;

delivery_dates : LIST [1:?] OF calendar_date;

INVERSE

prices : SET [1:?] OF structural_frame_item_priced FOR priced_item;

WHERE

WRP23 : SIZEOF (delivery_dates) = SIZEOF (purchased_products);

END_ENTITY;

(*

Attribute definitions:

vendors

Declares the first instance of person_and_organization, associated with the procure, which act as the vendor of the product(s) in this recorded transaction. There must be at least one instance of person_and_organization referenced by each instance of procure that is deemed to act as the vendor (seller or supplier) of the product(s). The same instance of person_and_organization can be referenced by any number of instances of procure.

purchaser

Declares the second instance of person_and_organization, associated with the procure, which acts as the purchaser of the product(s) in this recorded transaction. There must one (and only one) instance of person_and_organization referenced by each instance of procure that is deemed to act as the purchaser (or buyer) of the product(s). The same instance of person_and_organization can be referenced by any number of instances of procure. The purchaser may (but need not) be a defined by a separate instance of person_and_organization. That is, a purchaser may also act as the vendors in the recorded transaction.

purchased_products

Declares the list of one or more product(s) bought and sold in this recorded transaction. The items in the list are selected from instances of structural_frame_product, located_part, located_feature, located_joint_system, or located_assembly via the select

type `product_item_select`. There must be at least one instance of these entities referenced by each instance of `procure`. The same instance can be referenced by any number of instances of `procure`; i.e. a product can be bought and sold several times.

sales_contract

Declares the instance of contract associated with the `procure`, which describes the contractual arrangements of the transaction.

delivery_dates

Declares the list of one or more instances of `calendar_date`, associated with the `procure`, which are assumed to be the agreed dates for delivery of the product(s) bought and sold in this recorded transaction.

The number of delivery dates must be the same as the number of products involved in the transaction. If the procurement transaction involves only one product, there should be only one specified date, which is assumed to be the date on which the product will be delivered to the purchaser. If a number of products are to be delivered on the same date, then all the references in the list will be to the same instance of `calendar_date`. In this situation, it is implied that the products are batched for delivery, with all the products placed in one batch, which is delivered to the purchaser on that date.

Formal propositions:

prices

Declares the set of one or more instances of `structural_frame_item_priced`, associated with the `procure`, which are assumed to reference the agreed prices for the product(s) bought and sold in this recorded transaction. That is, the entity `procure` is existence dependent upon the entity `structural_frame_item_priced` (and thus, upon `currency_measure_with_unit`). In other words, a procurement transaction cannot exist without being assigned at least one price. There may be more than one price. The price may be a rate, if the SUBTYPE `currency_rate_with_unit` is used in place of `currency_measure_with_unit`.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

18.3.8 project_process_item

Entity definition:

A type of `project_plan_item` that specifies one process in a larger project plan. The process produces a product from the processing of one or more products. Both the processed products and the resulting products must be declared.

As a type of `project_plan_item` a `project_process_item` must be part of a `project_plan`, which in turn is associated with a project. The `project_process_item` must be given a name and number for referencing purposes. The item must also be given a start date, end date and a duration. The item may be given a sequence number - an integer that positions the item in the sequence of items in a project plan.

As a SUBTYPE of `structural_frame_item`, a `project_process_item` inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data

type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a `project_process_item` may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)
- certificates (via the entity `structural_frame_item_certified`)
- document references (via the entity `structural_frame_item_documented`)
- properties (via the entity `item_property_assigned`)
- item_references (via the entity `item_reference_assigned`).

A `project_process_item` may also be related to other SUBTYPEs of `structural_frame_item` via the entity `structural_frame_item_relationship`. However, the more constrained entity `project_plan_item_relationship` should be used to represent a one-to-one relationship between two instances of `project_process_item`.

EXPRESS specification:

```
*)
ENTITY project_process_item
SUBTYPE OF (project_plan_item);
    scheduled_process : structural_frame_process;
    resulting_product : SET [0:?] OF product_item_select;
    processed_products : SET [1:?] OF product_item_select;
END_ENTITY;
(*
```

Attribute definitions:

scheduled_process

Declares the instance of `structural_frame_process` associated with the `project_process_item`, which specifies the process performed and scheduled. There must be one (and only one) instance of `structural_frame_process` referenced by each instance of `project_process_item`. An instance of `structural_frame_process` may be referenced by any number of instances of `project_process_item`; i.e. a specific process may occur many times.

resulting_product

Specifies the set of product(s) produced as a result of the scheduled process. The items in the set are selected from instances of `structural_frame_product`, `located_part`, `located_feature`, `located_joint_system`, or `located_assembly` via the select type `product_item_select`. Any of these instances may be referenced by any number of instances of `project_process_item`; i.e. a product may require many scheduled processes.

processed_products

Specifies the products contributing to the production of the product resulting from the scheduled process. The items in the set are selected from instances of `structural_frame_product`, `located_part`, `located_feature`, `located_joint_system`, or `located_assembly` via the select type `product_item_select`. There must be at least one instance of these entities referenced by each instance of `project_process_item`. Each of these instances may be referenced by any number of instances of `project_process_item`; i.e. a product may contribute to many scheduled processes.

Informal propositions:

Because `project_process_item` is a SUBTYPE of `structural_frame_item` it may be associated with a standardized `item_reference` (via the entity `item_reference_assigned`).

Notes:

New for CIS/2.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition.

18.3.9 solder**Entity definition:**

A type of `structural_frame_process` that specifies a method of soldering that would, for example, be used to join two parts. Soldering is a joining process which produces coalescence of materials by heating them to a suitable temperature and by using a filler metal having a liquidus not exceeding 450°C (840°F) and below the solidus of the base materials. The filler metal is distributed between the closely fitted surfaces of the joint by capillary attraction.

EXPRESS specification:

```
*)
ENTITY solder
SUBTYPE OF (structural_frame_process);
    solder_type : soldering_type;
END_ENTITY;
(*
```

Attribute definitions:*solder_type*

Declares the classification of the method of soldering used to form the join as described in the definition of the TYPE `soldering_type`.

Notes:

New for CIS/2 2nd Edition

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.10 structural_frame_process**Entity definition:**

A type of `structural_frame_item` that provides the identification and description of one of the processes involved in the construction of a steel framed building. The address of the organization performing the process may also be specified.

As a SUBTYPE of `structural_frame_item`, a `structural_frame_process` inherits the three attributes `item_number`, `item_name`, and `item_description`. It also inherits the many implicit relationships that exist because other entities use `structural_frame_item` as a data type. (See Diagram 74 of the EXPRESS-G diagrams in Appendix B.) This means that a `structural_frame_process` may have:

- approvals (via the entity `structural_frame_item_approved`)
- prices (via the entity `structural_frame_item_priced`)

- certificates (via the entity structural_frame_item_certified)
- document references (via the entity structural_frame_item_documented)
- properties (via the entity item_property_assigned)
- item_references (via the entity item_reference_assigned).

A structural_frame_process may also be related to another structural_frame_process (or any other SUBTYPE of structural_frame_item) via the entity structural_frame_item_relationship.

EXPRESS specification:

*)

ENTITY structural_frame_process

SUPERTYPE OF (ONEOF

(assemble,
bend,
braze,
cut,
dispatch,
move,
procure,
solder,
surface_treatment,
weld))

SUBTYPE OF (structural_frame_item);

place_of_process : OPTIONAL organizational_address;

END_ENTITY;

(*

Attribute definitions:

place_of_process

Declares the instance of organizational_address that may be associated with this structural_frame_process where the process takes place; e.g. in the fabrication shop or on the construction site.

Informal propositions:

As this entity gives no details of what the process actually is, it is likely to be instanced as one of its SUBTYPES.

Notes:

New for CIS/2.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition (subtypes added).

18.3.11 surface_treatment

Entity definition:

A type structural_frame_process that provides the identity and description of a number of different processes that may be applied to the surfaces of the components of a steel-framed building. The surface_treatment must be categorized (by being instanced as one

of its SUBTYPEs) as either a cleaning process, a coating process, a grinding, hard stamping or thermal treatment. (See the definitions of the SUBTYPEs.)

EXPRESS specification:

```
*)
ENTITY surface_treatment
ABSTRACT SUPERTYPE OF (ONEOF
    (surface_treatment_clean,
     surface_treatment_coat,
     surface_treatment_grind,
     surface_treatment_hard_stamp,
     surface_treatment_thermal))
SUBTYPE OF (structural_frame_process);
    surface_finish_specification : text;
END_ENTITY;
(*
```

Attribute definitions:

surface_finish_specification

A text description of the final condition of the surface being treated as specified by the appropriate standard or code of practice. For example, the surface finish of a steel beam following a blast cleaning process may be specified as being to the Swedish standard SA2.

Formal propositions:

ABSTRACT SUPERTYPE

As this entity has been declared to be an abstract SUPERTYPE, it cannot be instanced on its own - it must be instanced with one of its SUBTYPEs.

Informal propositions:

Because surface_treatment is a SUBTYPE structural_frame_process (which is a SUBTYPE of structural_frame_item) it may be assigned a standardized item_reference (via the entity item_reference_assigned).

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.12 surface_treatment_clean

Entity definition:

A type of surface_treatment that declares the method of cleaning applied to the component as one of the manufacturing processes.

EXPRESS specification:

```
*)
ENTITY surface_treatment_clean
SUBTYPE OF (surface_treatment);
    method : cleaning_method;
```

END_ENTITY;
(*

Attribute definitions:

method

Declares the classification of the method of cleaning as either a chemical wash, or a blast clean. The method of cleaning may be classed as undefined.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.13 surface_treatment_coat

Entity definition:

A type of surface_treatment that identifies, describes and specifies a coating applied to the surface of a component of a steel frame building as one of the manufacturing processes. For example, a steel beam may be coated using a number of layers of paint. The number of layers must be specified as well as the thickness of each layer. Each layer may serve a different purpose and have a different specification.

EXPRESS specification:

```
*)
ENTITY surface_treatment_coat
SUBTYPE OF (surface_treatment);
    methods : LIST [1:?] OF coating_method;
    layer_thicknesses : LIST [1:?] OF positive_length_measure_with_unit;
    coating_specifications : LIST [1:?] OF coating;
DERIVE
    number_of_layers : INTEGER := SIZEOF(methods);
WHERE
    WRS41 : SIZEOF(layer_thicknesses) = number_of_layers;
    WRS42 : SIZEOF(coating_specifications) = number_of_layers;
END_ENTITY;
(*
```

Attribute definitions:

methods

Declares the class of the method used to apply each coat that make up the coating. The coating method may be classed as either sprayed, brushed, dipped, electroplated, or undefined. A method is required for each layer applied, such that a coating of three layers will require three method; e.g. .BRUSHED., .BRUSHED., .SPRAYED. The maximum size of the list of enumeration items shall be equal to the integer value of the number of layers that make up the coating process.

layer_thicknesses

Declares the list of one or more instances of length_measure_with_unit associated with the surface_treatment_coat, which provide the numerical value with an appropriate unit of the thickness of each layer making up the coating. There must be at least one instance of

length_measure_with_unit referenced by each instance of surface_treatment_coat. The maximum size of the list of instances of length_measure_with_unit shall be equal to the integer value of the number of layers that make up the coating process. This would normally be the 'dry film thickness' of each layer of paint. The thickness is measured in accordance with the appropriate standard or code of practice.

coating_specifications

Declares the list of one or more instances of coating associated with the surface_treatment_coat, which provide the identification, description and specification of each layer making up the coating. There must be at least one instance of coating referenced by each instance of surface_treatment_coat. The maximum size of the list of instances of coating shall be equal to the integer value of the number of layers that make up the coating process.

number_of_layers

This attribute derives the integer value of the number of layers that make up the coating process, from the size of the aggregation associated with the attribute methods.

Formal propositions:

DERIVE

The derived attribute number_of_layers will not appear in the STEP Part 21 file.

WRS41

The size of the aggregation associated with the attribute layer_thicknesses shall be equal to the integer value derived by the attribute number_of_layers. In other words, there must be as many instances of positive_length_measure_with_unit as there are coating_methods associated with this surface_treatment_coat.

WRS42

The size of the aggregation associated with the attribute coating_specifications shall be equal to the integer value derived by the attribute number_of_layers. In other words, there must be as many instances of coating as there are coating_methods associated with this surface_treatment_coat.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.14 surface_treatment_grind

Entity definition:

A type of surface_treatment that identifies, describes and specifies a grinding process applied to the surface of a component of a steel frame building as one of the manufacturing processes.

EXPRESS specification:

*)

ENTITY surface_treatment_grind

SUBTYPE OF (surface_treatment);

finished_surface_irregularity : OPTIONAL length_measure_with_unit;

END_ENTITY;

(*)

Attribute definitions:*finished_surface_irregularity*

Declares the instance of `length_measure_with_unit` that may be associated with the `surface_treatment_grind` to provide the numerical value with an appropriate unit of the final permitted irregularity of the surface finish following the grinding process. This parameter is measured in accordance with (or specified in) the appropriate standard or code of practice.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.15 surface_treatment_hard_stamp**Entity definition:**

A type of `surface_treatment` that identifies, describes and specifies a physical ‘hard stamp’ (or signature) applied to the surface of a component of a steel frame building as one of the manufacturing processes. For example, a steel beam may have its ‘piece number’ indented in its surface, rather than painted on.

EXPRESS specification:

*)

ENTITY `surface_treatment_hard_stamp`

SUBTYPE OF (`surface_treatment`);

`stamp_method` : `cutting_type`;

END_ENTITY;

(*)

Attribute definitions:*stamp_method*

Declares the classification of the method used to make the ‘hard stamp’ as being either sawn, flame cut, sheared, punched, drilled, laser cut, or cut by abrasion. The method used to make the ‘hard stamp’ may be classed as undefined.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.16 surface_treatment_thermal**Entity definition:**

A type of `surface_treatment` that identifies, describes and specifies a heating or cooling process applied to the surface of a component of a steel frame building as one of the manufacturing processes. If the rates of heating and cooling are required, then these are specified using the SUBTYPE `surface_treatment_thermal_timed`. An instance describes one heating and cooling cycle; for repeated cycles of heating and cooling with different temperatures, additional instances are required.

EXPRESS specification:

*)

ENTITY surface_treatment_thermal

SUPERTYPE OF (surface_treatment_thermal_timed)

SUBTYPE OF (surface_treatment);

initial_temperature : thermodynamic_temperature_measure_with_unit;

final_temperature : thermodynamic_temperature_measure_with_unit;

maximum_temperature : thermodynamic_temperature_measure_with_unit;

END_ENTITY;

(*

Attribute definitions:*initial_temperature*

Declares the first instance of thermodynamic_temperature_measure_with_unit associated with the surface_treatment_thermal, which provides the numerical value with an appropriate unit of the temperature of the surface of the component before the process starts.

final_temperature

Declares the second instance of thermodynamic_temperature_measure_with_unit associated with the surface_treatment_thermal, which provides the numerical value with an appropriate unit of the temperature of the surface of the component when the process has finished. If the component is allowed to cool naturally, the final temperature may be the same as the maximum temperature. If the cooling is controlled, the final temperature may be different from the maximum temperature and may equal the initial temperature.

maximum_temperature

Declares the third instance of thermodynamic_temperature_measure_with_unit associated with the surface_treatment_thermal, which provides the numerical value with an appropriate unit of the maximum temperature of the surface of the component when the process has finished. If the component is allowed to cool naturally, the final temperature may be the same as the maximum temperature. If the cooling is controlled, the final temperature may be different from the maximum temperature and may equal the initial temperature.

Informal propositions:

If the instance of surface_treatment_thermal is describing a controlled cooling (or quenching) process, the maximum temperature will equal the initial temperature.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.17 surface_treatment_thermal_timed**Entity definition:**

A type of surface_treatment_thermal that specifies the amounts of time taken to heat and cool the surface of a component subjected to heat treatment. Typical examples are shown in Figure 18.1 and Figure 18.2.

EXPRESS specification:

```

*)
ENTITY surface_treatment_thermal_timed
SUBTYPE OF (surface_treatment_thermal);
    time_to_maximum : time_measure_with_unit;
    time_at_maximum : time_measure_with_unit;
    time_to_final : time_measure_with_unit;
END_ENTITY;
(*

```

Attribute definitions:*time_to_maximum*

Declares the first instance of `time_measure_with_unit` associated with the `surface_treatment_thermal_timed`, which provides the numerical value with an appropriate unit of the period of time taken to raise the temperature of the surface of the component from its initial value to its maximum specified value. This provides a measure of the rate of heating of the process.

time_at_maximum

Declares the second instance of `time_measure_with_unit` associated with the `surface_treatment_thermal_timed`, which provides the numerical value with an appropriate unit of the period of time the temperature of the surface of the component is maintained at its maximum specified value.

time_to_final

Declares the third instance of `time_measure_with_unit` associated with the `surface_treatment_thermal_timed`, which provides the numerical value with an appropriate unit of the period of time taken to lower the temperature of the surface of the component from its maximum specified value to its final value. This provides a measure of the rate of cooling of the process.

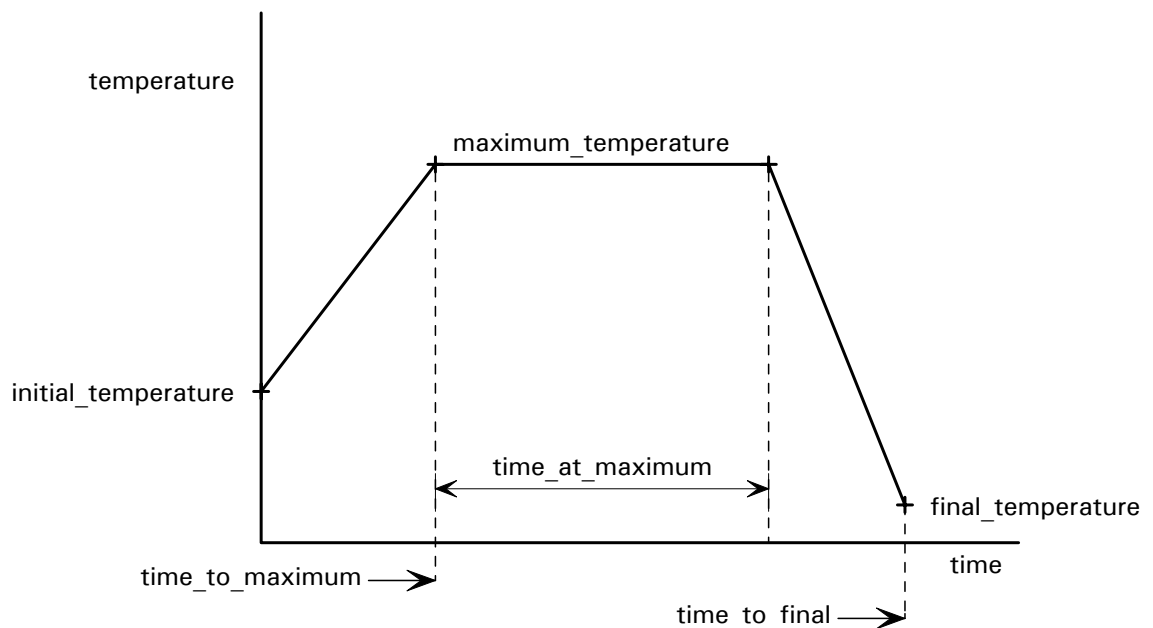


Figure 18.1 *An example of a heating and cooling process*

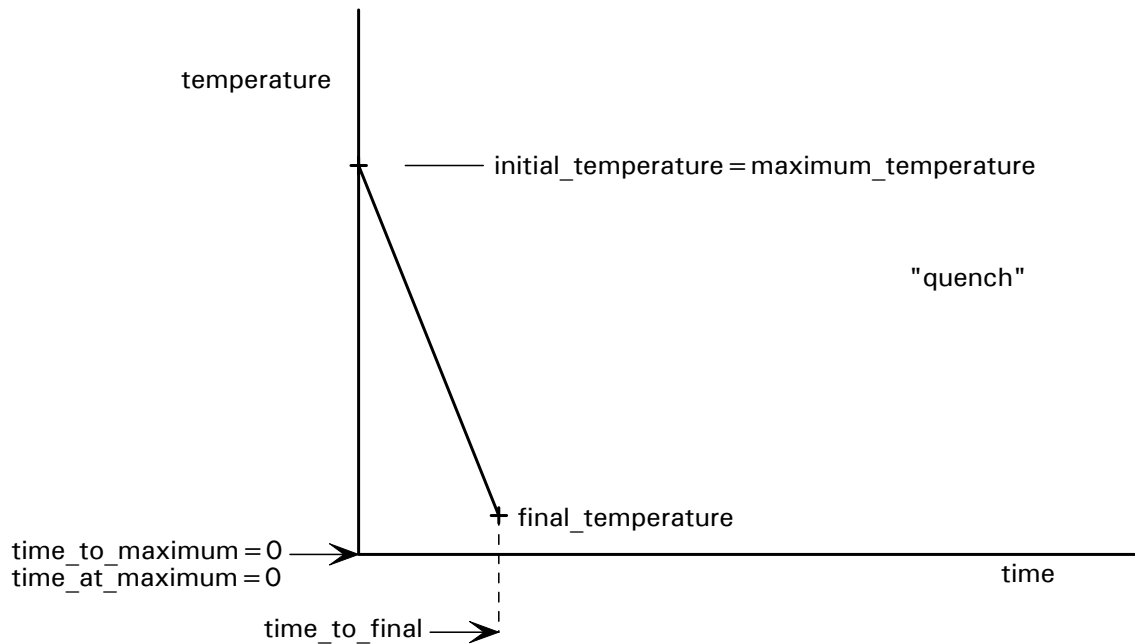


Figure 18.2 *An example of a quenching process*

Informal propositions:

If the instance of `surface_treatment_thermal_timed` is describing a controlled cooling (or quenching) process, the maximum temperature will equal the initial temperature and the time to maximum and the time at maximum will equal zero.

Notes:

New for CIS/2.

See diagram 57 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

18.3.18 weld

Entity definition:

A type of `structural_frame_process` which identifies, describes and specifies a welding process. The weld itself is specified through the associated `joint_system_welded`. Welding is defined in BS 499^[2] as “an operation in which two or more parts are united, by means of heat or pressure, or both, in such a way that there is continuity in the nature of the metal between these parts. A filler metal, the melting temperature of which is of the same order as that of the parent metal, may or may not be used.”

EXPRESS specification:

*)

ENTITY weld

SUPERTYPE OF (ONEOF(weld_arc,

weld_beam,

weld_gas,

weld_other,

weld_pressure,

weld_resistance,

weld_stud))

```

SUBTYPE OF (structural_frame_process);
    electrode_type : OPTIONAL label;
    weld_type : welding_type;
END_ENTITY;
(*)

```

Attribute definitions:

electrode_type

An optional short text description of the type of electrode used during the welding process.

weld_type

Declares the classification of the weld by the method of its formation. The weld may be classed as being formed by fusion, friction, flash, laser, or forge. The method of the weld's formation may be classed as undefined.

Informal propositions:

Because weld is a SUBTYPE structural_frame_process (which is a SUBTYPE of structural_frame_item) it may be assigned a standardized item_reference (via the entity item_reference_assigned).

Notes:

New for CIS/2.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition - Subtypes added

18.3.19 weld_arc

Entity definition:

A type of weld which identifies, describes and specifies an arc welding process. The weld itself is specified through the associated joint_system_welded. Arc welding is defined in BS 499^[2] as “*fusion welding in which heat for welding is obtained from an electric arc or arcs*”.

EXPRESS specification:

```

*)
ENTITY weld_arc
SUBTYPE OF (weld);
    weld_arc_type : welding_type_arc;
WHERE
    WRW13 : SELF\weld.weld_type = FUSION_WELD;
    WRW14 : EXISTS(SELF\weld.electrode_type);
END_ENTITY;
(*)

```

Attribute definitions:

weld_arc_type

Declares the classification of the arc weld by the method of its formation. The arc weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_arc.

Formal propositions:

WRW13

The arc weld shall be classed as a fusion weld via the attribute weld_type inherited from the supertype weld.

WRW14

The attribute electrode_type type inherited from the supertype weld must be given a value; i.e. the subtype removes the OPTIONAL clause from the supertype.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.20 weld_beam

Entity definition:

A type of weld which identifies, describes and specifies a beam welding process. The weld itself is specified through the associated joint_system_welded.

EXPRESS specification:

```

*)
ENTITY weld_beam
SUBTYPE OF (weld);
    weld_beam_type : welding_type_beam;
WHERE
    WRW19 : (SELF.weld.weld_type = FUSION_WELD)
            OR (SELF.weld.weld_type = LASER_WELD);
    WRW20 : NOT(EXISTS(SELF.weld.electrode_type));
END_ENTITY;
(*

```

Attribute definitions:

weld_beam_type

Declares the classification of the beam weld by the method of its formation. The arc weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_beam.

Formal propositions:

WRW19

The beam weld shall be classed as a fusion weld or a laser weld via the attribute weld_type inherited from the supertype weld.

WRW20

The attribute electrode_type type inherited from the supertype weld must not be given a value; i.e. the subtype prevents the use of an inappropriate attribute from the supertype.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.21 weld_gas

Entity definition:

A type of weld which identifies, describes and specifies a gas welding process. The weld itself is specified through the associated joint_system_welded. Gas welding is a welding process which produces coalescence by heating materials with an oxy fuel gas flame or flames with or without the application of pressure and with or without the use of filler metal.

EXPRESS specification:

```
*)
ENTITY weld_gas
SUBTYPE OF (weld);
    weld_gas_type : welding_type_gas;
WHERE
    WRW15 : SELF\weld.weld_type = FUSION_WELD;
    WRW16 : NOT(EXISTS(SELF\weld.electrode_type));
END_ENTITY;
(*)
```

Attribute definitions:

weld_gas_type

Declares the classification of the gas weld by the method of its formation. The gas weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_gas.

Formal propositions:

WRW15

The gas weld shall be classed as a fusion weld via the attribute weld_type inherited from the supertype weld.

WRW16

The attribute electrode_type type inherited from the supertype weld must not be given a value; i.e. the subtype prevents the use of an inappropriate attribute from the supertype.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.22 weld_other

Entity definition:

A type of weld which identifies, describes and specifies a welding process that cannot be classified under the other categories as represented by the other subtypes of weld. The weld itself is specified through the associated joint_system_welded.

EXPRESS specification:

```
*)
ENTITY weld_other
SUBTYPE OF (weld);
    weld_other_type : welding_type_other;
```


END_ENTITY;
(*

Attribute definitions:

weld_other_type

Declares the classification of the weld by the method of its formation. The weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_other.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.23 weld_pressure

Entity definition:

A type of weld which identifies, describes and specifies a pressure welding process. The weld itself is specified through the associated joint_system_welded.

EXPRESS specification:

```
*)
ENTITY weld_pressure
SUBTYPE OF (weld);
    weld_pressure_type : welding_type_pressure;
WHERE
    WRW17 : NOT((SELF\weld.weld_type = LASER_WELD)
                OR (SELF\weld.weld_type = FLASH_WELD));
    WRW18 : NOT(EXISTS(SELF\weld.electrode_type));
END_ENTITY;
(*
```

Attribute definitions:

weld_pressure_type

Declares the classification of the pressure weld by the method of its formation. The pressure weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_gas.

Formal propositions:

WRW17

The pressure weld shall not be classed as a laser weld or a flash weld via the attribute weld_type inherited from the supertype weld.

WRW18

The attribute electrode_type type inherited from the supertype weld must not be given a value; i.e. the subtype prevents the use of an inappropriate attribute from the supertype.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.24 weld_resistance

Entity definition:

A type of weld which identifies, describes and specifies a resistance welding process. The weld itself is specified through the associated joint_system_welded. Resistance welding is a welding process which produces coalescence of metals with the heat obtained from resistance of the work to electric current in a circuit of which the work is a part, and by the application of pressure. Pressure welding is defined in BS 499^[2] as “*welding in which, at some stage in the process, force is applied to surfaces in contact and in which the heat for welding is produced by the passage of electric current through the electrical resistance at, and adjacent to, these surfaces*”.

EXPRESS specification:

```
*)
ENTITY weld_resistance
SUBTYPE OF (weld);
    weld_resistance_type : welding_type_resistance;
END_ENTITY;
(*
```

Attribute definitions:

weld_resistance_type

Declares the classification of the resistance weld by the method of its formation. The resistance weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_resistance.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

18.3.25 weld_stud

Entity definition:

A type of weld which identifies, describes and specifies a stud welding process. The weld itself is specified through the associated joint_system_welded. Stud welding is defined in BS 499^[2] as “*the joining of a metal stud or similar part to a workpiece. Welding may be accomplished by arc, resistance, friction or other suitable process, with or without external gas shielding. The weld is made over the whole end area of the stud or attachment*”.

EXPRESS specification:

```
*)
ENTITY weld_stud
SUBTYPE OF (weld);
    weld_stud_type : welding_type_stud;
END_ENTITY;
(*
```

Attribute definitions:

weld_stud_type

Declares the classification of the stud weld by the method of its formation. The stud weld may be classed as being formed in a number of different ways as described in the definition of the TYPE welding_type_stud.

Notes:

New for CIS/2 2nd Edition.

See diagram 56 of the EXPRESS-G diagrams in Appendix B.

19 LPM/6 DATA MANAGEMENT

19.1 Data Management concepts and assumptions

The use of the data management constructs has significant implications for implementation. Software developers are recommended to read the 'Conformance Requirements' and the 'Implementation Guide' before writing their translator coding. It is also recommended that vendors develop and test 'Basic Translators' before attempting the more advanced DMC systems.

19.1.1 Data Management and CIS/2

Providing an adequate and efficient mechanism for data management within the LPM was seen as the key to the long-term success of the CIS. CIS/2 provides data management functionality in LPM/6 by separating the data from the meta-data, and then allowing those applications that can associate the meta-data with the data (i.e. DMC applications) to do so. A further constraint was that DMC and non-DMC applications have to be able to share data with each other. This is because different software applications will develop at different rates; some will include data management capabilities before others, but users of DMC applications will still wish to share data with users of non-DMC applications.

The meta-data includes some or all of the following information:

- The unique identifier for the item.
- The person who created the data.
- The date when it was created.
- Whether the data was new or old data that had been modified.
- The date when it had been modified.
- Whether the data has been approved or not.

For example, a DMC-application would be used to assign a unique identifier to an instance of 'assembly' and place it in a set that states who created the instance, when it was created, and whether it had been modified.

It should be noted that the engineering data can exist without the meta-data, and is not dependent upon it. On the other hand, the meta-data is dependent upon the data for which it has been created.

19.1.2 The purpose of data management

The main purpose of data management is to allow engineering data created by one software application to be updated and passed onto another application or fed back to the original application. This information flow and feedback must be made possible regardless of the application's scope, functionality, or position in the production process. To manage the data, an application must be able to 'recognize' incoming data instances as modified versions of existing data instances.

19.1.3 'DMC Translators' & 'DMC Applications'

A translator written for a CIS/2-conformant system that has data management capabilities is known as a DMC Translator. The extent of the data management functionality will depend upon the capabilities of the underlying application for which the translator is

written; i.e. whether the native application can manage data. It will also depend upon whether the system has both an import translator and an export translator.

Where a native application can manage data *and* is provided with *both* an import translator and an export translator, it shall be referred to as a DMC application. This is discussed in more detail in Section 6 of the *Implementation Guide*.

19.1.4 Meta-data & data management

In CIS/2, data management is considered to be the process of assigning meta-data to data. In simple terms, ‘Data Management Conformant’ (DMC) applications produce ‘managed data’ such that each item of data is given a unique identification and is placed in a data set, whereas Non-DMC applications will produce ‘unmanaged data’. Although several applications will contribute to and modify data instances, the unique identifier will be assigned by the application that originally created the meta-data to manage that data. Furthermore, ‘unmanaged data’ may be mapped onto meta data to become ‘managed data’, using a DMC application.

By definition, a DMC translator is required to be capable of supporting all of the data management constructs described in this section of LPM/6. Of most concern to the writers of DMC-Translators are the two entities `managed_data_item` and `managed_data_transaction`, which form the essential EXPRESS constructs in the LPM for data management by capturing the meta-data.

A `managed_data_item` represents an individual item of managed data and allows that data item to be assigned a unique identifier, and a history. A `managed_data_transaction` is simply a record of the use of a DMC application (e.g. an export or import process). A `managed_data_transaction` collects together managed data, and assigns the additional meta-data that is associated with each and every item of data in the collection.

The number of data entity constructs (used to represent engineering information) that a DMC translator supports will be dependent upon the scope of the application for which the DMC translator is written. The creation and processing of these data entity constructs is handled in the same way as those for a Basic CIS Translator.

19.1.5 Inter-working between DMC and Non-DMC applications

When a DMC Translator reads ‘unmanaged data’, the translator must be able to convert the data into ‘managed data’ by assigning meta-data. Since the history of the data is unknown, the translator assigns that meta-data as if the DMC application had created the data itself. In this case, the unique identifier will be determined by the application’s DMC Translator, and thus will reflect the name of DMC application that managed the data rather than the application that actually created the data. Any reference to the original creator will have been lost.

When a Non-DMC Translator reads ‘managed data’, the translator simply ignores the meta-data, importing only those entities that lie within the scope of the underlying application. The data is effectively ‘dumbed down’ and becomes unmanaged; i.e. only the ‘raw’ engineering data remains.

19.2 Data Management type definitions

No types have been modified in this subject area for the 2nd Edition.

The following types have been added to this subject area for the 2nd Edition:

- `globally_unique_id`

19.2.1 data_status_type

Type definition:

Classifies the instance of managed_data_deleted as either deleted, superseded, archived, erroneous or undefined. This type may be used to describe the reason why the engineering data associated with the instance of managed_data_item has been deleted.

The classification “deleted” states that the engineering data formerly associated with the instance of managed_data_item has simply been deleted by the underlying engineering application.

The classification “superseded” states that the engineering data has been superseded by data represented by another instance of managed_data_item. The entities managed_data_group and group_relationship may be used to represent the succession of one set of engineering data by another. The entity group_usage (a SUBTYPE of group_relationship) may be used to uniquely identify the relationship between the two groups of managed data, and ensure that the relationship is not self defining; i.e. that one data set is not dependent upon itself either directly or indirectly.

The classification “archived” states that the engineering data has been archived. The associated engineering data is no longer considered to be of immediate interest even though it may still be valid. The data may have been archived by the underlying application or, more likely, by a central project database.

The classification “erroneous” states that the engineering data has been deleted because it was erroneous. That is, the engineering data formerly associated with the instance of managed_data_item has been deleted by a DMC application because it contained errors. The data has been scrapped, rather than superseded or archived. However, the original meta-data and the instance of managed_data_item live on.

Finally, the classification “unknown” is used when the reason for the deletion of the engineering data is unknown. (This classification should only be used in exceptional circumstances.

EXPRESS specification:

*)

TYPE data_status_type

= ENUMERATION OF

(deleted, superseded, archived, erroneous, undefined);

END_TYPE;

(*

Notes

New for CIS/2. Unchanged for 2nd Edition.

19.2.2 globally_unique_id

Type definition:

An alphanumeric identifier that is unique throughout the software world. This identifier is a string encoding of a globally unique unsigned 128bit integer and is also known as a Globally Unique Identifier (GUID) or a Universal Unique Identifier (UUID). Algorithms for generating the 128bit integer and for encoding this integer in a string representation are defined in documentation published by ISO^[33] and by the Open Group^[37]. The string representation consists of a sequence of hexadecimal fields separated by single dashes

totalling 36 characters. An example of the `globally_unique_id` is the string “1A81FD96-D7A8-47d3-8DF7-BEEA6FF45184”.

The Microsoft Foundation Class (MFC) function "CoCreateGuid", which is an implementation of the referenced algorithms, may be used by CIS/2 implementers on Windows platforms to create this identifier (as a GUID). (See the Microsoft Online Documentation^[35].) Other implementations are available as UUID generators written in Java and other languages for use on MacOS and the various Unix platforms.

EXPRESS specification:

```
*)
TYPE globally_unique_id
= STRING;
WHERE
    WRTG1 : LENGTH(SELF) > 0;
END_TYPE;
(*
```

Notes

New for CIS/2 2nd Edition.

19.2.3 select_data_item

Type definition:

Allows the selection of one of the data entities in the LPM. This is achieved through the use of several SELECT constructs which ultimately refer to a particular entity. The types of entities in the SELECT constructs are all the highest level supertypes or ‘stand alone’ (‘orphan’) entities. Thus, all data entities (as opposed to meta-data entities) may be selected. This type is used when data management conformant applications need to populate the entity managed_data_item to SELECT which data item is be given meta data.

This type also allows the selection of an instance of the entity managed_data_deleted, which allows the data associated with the meta-data to be classed as ‘deleted’.

EXPRESS specification:

```
*)
TYPE select_data_item
= SELECT
    (managed_data_deleted,
    select_generic_item,
    select_analysis_item,
    select_design_item,
    select_physical_item,
    select_project_definition_item,
    select_structural_item);
END_TYPE;
(*
```

Notes:

New for CIS/2. Unchanged for 2nd Edition.

19.2.4 select_data_source

Type definition:

Allows the selection of the source of the data (imported or exported) to be represented by an instance of either managed_application_installation or step_file. This indicates that the data has come from (or is going to) a STEP Part 21 file or another DMC application.

EXPRESS specification:

```
*)
TYPE select_data_source
= SELECT
    (managed_application_installation, step_file);
END_TYPE;
(*
```

Notes

New for CIS/2. Unchanged for 2nd Edition.

19.3 Data Management entity definitions

The following entities have been modified in this subject area for the 2nd Edition:

- managed_data_item

The following entities have been added to this subject area for the 2nd Edition:

- managed_data_item_with_history

19.3.1 managed_application_installation

Entity definition:

A representation of an installation of a DMC application. This provides one aspect of the meta-data associated with the engineering data. It is a requirement of CIS/2 that the particular instance of software responsible for managing the data has a unique identifier, which is used in the management of the history of the data being managed.

Clearly there will be many copies of any given issue of a software application, but the identifier must be unique to a particular licenced copy of that application. This implies that DMC translators can only be run with a legal copy of the underlying application software.

A single user copy of an application, or a network installation of the application having only one licence, has a single registered installation identifier, and may be represented by one instance of this entity.

A network installation of the application that can have a number of simultaneous users requires multiple licences and a corresponding number of unique identifiers. The DMC application must be able to manage the users such that, at any time, each user is assigned a different identifier from the list of identifiers that have been licenced. In such a scenario, the installations should be represented by multiple instances of this entity; one for each licence.

The managed_application_installation contains two components of the unique identifier for an instance of engineering data; one for the application and one for the particular installation.

EXPRESS specification:

*)

```

ENTITY managed_application_installation;
    application_id : INTEGER;
    application_name : label;
    application_version : label;
    application_description : OPTIONAL text;
    application_vendor : OPTIONAL organization;
    installation_id : INTEGER;
    installation_name : label;
    installation_owner : OPTIONAL organization;
UNIQUE
    URM1 : application_id, installation_id;
WHERE
    WRM1 : installation_id > 0;
    WRM2 : application_id > 0;
END_ENTITY;
(*)

```

Attribute definitions:***application_id***

The integer value that uniquely identifies the application within the context of the CIS/2 data exchange. The value should be unique to an application and its vendor, as it is intended to identify the software vendor, the software application and the version of the application.

application_name

A short human-readable alphanumeric reference for the application; e.g. “QSteel”. The instance of managed_application_installation represents one installation of the application named here.

application_version

A short human-readable alphanumeric reference for the version of the application; e.g. “2.3a”. Many software applications go through several versions in their commercial life; each must have a different application_version and a unique application_id.

application_description

An optional free text description of the application. This may include a description of the scope and functionality of the application. The instance of managed_application_installation represents one installation of the application described here.

application_vendor

Declares the second instance of organization that may be associated with the managed_application_installation. This organization is deemed to have sold the installation of the application. There may be one, and only one, organization that acts as the vendor of the managed_application_installation. That is, only one organization can act as the registered retailer of the installation of the application. This organization is responsible for registering and maintaining the application_id and the associated set of

installation_ids. This organization may, but need not, be the original authors of the application; it could be a third party.

installation_id

The integer value that uniquely identifies the installation within the context of the application. The value should be unique to one installation of a particular application, as it is intended to identify the ‘registered keeper’ of the software. To ensure its uniqueness within the domain of the particular software application, the software vendor is required to assign and maintain the installation id for each installation of each version of each DMC-application.

installation_name

A short alphanumeric reference for the particular installation of the application. This could, but need not be the ‘licence key’ of the installation.

installation_owner

Declares the instance of organization associated with the managed_application_installation. This organization is deemed to own the installation of the application. There may be one, and only one, organization that acts as the owner of the managed_application_installation. That is, only one organization can act as the registered keeper of the installation of the application.

Formal propositions:

URM1

The combination of the values assigned to the attributes application_id and installation_id shall be unique to the managed_application_installation. That is, taken together, the application_id and installation_id uniquely identify the instance of managed_application_installation. This allows the installation to be uniquely identified within the context of the application.

WRM1

The value assigned to the attribute installation_id shall be greater than zero. That is, the installation_id must be a positive integer.

WRM2

The value assigned to the attribute application_id shall be greater than zero. That is, the application_id must be a positive integer.

Informal propositions

Each instance of the entity managed_data_item is assigned a three part unique identifier derived from the combination of its instance_id and the attributes installation_id and application_id from this entity.

Notes

New for CIS/2.

To ensure its uniqueness within the domain of CIS/2, the application_id should be registered with the **CIS/2 International Technical Committee (ITC)** before it is used. Before developing a DMC translator, a software vendor needs to apply to the ITC to obtain their registered ‘application id’.

See Diagram 1 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition – although it now has a smaller role in data management. That is, where all that is required is the passing of a GUID then this entity will not be instanced. However, where the data management requirement involves capturing and maintaining the history of the data collection then this entity becomes mandatory and its instances will place the GUID into the wider CIS identification context.

19.3.2 managed_data_creation

Entity definition:

A type of managed_data_transaction that records the use of an installation of a DMC application that created managed data. It is asserted that new unique identifiers were created during this event. The managed data may actually be old data that has been assigned new meta-data. That is, it is the data management that is new rather than the data itself. This distinction becomes important when DMC applications import unmanaged data.

The purpose of this entity is to constrain the use of its SUPERTYPE.

EXPRESS specification:

```
*)
ENTITY managed_data_creation
SUBTYPE OF (managed_data_transaction);
DERIVE
    created_set : SET [1:?] OF managed_data_item := bag_to_set (USEDIN (SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));
WHERE
    WRM3 : SELF\managed_data_transaction.new_ids_assigned = TRUE;
    WRM4 : SIZEOF(QUERY(tmp <* created_set | tmp.originating_application :<>:
        (SELF\managed_data_transaction.application)) ) = 0;
END_ENTITY;
(*
```

Attribute definitions:

created_set

This attribute derives the set of instances of managed_data_item that reference this instance of managed_data_creation (via the attribute history). This collates the managed data created during the event recorded by this entity.

Formal propositions:

DERIVE

The derived attribute created_set will not appear when the instance is encoded in a STEP Part 21 file.

WRM3

The attribute new_ids_assigned (inherited from the SUPERTYPE managed_data_transaction shall be assigned the value TRUE. That is, new instance_ids have been assigned, which means that at least one new instance of managed_data_item was created. (Rules in the entity managed_data_item require that all the instances of managed_data_item making use of this entity are new.)

WRM4

The size of the set of instances of `managed_data_item` (derived by the attribute `created_set`) that have an originating application different from the instance of `managed_application_installation` referenced by the attribute `application` (inherited from the SUPERTYPE `managed_data_transaction`) shall be zero. In other words, the installation of the DMC application that created the managed data must be the same installation whose use is recorded here.

Notes

New for CIS/2.

See Diagram 1 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

19.3.3 managed_data_deleted**Entity definition:**

A derived collection of instances of `managed_data_item` for which the associated engineering data has been deleted. The collection of data must be given a status, which may declare the classification of the data as either deleted, superseded, archived, erroneous, or undefined.

This entity allows the item of data associated with the `managed_data_item` to be declared as deleted. This allows the meta-data associated with the data to be maintained after the item of engineering data has been deleted by an application. In such as event, only the meta-data is exchanged rather than the deleted data itself. This minimizes the amount of redundant data that will be exchanged by DMC applications.

EXPRESS specification:

*)

```
ENTITY managed_data_deleted;
  data_status : data_status_type;
DERIVE
```

```
  deleted_data_items : SET [1:?] OF managed_data_item := bag_to_set (USEDIN (SELF,
    'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.DATA_ITEM'));
```

```
END_ENTITY;
```

(*

Attribute definitions:*data_status*

Declares the classification of the data as either deleted, superseded, archived, or erroneous. The status of the data may be classed as undefined. (See definition of the type `data_status`, for the meaning of the classification.)

deleted_data_items

This attribute derives the set of instances of `managed_data_item` that reference this instance of `managed_data_deleted` (via the attribute `data_item`). This collates the data that has been deleted.

Formal propositions:**DERIVE**

The derived attribute `deleted_data_items` will not appear when the instance is encoded in a STEP Part 21 file.

Notes

New for CIS/2.

See Diagram 6 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

19.3.4 managed_data_export**Entity definition:**

A type of `managed_data_transaction` that records the use of an installation of DMC that exported managed data. The managed data may be exported to a STEP Part 21 file or another DMC application.

The purpose of this entity is to constrain the use of its SUPERTYPE.

EXPRESS specification:

*)

ENTITY `managed_data_export`

SUBTYPE OF (`managed_data_transaction`);

`data_destination` : `select_data_source`;

DERIVE

`exported_set` : SET [1:?] OF `managed_data_item` := `bag_to_set` (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));

`assignments` : SET [0:?] OF `managed_data_group` := `bag_to_set` (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_GROUP.ITEMS'));

END_ENTITY;

(*

Attribute definitions:***data_destination***

Declares the instance of `managed_application_installation` or `step_file` associated with the `managed_data_export` to define where the managed data was exported to. The choice is made via the SELECT type `select_data_source`. The managed data may be recorded as being exported to another DMC application or a STEP Part 21 file.

exported_set

This attribute derives the set of instances of `managed_data_item` that reference this instance of `managed_data_export` (via the attribute history). This collates the managed data exported during the event recorded by this entity.

assignments

This attribute derives the set of instances of `managed_data_group` that reference this instance of `managed_data_export` (via the attribute items). This collates the relatives of the `managed_data_export`; i.e. those other instances of `managed_data_transaction` that have been placed in the same group of managed data.

The entity `managed_data_group` is a SUBTYPE of `group_assignment`, and as such, it can be assigned to (and deemed to be the contents of) a `step_file`. It can also be assigned an approval and/or an action.

Formal propositions:

DERIVE

The derived attributes `exported_set` and assignments will not appear when the instance is encoded in a STEP Part 21 file.

Notes

New for CIS/2.

See Diagram 1 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

19.3.5 `managed_data_group`

Entity definition:

A type of `group_assignment` that collects together a set of managed data. Due to its ANDOR SUPERTYPE, it can be assigned to (and deemed to be the content of) a `step_file`. It can also be assigned an approval and/or an action.

EXPRESS specification:

```
*)
ENTITY managed_data_group
  SUBTYPE OF (group_assignment);
    items : SET [1:?] OF managed_data_transaction;
END_ENTITY;
(*
```

Attribute definitions:

items

Declares the set of instances of `managed_data_transaction` that are associated with the `managed_data_group`. There must be at least one instance of `managed_data_transaction` referenced by each instance of `managed_data_group`. The SET data type prevents repetition of instances. Each instance of `managed_data_transaction` is referenced by a number of instances of `managed_data_item`. In this way, the managed data is grouped together. As a group, they can then be associated with a common action or approval.

Notes

New for CIS/2.

See Diagram 2 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

19.3.6 `managed_data_import`

Entity definition:

A type of `managed_data_transaction` that records the use of an installation of DMC that imported managed data. The managed data may be imported from a STEP Part 21 file or another DMC application.

The purpose of this entity is to constrain the use of its SUPERTYPE.

EXPRESS specification:

```

*)
ENTITY managed_data_import
SUBTYPE OF (managed_data_transaction);
    data_source : select_data_source;
DERIVE
    imported_set : SET [1:?] OF managed_data_item := bag_to_set (USEDIN (SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));
    assignments : SET [0:?] OF managed_data_group := bag_to_set (USEDIN (SELF,
        STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_GROUP.ITEMS'));
END_ENTITY;
(*

```

Attribute definitions:*data_source*

Declares the instance of managed_application_installation or step_file associated with the managed_data_import to define where the managed data was imported from. The choice is made via the SELECT type select_data_source. The managed data may be recorded as being imported from another DMC application or a STEP Part 21 file.

imported_set

This attribute derives the set of instances of managed_data_item that reference this instance of managed_data_import (via the attribute history). This collates the managed data imported during the event recorded by this entity.

assignments

This attribute derives the set of instances of managed_data_group that reference this instance of managed_data_import (via the attribute items). This collates the relatives of the managed_data_import; i.e. those other instances of managed_data_transaction that have been placed in the same group of managed data.

The entity managed_data_group is a SUBTYPE of group_assignment, and as such, it can be assigned to (and deemed to be the contents of) a step_file. It can also be assigned an approval and/or an action.

Formal propositions:**DERIVE**

The derived attributes imported_set and assignments will not appear when the instance is encoded in a STEP Part 21 file.

Notes

New for CIS/2.

See Diagram 1 of the EXPRESS-G diagrams in Appendix B.

Unchanged in 2nd Edition.

19.3.7 managed_data_item**Entity definition:**

An individual item of managed data. This entity provides the key to data management in CIS/2. It is the principal mechanism whereby data items are assigned a unique identifier and can optionally be placed in a data set.

EXPRESS specification:

```

*)
ENTITY managed_data_item;
    instance_id : globally_unique_id;
    originating_application : OPTIONAL managed_application_installation;
    data_item : select_data_item;
    history : LIST [0:?] OF UNIQUE managed_data_transaction;
    original_data : LOGICAL;
UNIQUE
    URM5 : instance_id, data_item;
WHERE
    WRM36 : unique_data_item(data_item);
END_ENTITY;
(*

```

Attribute definitions:***instance_id***

The identifier that uniquely identifies the managed_data_item in the context of the originating application. This value should be unique to an instance created by a particular implementation of an application, as it is intended to identify the instance of data. To ensure its uniqueness within the domain of the particular installation of an application, the DMC application is required to assign and maintain these instance ids. (See the definition of the type globally_unique_id.)

Note – the underlying data type of this attribute has changed from an INTEGER to a STRING. For upward compatibility, an instance_id written in accordance with the first edition of CIS/2 needs to be converted from an INTEGER to a STRING.

originating_application

Declares the instance of managed_application_installation that may be associated with the managed_data_item, which is deemed to be the installation of the DMC application that created the managed data. An item of managed data can only be created by one application, but that application can create any amount of managed data.

data_item

Declares the instance of the data entity associated with the managed_data_item, to which meta data will be provided. Alternatively, the attribute may SELECT the entity managed_data_deleted, to indicate that the engineering data that was associated with this instance of managed_data_item has been deleted.

An entity instance is chosen by selecting from a hierarchical list of SELECT types and SUPERTYPE-SUBTYPE constructs. There must be one (and only one) **unique** instance of the data entity referenced by each instance of the managed_data_item. The unique identifier is assigned to (and therefore identifies) the instance of the data entity referred to here. An instance of a data entity may belong to one (and only one) instance of managed_data_item.

history

Declares the list of zero, one or more unique instances of managed_data_transaction that may be associated with the managed_data_item. Collectively, these instances provide the history of the managed data. This attribute has a LIST data type because the order of the list is important; in particular, DMC application must be able to recognize the first and

last events in the history of the managed data. The UNIQUE constraint prevents repetition of instances in the list; i.e. each event can only happen once.

original_data

Declares whether the managed_data_item has been created for original engineering data that the DMC application itself created, or for engineering data imported from elsewhere. Normally, the application underlying the DMC translator will have created the engineering data being managed, in which case, this attribute should be assigned the value TRUE. However, when a DMC application imports unmanaged data (from a non-DMC application), it may create meta-data for that data. In that case, this attribute should be assigned the value FALSE. In exceptional circumstances, this attribute may be assigned the value UNKNOWN.

Formal propositions:

URM5

The combination of the value assigned to the attribute instance_id and the instance of the data entity associated with the managed_data_item (through the attribute data_item) shall be unique to this instance of managed_data_item. That is, the meta-data provided by the managed_data_item (in particular, its unique identifier) shall be unique to the item of data associated here. (See also WHERE rule WRM36.)

WRM36

The instance of the data entity referenced by the attribute data_item shall be unique to this managed_data_item, or it shall be of the type managed_data_deleted. This rule uses the LPM/6 function unique_data_item, which calculates the number of times the data_item referenced is used by all the instances of managed_data_item in the information base. If the data_item is of the type managed_data_deleted, the function returns a value of UNKNOWN. If the data_item has been used only once, the function returns a value of TRUE. Otherwise, function returns a value of FALSE (and the instance is not valid).

Notes:

New for CIS/2.

Developers of DMC Translators should be aware of the complexities that are associated with SELECT constructs. In particular, they should study clause 11.1.8 of the *Technical Corrigendum to STEP Part 21*, which explains how SELECT constructs are mapped onto the exchange structure.

See diagram 1 of the EXPRESS-G diagrams in Appendix B.

Modified for 2nd Edition. This entity has been greatly simplified to allow instances of managed_data_item to be created without corresponding instances of managed_application_installation. This allows data to have a unique id without a history. The intention of this change is to simplify the implementation of DMC translators.

If DMC translators written in accordance with the first edition of CIS/2 wish to be upwardly compatible with the 2nd Edition, the old instance_id needs to be converted from an INTEGER to a STRING.

19.3.8 managed_data_item_with_history

Entity definition:

An individual item of managed data that has a history. This entity constrains the SUPERTYPE entity managed_data_item to having at least one item of history. It also places rules on the use of the managed_data_item and the contents of the history. The

first instance in the history list will represent the event that initiated the history tracking of the managed data. (This may, but need not, correspond to the event that created the original engineering data.) The entity managed_data_transaction is existence dependent on the SUPERTYPE entity managed_data_item. This entity effectively make the dependence mutual; i.e. an instance of managed_data_item_with_history cannot exist without a corresponding instance of the managed_data_transaction.

EXPRESS specification:

*)

ENTITY managed_data_item_with_history

SUBTYPE OF (managed_data_item);

DERIVE

number_of_uses : INTEGER := SIZEOF(SELF\managed_data_item.history);

first_managing_application : managed_application_installation :=

SELF\managed_data_item.history[1]

\managed_data_transaction.application;

first_managing_person : person :=

SELF\managed_data_item.history[1]

\managed_data_transaction.user.the_person;

date_first_managed : calendar_date :=

SELF\managed_data_item.history[1]

\managed_data_transaction.processing_date.date_component;

last_managing_application : managed_application_installation :=

SELF\managed_data_item.history[number_of_uses]

\managed_data_transaction.application;

last_managing_person : person :=

SELF\managed_data_item.history[number_of_uses]

\managed_data_transaction.user.the_person;

date_last_managed : calendar_date :=

SELF\managed_data_item.history[number_of_uses]

\managed_data_transaction.processing_date.date_component;

WHERE

WRM60 : number_of_uses > 0;

WRM61 : EXISTS(SELF\managed_data_item.originating_application);

WRM6 : first_managing_application :=: originating_application;

WRM7 : ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_CREATION'

IN TYPE OF(SELF\managed_data_item.history[1]));

WRM8 : NOT ((original_data = TRUE) AND

('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_IMPORT'

IN TYPE OF(SELF\managed_data_item.history[1])));

WRM37 : SIZEOF(QUERY(creation <* SELF\managed_data_item.history |

('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_CREATION') IN

TYPE OF(creation))) = 1;

WRM38 : NOT(('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_DELETED'

IN TYPE OF(SELF\managed_data_item.data_item))

AND (number_of_uses = 1));

WRM39 : NOT(('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_DELETED'

IN TYPE OF(SELF\managed_data_item.data_item))

```

        AND (SIZEOF(QUERY(modification <* SELF\managed_data_item.history |
        ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_MODIFICATION')
        IN TYPE OF(modification))) < 1) );
END_ENTITY;
(*

```

Attribute definitions:

number_of_uses

This attribute derives the integer value of the number of uses of the managed_data_item from the size of the list of instances of managed_data_transaction referenced by the attribute history. This value should correspond to the number of data processing events associated with the managed_data_item.

first_managing_application

This attribute derives the instance of managed_application_installation referenced by the first instance of managed_data_transaction referenced by the attribute history. This is deemed to be the installation of the DMC application that initiated the history tracking of the managed data. This may (but need not) be the installation of the DMC application that first managed the data.

first_managing_person

This attribute derives the instance of person associated with the instance of person_and_organization referenced by the first instance of managed_data_transaction referenced by the attribute history. This is deemed to be the person who initiated the history tracking of the managed data. This may (but need not) be the person who first managed the data.

date_first_managed

This attribute derives the instance of calendar_date associated with the instance of date_and_time referenced by the first instance of managed_data_transaction referenced by the attribute history. This is deemed to be the date on which history tracking was initiated. This may (but need not) be the date on which the data was first managed.

last_managing_application

This attribute derives the instance of managed_application_installation referenced by the last instance of managed_data_transaction referenced by the attribute history. This is deemed to be the last recorded installation of the DMC application that managed the data. This may (but need not) be the application that operated on the data most recently.

last_managing_person

This attribute derives the instance of person associated with the instance of person_and_organization referenced by the last instance of managed_data_transaction referenced by the attribute history. This is deemed to be the last recorded person who managed the data. This may (but need not) be the person who operated on the data most recently.

date_last_managed

This attribute derives the instance of calendar_date associated with the instance of date_and_time referenced by the last instance of managed_data_transaction referenced by the attribute history. This is deemed to be the last recorded date on which the data was managed. This may (but need not) be the date when the managed data was last processed.

Formal propositions:**DERIVE**

The derived attributes `number_of_uses`, `first_managing_application`, `first_managing_person`, `date_first_managed`, `last_managing_application`, `last_managing_person`, and `date_last_managed` will not appear when the instance is encoded in a STEP Part 21 file.

WRM60

The size of the list of unique instances of `managed_data_transaction` (derived from the SUPERTYPE `managed_data_item` by the attribute `number_of_uses`) shall be greater than zero. In other words, the data must have a history.

WRM61

The attribute `originating_application` must have a value. In other words, the data must be associated with an application. This effectively removes the OPTIONAL keyword in the SUPERTYPE entity `managed_data_item`.

WRM6

The instance of `managed_application_installation` derived by the attribute `first_managing_application` shall be instance equal to the instance of `managed_application_installation` referenced by the attribute `originating_application`. In other words, the installation of the DMC application that initiated the history tracking of the data must be the one deemed to be the originating application. This should be the application that provides the unique identifier.

WRM7

The first instance of `managed_data_transaction` in the list referenced by the attribute `history` shall be of the type `managed_data_creation`. In other words, the first data processing event that may be recorded for each instance of `managed_data_item` will be its creation.

WRM8

The attribute `original_data` shall not be assigned a value TRUE if the first instance of `managed_data_transaction` in the list referenced by the attribute `history` is of the type `managed_data_import`. In other words, the data cannot be deemed as original if has been imported. This applies when a DMC application adds meta-data to a set of 'unmanaged data' imported from an external source.

Note; an instance of the ANDOR SUPERTYPE `managed_data_transaction` can be both a `managed_data_import` and a `managed_data_creation`. A data processing event that imported data and created meta-data will be captured this way.

WRM37

The size of the set of instances of `managed_data_transaction` in the list referenced by the attribute `history` that are of the type `managed_data_creation` shall equal one. In other words, an instance of `managed_data_item` can only be created once.

WRM38

The instance of the entity referenced by the attribute `data_item` shall not be of the type `managed_data_deleted` if the value derived for the attribute `number_of_uses` is equal to one. That is, the first recorded data processing event cannot delete engineering data. In

other words, an instance of `managed_data_item` has to be created before the data associated with it can be deleted.

WRM39

The instance of the entity referenced by the attribute `data_item` shall not be of the type `managed_data_deleted` if the size of the list of instances of `managed_data_transaction` referenced by the attribute `history` that are of the type `managed_data_modification` is less than one. That is, there must be at least one instance of `managed_data_modification` associated with the `managed_data_item` if the engineering data has been deleted. In other words, the data processing event that deleted the engineering data is recorded by the instance of `managed_data_modification`. (Deletion is considered in CIS/2 to be a type of data modification.)

Informal propositions:

The ‘unique identifier’ that uniquely identifies the instance of the data item is provided by `instance_id`. This identification is qualified in this entity by the combination of the `application_id` and the `installation_id` (from the `managed_application_installation`). To ensure the uniqueness of the `instance_id`, the DMC application is required to assign and maintain these instance ids.

Notes:

New for CIS/2 2nd Edition – extracted from `managed_data_item`.

See diagram 1 of the EXPRESS-G diagrams in Appendix B.

19.3.9 managed_data_modification

Entity definition:

A type of `managed_data_transaction` that records the use of an installation of DMC application that modified (or deleted) managed data. It is asserted that **no** new unique identifiers were created during this event. The process merely maintained exiting unique identifiers.

The purpose of this entity is to constrain the use of its SUPERTYPE.

EXPRESS specification:

*)

ENTITY `managed_data_modification`

SUBTYPE OF (`managed_data_transaction`);

DERIVE

`modified_set` : SET [1:?] OF `managed_data_item` := `bag_to_set` USEDIN(`SELF`,
'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY');

WHERE

`WRM9` : `SELF`\`managed_data_transaction.new_ids_assigned` = FALSE;

END_ENTITY;

(*

Attribute definitions:

modified_set

This attribute derives the set of instances of `managed_data_item` that reference this instance of `managed_data_modification` (via the attribute `history`). This collates the managed data modified during the event recorded by this entity.

Formal propositions:**DERIVE**

The derived attribute `modified_set` will not appear when the instance is encoded in a STEP Part 21 file.

WRM9

The attribute `new_ids_assigned` (inherited from the SUPERTYPE `managed_data_transaction`) shall be assigned the value FALSE. That is, **no** new `instance_ids` have been assigned, which means **no** new instances of `managed_data_item` were created. It is implied that all the instances of `managed_data_item` are modified versions of previously existing instances.

Notes

New for CIS/2.

See Diagram 1 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

19.3.10 managed_data_transaction**Entity definition:**

This entity records one use of an installation of a DMC application by a particular person at a particular moment in time; i.e. a single (external) data transaction. (CIS/2 is not concerned with data processing that remains internal to an application – it is only concerned with those that are shared with the outside world. This is analogous to the formal issue of drawings during a construction project – although many revisions are made to drawings on a daily basis, only those that are ‘signed-off’ and issued beyond a company are recorded.)

This entity provides some of the meta-data for the associated engineering data. A collection of instances of this entity provide the history of the associated engineering data.

The SUBTYPEs of this entity categorize the data processing event as either an export or import, and/or either a creation or modification of the meta-data and its associated engineering data.

This entity is intended to capture external data exchange events (e.g. the import or export of managed neutral data between applications), rather than internal data processing events (e.g. the creation and modification of native data within an application). While there are likely to thousands of internal processing events, the number of external data processing events is likely to be much smaller (and more manageable).

EXPRESS specification:

*)

ENTITY `managed_data_transaction`

SUPERTYPE OF (ONEOF(`managed_data_export`, `managed_data_import`) ANDOR

ONEOF (`managed_data_creation`, `managed_data_modification`));

application : `managed_application_installation`;

user : `person_and_organization`;

processing_date : `date_and_time`;

new_ids_assigned : BOOLEAN;

life_cycle_stage : OPTIONAL label;

transaction_description : OPTIONAL text;
 INVERSE
 processed_items : SET [1:?] OF managed_data_item FOR history;
 UNIQUE
 URM3 : application, processing_date;
 END_ENTITY;
 (*)

Attribute definitions:

application

Declares the instance of managed_application_installation associated with the managed_data_transaction. It is implied that the instance of managed_data_transaction represents one use of the particular installation of the application represented here. There must be one, and only one, instance of managed_application_installation referenced by each instance of managed_data_transaction. An instance of managed_application_installation may be referenced by any number of instances of managed_data_transaction. In other words, a DMC application may be used many times, but this entity captures just one use of it.

user

Declares the instance of person_and_organization associated with the managed_data_transaction. It is implied that the instance of person_and_organization represents the user of the DMC application (referenced by the attribute application) at the time of the event recorded by the instance of managed_data_transaction. There must be one, and only one, instance of person_and_organization referenced by each instance of managed_data_transaction. An instance of person_and_organization may be referenced by any number of instances of managed_data_transaction. In other words, a person may use the DMC application many times, but this entity captures just one use of it. The person_and_organization referenced here may, but need not, be the owner (or registered keeper) of the installation of the application.

processing_date

Declares the instance of date_and_time associated with the managed_data_transaction. It is implied that the instance of date_and_time represents the moment in time when the DMC application (referenced by the attribute application) was used; i.e. when the data was processed. There must be one, and only one, instance of date_and_time referenced by each instance of managed_data_transaction. An instance of date_and_time may be referenced by **only one** instance of managed_data_transaction. In other words, this entity captures just one use of the installation of the DMC application.

new_ids_assigned

Declares whether or not new instance_ids have been assigned to the instances of managed_data_item that reference the managed_data_transaction. The value should be set to TRUE if new instance_ids have been assigned, or FALSE if new instance_ids have not been assigned. A new instance_id is only assigned to a managed_data_item when that instance is created. Thus, when new instances of managed_data_item are created which reference this entity, this attribute should be assigned a value of TRUE. When a DMC application modifies data, this attribute should be assigned a value of FALSE.

It should be noted that when a DMC application imports unmanaged data it may assign meta-data to that data, thus creating new instances of managed_data_item. Each one of these instances of managed_data_item will have a new instance_id. Here, the meta-data

is assigned to engineering data that is not original; i.e. the DMC application that created the meta-data is not the originator of the data.

life_cycle_stage

A short human-readable reference to the stage in the life cycle of the product data the managed_data_transaction occurred; e.g. “as planned”, “as designed”, “as built”.

transaction_description

An optional free text description of the data processing event recorded by the managed_data_transaction.

Formal propositions:

ANDOR SUPERTYPE

As this entity has been declared to be an ANDOR SUPERTYPE, it may be instantiated with more than one of its SUBTYPES. In such an event, a complex instance is produced, which is encoded in the STEP Part 21 file using external mapping.

processed_items

There must be at least one instance of managed_data_item associated with each instance of managed_data_transaction. This makes this entity existence dependent upon the entity managed_data_item. In other words, the use of a DMC application will only be recorded if meta-data is created or modified.

URM3

The combination of the instances of managed_application_installation (referenced by the attribute application) and date_and_time (referenced by the attribute processing_date) shall be unique to this instance of managed_data_transaction. In other words, an installation of a DMC application can only be used once at a particular moment in time.

It should be noted that this rule checks the uniqueness of the instance of date_and_time, rather than its numerical value. It is implied that the attribute values of the instances of calendar_date and local_time referenced by the instance of date_and_time capture a unique moment in time, although nothing in LPM/6 enforces this constraint.

Notes

New for CIS/2.

See Diagram 1 of the EXPRESS-G diagrams in Appendix B.

Unchanged for 2nd Edition.

19.4 Data Management function definitions

Data Management is one of only two subject areas that have EXPRESS functions defined within them. All other functions are imported from STEP Parts 41, 42, 43, and 44 via schema interfacing.

19.4.1 unique_data_item

Function definition:

This function calculates the number of times an entity instance is used by all the instances of managed_data_item in the information base. If the data_item is of the type managed_data_deleted, the function returns a value of UNKNOWN. If the data_item has

been used only once, the function returns a value of TRUE. Otherwise, function returns a value of FALSE (and the instance is not valid).

This function is called by the entity managed_data_item, to check the uniqueness of the managed data.

EXPRESS specification:

*)

FUNCTION unique_data_item

(item : select_data_item): LOGICAL;

LOCAL

bag_of_managed_items : BAG OF managed_data_item;

END_LOCAL;

IF ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_DELETED' IN TYPE OF(item))
THEN

RETURN (UNKNOWN);

END_IF;

-- find the managed_data_item in which the item is used

-- and add to the bag_of_managed_items

bag_of_managed_items := USEDIN(item,
'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.DATA_ITEM');

IF SIZEOF (bag_of_managed_items) = 1 THEN

RETURN (TRUE);

ELSE

RETURN (FALSE);

END_IF;

END_FUNCTION;

(*

Argument definitions:

item

The instance of the entity chosen via the SELECT type select_data_item that has been referenced by the attribute data_item of the instance of the entity managed_data_item calling this function. This is input to the function. (See also the definition for the type select_data_item.)

Notes

New for CIS/2.

This is one of only 4 functions specifically defined for LPM/6. All other functions are imported from STEP Parts 41, 42, 43, and 44 via schema interfacing.

20 REFERENCES

1. AMERICAN WELDING SOCIETY
Welding Science and Technology, Welding Handbook, 9th Ed., Vol. 1,
American Welding Society, USA, 2001
2. BS 499: PART 1: 1991
Welding terms and symbols: Part 1 Glossary for welding brazing and thermal cutting.
(Including amendment 9227: 1996)
British Standards Institution, London, 1991
3. CIMSTEEL
Design for Manufacture Guidelines
SCI Publication P150, The Steel Construction Institute, UK, 1995
4. COMPUTING SUPPLIERS FEDERATION
CAD/CAM Glossary
CAD/CAM Forum of the Computing Suppliers Federation
Worcester, UK, 1996
5. CROWLEY, A.J.
The Development and Implementation of a Product Model for Constructional Steelwork
PhD Thesis, University of Leeds, UK, 1998
6. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Volume 1 - Overview
SCI Publication P265, The Steel Construction Institute, UK, 2000
7. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Volume 2 - Implementation Guide
SCI Publication P266, The Steel Construction Institute, UK, 2000
8. CROWLEY, A.J., WARD, M.A. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Volume 3 - The Information Requirements
SCI Publication P267, The Steel Construction Institute, UK, (in preparation)
9. CROWLEY, A.J. SMITH, A.M. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Volume 5 - Conformance Requirements
SCI Publication P269, The Steel Construction Institute, UK, 2000
10. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Volume 6 - Worked Examples
SCI Publication P270, The Steel Construction Institute, UK, (in preparation)
11. ENV 1991-1: 1994
Eurocode 1: Basis of design and actions on structures
Part 1: Basis of Design
European Committee for Standardization (CEN), Brussels, 1994
12. ENV 1993-1-1: 1992
Eurocode 3: Design of steel structures
Part 1.1: General rules and rules for buildings
European Committee for Standardization (CEN), Brussels, 1992
13. HAMBURG, S.E. & HOLLAND, M.V.
Leaping ahead with EDI, pp. 42-48 in *Modern Steel Construction*, Vol. 39, No. 1
American Institute of Steel Construction, Inc., Chicago, USA, February 1999
14. ILLINGWORTH, V. (ED)
Oxford Dictionary of Computing,
Oxford University Press, Oxford, England, 1997

15. ISO 10303-1: 1994
Industrial automation systems - Product data representation and exchange
Part 1: Overview and Fundamental Principles
ISO/IEC, Geneva, Switzerland, 1994
16. ISO 10303-11: 1994
Industrial automation systems - Product data representation and exchange
Part 11: The EXPRESS Language Reference Manual
ISO/IEC, Geneva, Switzerland, 1994
17. ISO 10303-21: 1994
Industrial automation systems - Product data representation and exchange
Part 21: Clear text encoding of the exchange structure
ISO/IEC, Geneva, Switzerland, 1994
(Plus Technical Corrigendum 1 published 1996-08-15)
18. ISO 10303-22: 1998
Industrial automation systems - Product data representation and exchange
Part 22: Standard Data Access Interface
ISO/IEC, Geneva, Switzerland, 1998
19. ISO WD 10303-23: 1996
Industrial automation systems - Product data representation and exchange
Part 23: C++ Programming Language Binding to the Standard Data Access Interface
20. ISO WD 10303-24 (N379): 1995
Industrial automation systems - Product data representation and exchange
Part 24: C Language Late Binding to the Standard Data Access Interface
ISO/IEC, Geneva, Switzerland, 1995
21. ISO WD 10303-26: 1996
Industrial automation systems - Product data representation and exchange
Part 26: Interface Definition Language Binding to the Standard Data Access Interface.
ISO/IEC, Geneva, Switzerland, 1996
22. ISO DIS 10303-31: 1992
Industrial automation systems - Product data representation and exchange
Part 31: Conformance Testing Methodology & Framework: General Concepts.
ISO/IEC, Geneva, Switzerland, 1992
23. ISO 10303-41: 2000
Industrial automation systems - Product data representation and exchange
Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
24. ISO 10303-42: 2000
Industrial automation systems - Product data representation and exchange
Part 42: Integrated Generic Resources: Geometric & Topological Representation
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
25. ISO 10303-43: 2000
Industrial automation systems - Product data representation and exchange
Part 43: Integrated Generic Resources: Representation Structures
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
26. ISO 10303-44: 2000
Industrial automation systems - Product data representation and exchange
Part 44: Integrated Generic Resources: Product Structure Configuration
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
27. ISO 10303-45: 1998
Industrial automation systems - Product data representation and exchange
Part 45: Integrated Generic Resources: Materials

- ISO/IEC, Geneva, Switzerland, 1998
28. ISO 10303-104: 2000
Industrial automation systems - Product data representation and exchange
Part 104: Integrated application resources: Finite Element Analysis
ISO/IEC, Geneva, Switzerland, 2000
 29. ISO 10303-201: 1994
Industrial automation systems - Product data representation and exchange
Part 201: Application protocol: Explicit Draughting
ISO/IEC, Geneva, Switzerland, 1994
 30. ISO 10303-203: 1994
Industrial automation systems - Product data representation and exchange
Part 203: Application protocol: Configuration Controlled Design
ISO/IEC, Geneva, Switzerland, 1994
 31. ISO 10303-225: 1999
Industrial automation systems - Product data representation and exchange
Part 225: Application protocol: Building Elements using Explicit Shape Representation
ISO/IEC, Geneva, Switzerland, 1999
 32. ISO/WD 10303-230 (N551): 1996
Industrial automation systems - Product data representation and exchange
Part 230: Application protocol: Building Structural Frame: Steelwork
ISO TC184/SC4/WG3 (T12), 1996
 33. ISO/IEC 11578: 1996
Information technology - Open Systems Interconnection - Remote Procedure Call (RPC)
ISO/IEC, Geneva, Switzerland, 1996
 34. ISO 4063: 1998
Welding and allied processes – Nomenaclature of processes and reference numbers
ISO/IEC, Geneva, Switzerland, 1998
 35. MICROSOFT INC
COM and ActiveX Object Services Microsoft Visual Studio Online Documentation
Microsoft Developer Network (<http://msdn.microsoft.com/>)
 36. PARKER, S.P. (ED)
McGraw-Hill Dictionary of Engineering,
McGraw-Hill Inc., New York, USA, 1997
 37. THE OPEN GROUP
DCE 1.1: Remote Procedure Call, Open Group Technical Standard, Document number
C706, August 1997. (<http://www.opengroup.org/publications/catalog/c706.htm>)
 38. WILSON, P.R.
Euler Formulas and Geometric Modeling, pp.22-36 in *IEEE Computer Graphics
Applications*, Vol. 5, No 8, August 1985

APPENDIX A LPM/6 EXPRESS SCHEMA (LONG FORM)

(Schema Declaration *)*

SCHEMA STRUCTURAL_FRAME_SCHEMA;

(Object identifier *)*

(*
{cimsteel logical product model version (6) object (1) structural-frame-schema(1)}
*)

(Constant Declarations - **New for LPM/6** *)*

CONSTANT

dummy_gri : geometric_representation_item := representation_item("")||
geometric_representation_item();

dummy_tri : topological_representation_item := representation_item("")||
topological_representation_item();

END_CONSTANT; (* STEP Part 42 2nd Edition *)

(TYPE Declarations *)*

TYPE action_source_accidental

= ENUMERATION OF

(fire, impulse, impact, undefined);

END_TYPE;

TYPE action_source_permanent

= ENUMERATION OF

(dead,
self_weight,
prestress,
lack_of_fit,
undefined);

END_TYPE;

TYPE action_source_variable_long_term

= ENUMERATION OF

(live,
system_imperfection,
settlement,
temperature_effect,
undefined);

END_TYPE;

TYPE action_source_variable_short_term

= ENUMERATION OF

(buoyancy,
wind,
snow,
ice,
current,
wave,
rain,
undefined);

END_TYPE;

TYPE action_source_variable_transient

= ENUMERATION OF

(transport, erection, propping, undefined);

END_TYPE;

(* [Modified for LPM/6](#) *)

TYPE ahead_or_behind

= ENUMERATION OF

(ahead, exact, behind);

END_TYPE; (* STEP Part 41 (Modified in 2nd edition) *)

TYPE area_measure

= REAL;

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

(* [Modified for LPM/6](#) *)

TYPE assembly_component_select

= SELECT

(located_assembly,
located_part,
located_feature,
located_joint_system);

END_TYPE;

TYPE axis2_placement

= SELECT (axis2_placement_2d, axis2_placement_3d);

END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

TYPE b_spline_curve_form

= ENUMERATION OF

(polyline_form,
circular_arc,
elliptic_arc,

```

    parabolic_arc,
    hyperbolic_arc,
    unspecified);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE b_spline_surface_form
= ENUMERATION OF
    (plane_surf,
    cylindrical_surf,
    conical_surf,
    spherical_surf,
    toroidal_surf,
    surf_of_revolution,
    ruled_surf,
    generalised_cone,
    quadric_surf,
    surf_of_linear_extrusion,
    unspecified);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE bending_method
= ENUMERATION OF
    (hot_bend, cold_bend, undefined);
END_TYPE;

```

```

(* Modified for LPM/6 *)
TYPE boolean_operand
= SELECT
    (solid_model,
    half_space_solid,
    csg_primitive,
    boolean_result,
    half_space_2d);
END_TYPE; (* STEP Part 42 (Modified in 2nd edition) *)

```

```

TYPE boolean_operator
= ENUMERATION OF
    (union, intersection, difference);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

(* New for LPM/6 - see Issue 81 *)
TYPE boolean_value
= BOOLEAN;
END_TYPE;

```

(* **New for LPM/6** *)

```
TYPE brazing_type
= ENUMERATION OF
    (diffusion_brazing,
    dip_brazing,
    furnace_brazing,
    induction_brazing,
    infrared_brazing,
    resistance_brazing,
    torch_brazing);
END_TYPE;
```

```
TYPE buckling_direction
= ENUMERATION OF
    (x_dir, y_dir, z_dir);
END_TYPE;
```

(* **Modified for LPM/6** - see Issue 83 *)

```
TYPE cardinal_point_ref
= INTEGER;
WHERE
    WRTC1 : {0 < SELF < 100};
END_TYPE;
```

```
TYPE castellation_type
= ENUMERATION OF
    (circular, hexagonal, octagonal, undefined);
END_TYPE;
```

```
TYPE chemical_mechanism_type
= ENUMERATION OF
    (adhesive, grout, filler, sealant, undefined);
END_TYPE;
```

```
TYPE cleaning_method
= ENUMERATION OF
    (chemical_wash, blast_clean, undefined);
END_TYPE;
```

```
TYPE coating_method
= ENUMERATION OF
    (sprayed, brushed, dipped, electroplated, undefined);
END_TYPE;
```

```
TYPE coating_purpose
= ENUMERATION OF
    (corrosion_protection,
```



```

        fire_protection,
        aesthetic,
        undefined);
END_TYPE;

```

```

TYPE complexity_level
= ENUMERATION OF
    (low, medium, high);
END_TYPE;

```

```

TYPE connection_type
= ENUMERATION OF
    (pinned,
    semi_rigid_full_str,
    semi_rigid_partial_str,
    rigid_full_str,
    rigid_partial_str);
END_TYPE;

```

```

TYPE context_dependent_measure
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE count_measure
= NUMBER;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

TYPE csg_primitive
= SELECT
    (sphere,
    ellipsoid,
    block,
    right_angular_wedge,
    faceted_primitive,
    rectangular_pyramid,
    torus,
    right_circular_cone,
    eccentric_cone,
    right_circular_cylinder,
    cyclide_segment_solid,
    primitive_2d);
END_TYPE; (* STEP Part 42 (modified in 2nd edition) *)

```

```

TYPE csg_select
= SELECT
    (boolean_result, csg_primitive);

```

END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

TYPE curve_on_surface

= SELECT

(pcurve, surface_curve, composite_curve_on_surface);

END_TYPE; (* STEP Part 42 (Unchanged in 2nd edition) *)

TYPE cutting_type

= ENUMERATION OF

(sawn,

flame_cut,

sheared,

punched,

drilled,

laser,

abrasion,

undefined);

END_TYPE;

TYPE data_status_type

= ENUMERATION OF

(deleted, superseded, archived, erroneous, undefined);

END_TYPE;

TYPE day_in_month_number

= INTEGER;

WHERE

WRTD1 : { 1 <= SELF <= 31 };

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE degrees_rotation

= INTEGER;

WHERE

WRTD2 : {-180 < SELF <= 180};

END_TYPE;

TYPE derived_measure

= SELECT

(force_per_length_measure,

inertia_measure,

linear_acceleration_measure,

linear_stiffness_measure,

linear_velocity_measure,

mass_per_length_measure,

modulus_measure,

moment_measure,

rotational_acceleration_measure,

```

        rotational_stiffness_measure,
        rotational_velocity_measure);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE description_attribute_select
= SELECT
    (person_and_organization_role,
    person_and_organization,
    representation);
END_TYPE; (* STEP Part 41 2nd edition (reduced) *)

```

```

TYPE descriptive_measure
= STRING;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE dimension_count
= INTEGER;
WHERE
    WRTD3 : SELF > 0;
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE direct_or_indirect_action
= ENUMERATION OF
    (direct_action, indirect_action);
END_TYPE;

```

(* New for LPM/6 - see Issue 97 *)

```

TYPE drawing_class
= ENUMERATION OF
    (assembly_drawing,
    part_drawing,
    placement_drawing,
    undefined);
END_TYPE;

```

```

TYPE dynamic_analysis_type
= ENUMERATION OF
    (free_vibration,
    stressed_free_vibration,
    damped_vibration,
    linear_dynamic,
    response_spectrum,
    undefined);
END_TYPE;

```

```
TYPE elastic_or_plastic_resistance
= ENUMERATION OF
    (elastic_resistance, plastic_resistance);
END_TYPE;
```

```
TYPE element_surface_shape
= ENUMERATION OF
    (quadrilateral, triangle);
END_TYPE;
```

(* Modified for LPM/6 *)

```
TYPE element_volume_shape
= ENUMERATION OF
    (hexahedron_element,
    wedge_element,
    tetrahedron_element,
    pyramid_element);
END_TYPE;
```

```
TYPE fabrication_type
= ENUMERATION OF
    (rolled,
    welded,
    cold_formed,
    cast,
    forged,
    extruded,
    undefined);
END_TYPE;
```

```
TYPE force_measure
= REAL;
END_TYPE;
```

```
TYPE force_per_length_measure
= REAL;
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE founded_item_select
= SELECT
    (founded_item,
    representation_item);
END_TYPE; (* STEP Part 43 2nd Edition *)
```

```
TYPE frame_continuity
= ENUMERATION OF
```

```
(simple, continuous, semi_continuous);
END_TYPE;
```

```
TYPE frame_type
= ENUMERATION OF
    (space_frame,
     space_truss,
     plane_frame,
     plane_truss,
     grillage,
     undefined);
END_TYPE;
```

```
TYPE frequency_measure
= REAL;
END_TYPE;
```

```
TYPE geometric_set_select
= SELECT
    (point, curve, surface);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```
TYPE global_or_local_load
= ENUMERATION OF
    (global_load, local_load);
END_TYPE;
```

```
TYPE global_or_local_resistance
= ENUMERATION OF
    (global_resistance, local_resistance);
END_TYPE;
```

(* **New for LPM/6** - See Issues 80 and 103 *)

```
TYPE globally_unique_id
= STRING;
WHERE
    WRTG1 : LENGTH(SELF) > 0;
END_TYPE;
```

```
TYPE hour_in_day
= INTEGER;
WHERE
    WRTH1 : { 0 <= SELF < 24 };
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

(* New for LPM/6 *)

TYPE id_attribute_select

= SELECT

(action,
address,
group,
representation);

END_TYPE; (* STEP Part 41 2nd edition (reduced) *)

TYPE identifier

= STRING;

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE inertia_measure

= REAL;

WHERE

WRTI1 : (SELF > 0.0);

END_TYPE;

TYPE knot_type

= ENUMERATION OF

(UNIFORM_KNOTS,
UNSPECIFIED,
QUASI_UNIFORM_KNOTS,
PIECEWISE_BEZIER_KNOTS);

END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

TYPE label

= STRING;

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE left_or_right

= ENUMERATION OF

(left_hand, right_hand);

END_TYPE;

TYPE length_measure

= REAL;

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE linear_acceleration_measure

= REAL;

END_TYPE;

TYPE linear_stiffness_measure

= REAL;

WHERE

```

    WRTL1 : (SELF >= 0.0);
END_TYPE;

```

```

TYPE linear_velocity_measure
= REAL;
END_TYPE;

```

```

TYPE list_of_reversible_topology_item
= LIST [0:?] OF reversible_topology_item;
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE loading_status
= ENUMERATION OF
    (load_increasing,
    load_decreasing,
    load_constant,
    unloaded);
END_TYPE;

```

```

TYPE mass_measure
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE mass_per_length_measure
= REAL;
WHERE
    WRTM1 : (SELF > 0.0);
END_TYPE;

```

```

TYPE maximum_or_minimum
= ENUMERATION OF
    (maximum, minimum);
END_TYPE;

```

(* **New for LPM/6** - see Issue 81 *)

```

TYPE measure_select
= SELECT (measure_with_unit, measure_value, boolean_value);
END_TYPE;

```

```

TYPE measure_value
= SELECT
    (length_measure,
    mass_measure,
    time_measure,
    thermodynamic_temperature_measure,
    plane_angle_measure,
    solid_angle_measure,

```

```

    area_measure,
    volume_measure,
    ratio_measure,
    parameter_value,
    numeric_measure,
    force_measure,
    frequency_measure,
    pressure_measure,
    context_dependent_measure,
    descriptive_measure,
    positive_length_measure,
    positive_plane_angle_measure,
    positive_ratio_measure,
    count_measure,
    derived_measure);
END_TYPE; (* STEP Part 41 expanded (2nd Edition unchanged) *)

```

(* New for LPM/6 *)

```

TYPE member_beam_role
= ENUMERATION OF
    (edge_beam,
    eaves_beam,
    gantry_girder,
    joist,
    lintel,
    portal_rafter,
    purlin,
    rafter,
    ring_beam,
    side_rail,
    waling_beam);
END_TYPE;

```

```

TYPE member_beam_type
= ENUMERATION OF
    (box_girder,
    fish_bellied_beam,
    haunched_beam,
    plate_girder,
    stub_girder,
    tapered_beam);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE member_brace_type
= ENUMERATION OF
    (cross_brace,

```



```

    diagonal_brace,
    horizontal_brace,
    knee_brace,
    lateral_brace,
    longitudinal_brace,
    plan_brace,
    raker,
    sway_brace,
    vertical_brace);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE member_cable_type
= ENUMERATION OF
    (stay,
     suspension_cable,
     suspension_chain);
END_TYPE;

```

```

TYPE member_class
= ENUMERATION OF
    (primary_member,
     secondary_member,
     tertiary_member,
     undefined_class);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE member_column_type
= ENUMERATION OF
    (battened_column,
     box_column,
     compound_strut,
     portal_column);
END_TYPE;

```

```

TYPE member_cubic_type
= ENUMERATION OF
    (floor,
     stair,
     ramp,
     structural_core,
     structural_shell,
     undefined);
END_TYPE;

```

(* Modified for LPM/6 *)

```
TYPE member_linear_type
= ENUMERATION OF
    (beam,
     column,
     truss_element,
     brace,
     spring_element,
     cable,
     pipe,
     wire,
     tie,
     undefined,
     arch,
     beam_column);
END_TYPE;
```

```
TYPE member_planar_type
= ENUMERATION OF
    (wall,
     slab,
     stair_element,
     ramp_element,
     undefined,
     plate);
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE member_plate_type
= ENUMERATION OF
    (bearing_plate,
     diaphragm,
     flange,
     web);
END_TYPE;
```

```
TYPE member_role
= ENUMERATION OF
    (compression_member,
     tension_member,
     bending_member,
     combined_member,
     undefined_role);
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE member_slab_type
= ENUMERATION OF
    (flat_slab,
    ribbed_slab,
    solid_slab,
    trough_slab,
    voided_slab,
    waffle_slab);
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE member_wall_type
= ENUMERATION OF
    (load_bearing_wall,
    retaining_wall,
    shear_wall);
END_TYPE;
```

```
TYPE minute_in_hour
= INTEGER;
WHERE
    WRTM2 : { 0 <= SELF <= 59 };
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

```
TYPE minutes_rotation
= INTEGER;
WHERE
    WRTM3 : {0 <= SELF < 60};
END_TYPE;
```

```
TYPE modulus_measure
= REAL;
WHERE
    WRTM4 : (SELF > 0.0);
END_TYPE;
```

```
TYPE moment_measure
= REAL;
END_TYPE;
```

```
TYPE month_in_year_number
= INTEGER;
WHERE
    WRTM5 : { 1 <= SELF <= 12 };
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

(* New for LPM/6 *)

```
TYPE name_attribute_select
= SELECT
  (address,
   derived_unit,
   person_and_organization);
END_TYPE; (* STEP Part 41 2nd edition (Reduced) *)
```

```
TYPE numeric_measure
= NUMBER;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

```
TYPE orientation_select
= SELECT
  (plane_angle_measure_with_unit, direction);
END_TYPE;
```

```
TYPE parameter_value
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

```
TYPE part_select
= SELECT
  (part, design_part, located_part);
END_TYPE;
```

```
TYPE pcurve_or_surface
= SELECT
  (pcurve, surface);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```
TYPE plane_angle_measure
= REAL;
END_TYPE;
```

```
TYPE plane_stress_or_strain
= ENUMERATION OF
  (plane_stress, plane_strain, undefined);
END_TYPE;
```

```
TYPE positive_length_measure
= length_measure;
WHERE
  WRTP1 : SELF > 0.0;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

```

TYPE positive_plane_angle_measure
= plane_angle_measure;
WHERE
    WRTP2 : SELF > 0.0;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE positive_ratio_measure
= ratio_measure;
WHERE
    WRTP3 : SELF > 0.0;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE preferred_surface_curve_representation
= ENUMERATION OF
    (curve_3d, pcurve_s1, pcurve_s2);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE pressure_measure
= REAL;
END_TYPE;

```

```

(* New for LPM/6 - See Issue 15 *)
TYPE product_item_select
= SELECT
    (structural_frame_product, assembly_component_select);
END_TYPE;

```

```

TYPE project_select
= SELECT
    (project, zone_of_project);
END_TYPE;

```

```

(* Modified for LPM/6 *)
TYPE projected_or_true_length
= ENUMERATION OF
    (projected_length, true_length);
END_TYPE;

```

```

TYPE ratio_measure
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE reversible_topology
= SELECT
    (reversible_topology_item,
    list_of_reversible_topology_item,
    set_of_reversible_topology_item);

```

END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

TYPE reversible_topology_item

= SELECT

(edge,

path,

face,

face_bound,

closed_shell,

open_shell);

END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

TYPE role_select

= SELECT

(group_assignment);

END_TYPE; (* STEP Part 41 2nd edition (reduced) *)

TYPE rotational_acceleration_measure

= REAL;

END_TYPE;

TYPE rotational_stiffness_measure

= REAL;

WHERE

WRTR1 : (SELF >= 0.0);

END_TYPE;

TYPE rotational_velocity_measure

= REAL;

END_TYPE;

TYPE second_in_minute

= REAL;

WHERE

WRTS1 : { 0 <= SELF < 60 };

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE seconds_rotation

= REAL;

WHERE

WRTS2 : {0.0 <= SELF < 60.0};

END_TYPE;

TYPE select_analysis_item

= SELECT

```

(select_analysis_model_item,
select_loading_item,
select_response_item);
END_TYPE;

```

```

TYPE select_analysis_model_item
= SELECT
  (analysis_method,
  boundary_condition,
  element_eccentricity,
  element_node_connectivity,
  analysis_model,
  analysis_model_mapping,
  analysis_model_relationship,
  element,
  element_mapping,
  node,
  node_dependency,
  release);
END_TYPE;

```

```

TYPE select_data_item
= SELECT
  (managed_data_deleted,
  select_generic_item,
  select_analysis_item,
  select_design_item,
  select_physical_item,
  select_project_definition_item,
  select_structural_item);
END_TYPE;

```

```

TYPE select_data_source
= SELECT
  (managed_application_installation, step_file);
END_TYPE;

```

```

TYPE select_design_item
= SELECT
  (assembly_design,
  assembly_map,
  assembly_relationship,
  design_criterion,
  design_joint_system,
  design_part,
  design_result,
  effective_buckling_length,

```

```

        functional_role,
        resistance,
        restraint);
END_TYPE;

```

```
(* Modified for LPM/6 *)
```

```

TYPE select_generic_item
= SELECT
    (action,
    action_directive,
    action_method,
    address,
    approval,
    approval_status,
    box_domain,
    certification,
    certification_type,
    contract,
    contract_type,
    coordinated_universal_time_offset,
    date,
    date_and_time,
    derived_unit,
    derived_unit_element,
    description_attribute,
    dimensional_exponents,
    document,
    document_type,
    document_relationship,
    document_representation_type,
    document_usage_constraint,
    founded_item,
    functionally_defined_transformation,
    group,
    group_assignment,
    group_relationship,
    id_attribute,
    item_defined_transformation,
    local_time,
    measure_qualification,
    measure_with_unit,
    name_attribute,
    named_unit,
    object_role,
    organization,
    organization_relationship,
    person,

```



```

    person_and_organization,
    person_and_organization_role,
    representation,
    representation_context,
    representation_item,
    representation_map,
    representation_relationship,
    role_association,
    surface_patch,
    value_qualifier,
    versioned_action_request);
END_TYPE;

```

```

TYPE select_loading_item
= SELECT (
    applied_load,
    load_case,
    load,
    loaded_product,
    loading_combination,
    load_combination_occurrence,
    physical_action);
END_TYPE;

```

```

TYPE select_physical_item
= SELECT
    (located_item, located_part_joint);
END_TYPE;

```

```

TYPE select_project_definition_item
= SELECT
    (assembly,
    building,
    building_complex,
    currency_measure_with_unit,
    project,
    project_plan,
    project_plan_item,
    project_plan_item_relationship,
    project_organization,
    site,
    structure);
END_TYPE;

```

```

TYPE select_response_item
= SELECT (
    analysis_result,

```

```

        analysis_results_set,
        design_result,
        reaction);
END_TYPE;
```

(* [Modified for LPM/6](#) *)

```

TYPE select_structural_item
= SELECT
    (coord_system,
    grid,
    grid_intersection,
    grid_offset,
    geographical_location,
    item_cost_code,
    item_cost_code_assigned,
    item_property,
    item_property_assigned,
    item_reference,
    item_reference_assigned,
    item_ref_source,
    item_ref_source_documented,
    section_properties,
    setting_out_point,
    structural_frame_item,
    structural_frame_item_approved,
    structural_frame_item_certified,
    structural_frame_item_documented,
    structural_frame_item_priced,
    structural_frame_item_relationship,
    zone);
END_TYPE;
```

```

TYPE set_of_reversible_topology_item
= SET [1:?] OF reversible_topology_item;
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```

TYPE shell
= SELECT
    (vertex_shell,
    wire_shell,
    open_shell,
    closed_shell);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```

TYPE shop_or_site
= ENUMERATION OF
    (shop_process, site_process, undefined);
```

END_TYPE;

TYPE si_prefix
= ENUMERATION OF
(EXA,
PETA,
TERA,
GIGA,
MEGA,
KILO,
HECTO,
DECA,
DECI,
CENTI,
MILLI,
MICRO,
NANO,
PICO,
FEMTO,
ATTO);

END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE si_unit_name
= ENUMERATION OF
(METRE,
GRAM,
SECOND,
AMPERE,
KELVIN,
MOLE,
CANDELA,
RADIAN,
STERADIAN,
HERTZ,
NEWTON,
PASCAL,
JOULE,
WATT,
COULOMB,
VOLT,
FARAD,
OHM,
SIEMENS,
WEBER,
TESLA,
HENRY,
DEGREE_CELSIUS,

```
LUMEN,  
LUX,  
BECQUEREL,  
GRAY,  
SIEVERT);  
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

```
TYPE site_select  
= SELECT  
  (site,  
   located_site,  
   zone_of_site,  
   zone_of_building,  
   building_complex);  
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE soldering_type  
= ENUMERATION OF  
  (dip_soldering,  
   furnace_soldering,  
   induction_soldering,  
   infrared_soldering,  
   iron_soldering,  
   resistance_soldering,  
   torch_soldering,  
   wave_soldering);  
END_TYPE;
```

```
TYPE solid_angle_measure  
= REAL;  
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)
```

```
TYPE spatial_variation  
= ENUMERATION OF  
  (free_action, fixed_action);  
END_TYPE;
```

```
TYPE start_or_end_face  
= ENUMERATION OF  
  (start_face, end_face);  
END_TYPE;
```

```
TYPE static_analysis_type  
= ENUMERATION OF  
  (elastic_1st_order,  
   elastic_2nd_order,
```

```

    rigid_plastic,
    elasto_plastic,
    elastic_perfectly_plastic,
    undefined);
END_TYPE;

```

```

TYPE static_or_dynamic
= ENUMERATION OF
    (static, dynamic, quasi_dynamic);
END_TYPE;

```

```

TYPE structure_select
= SELECT
    (structure,
    located_structure,
    zone_of_structure,
    zone_of_building,
    located_assembly);
END_TYPE;

```

```

TYPE surface_model
= SELECT
    (shell_based_surface_model, face_based_surface_model);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE text
= STRING;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE thermodynamic_temperature_measure
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE time_measure
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

TYPE top_or_bottom
= ENUMERATION OF
    (top_edge, bottom_edge);
END_TYPE;

```

```

TYPE transformation
= SELECT
    (item_defined_transformation, functionally_defined_transformation);
END_TYPE; (* STEP Part 43 (unchanged in the 2nd edition) *)

```

TYPE transition_code
= ENUMERATION OF
 (discontinuous,
 continuous,
 cont_same_gradient,
 cont_same_gradient_same_curvature);
END_TYPE; (* STEP Part 42 (unchanged in the 2nd edition) *)

TYPE trimming_preference
= ENUMERATION OF
 (cartesian, parameter, unspecified);
END_TYPE; (* STEP Part 42 (unchanged in the 2nd edition) *)

TYPE trimming_select
= SELECT
 (cartesian_point, parameter_value);
END_TYPE; (* STEP Part 42 (unchanged in the 2nd edition) *)

TYPE unit
= SELECT
 (named_unit, derived_unit);
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

TYPE value_qualifier
= SELECT
 (precision_qualifier, type_qualifier, uncertainty_qualifier);
END_TYPE; (* STEP Part 45 (unchanged in TC1) *)

TYPE vector_or_direction
= SELECT
 (direction, vector);
END_TYPE; (* STEP Part 42 (unchanged in the 2nd edition) *)

TYPE volume_measure
= REAL;
END_TYPE; (* STEP Part 41 (unchanged in 2nd edition) *)

(* New for LPM/6 - see Issues 87 and 88 *)

TYPE weld_alignment
= ENUMERATION OF(staggered, chained);
END_TYPE;

(* New for LPM/6 - see Issues 87 and 88 *)

TYPE weld_backing_type
= ENUMERATION OF
 (none,
 permanent,

```

    copper_backing_bar,
    ceramic_tape,
    flare_backing_ring,
    permanent_backing_ring,
    removable_backing_ring,
    user_defined);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_configuration
= ENUMERATION OF
    (butt_joint,
    tee_joint,
    corner_joint,
    lap_joint,
    edge_joint,
    cruciform_joint,
    undefined);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_intermittent_rule
= ENUMERATION OF
    (none,
    fixed_rule,
    member_depth,
    percent_length);
END_TYPE;

```

```

TYPE weld_penetration
= ENUMERATION OF
    (full_penetration,
    deep_penetration,
    partial_penetration,
    undefined);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_shape_bevel
= ENUMERATION OF
    (flare_single_V,
    flare_double_V,
    flare_single_bevel,
    flare_double_bevel,
    single_bevel,
    double_bevel,
    single_V,

```

```

    double_V,
    single_J,
    double_J,
    single_U,
    double_U,
    user_defined);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_shape_but
= ENUMERATION OF (square, scarf, user_defined);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_sidedness
= ENUMERATION OF (one_side, both_sides);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_surface_shape
= ENUMERATION OF
    (flush,
     convex,
     concave,
     undefined);
END_TYPE;

```

(* **New for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_taper_type
= ENUMERATION OF (non_taper, one_side_taper, both_sides_taper);
END_TYPE;

```

(* **Modified for LPM/6** - see Issues 87 and 88 *)

```

TYPE weld_type
= ENUMERATION OF
    (butt_weld,
     fillet_weld,
     spot_weld,
     plug_weld,
     seam_weld,
     slot_weld,
     stud_weld,
     surfacing_weld,
     undefined);
END_TYPE;

```



```

TYPE welding_type
= ENUMERATION OF
    (fusion_weld,
     friction_weld,
     flash_weld,
     laser_weld,
     forge_weld,
     undefined);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE welding_type_arc
= ENUMERATION OF
    (generic_arc_welding,
     metal_arc_welding,
     manual_metal_arc_welding,
     gravity_arc_welding,
     self_shielded_arc_welding,
     submerged_arc_welding,
     gas_shielded_metal_arc_welding,
     metal_inert_gas_welding,
     metal_active_gas_welding,
     tubular_inert_gas_welding,
     tubular_active_gas_welding,
     tungsten_inert_gas_welding,
     atomic_hydrogen_welding,
     plasma_arc_welding,
     carbon_arc_welding,
     magnetically_impelled_arc_buttwelding);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE welding_type_beam
= ENUMERATION OF
    (electron_beam_welding,
     laser_beam_welding,
     gas_laser_welding);
END_TYPE;

```

(* New for LPM/6 *)

```

TYPE welding_type_gas
= ENUMERATION OF
    (generic_gas_welding,
     oxyacetylene_welding,
     oxyhydrogen_welding,
     oxypropane_welding);
END_TYPE;

```

(* New for LPM/6 *)

```
TYPE welding_type_other
= ENUMERATION OF
    (aluminothermic_welding,
     electroslog_welding,
     electrogas_welding,
     induction_welding,
     induction_butt_welding,
     induction_seam_welding,
     infrared_welding,
     percussion_welding);
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE welding_type_pressure
= ENUMERATION OF
    (generic_pressure_welding,
     ultrasonic_welding,
     friction_welding,
     forge_welding,
     explosive_welding,
     diffusion_welding,
     oxyfuel_gas_pressure_welding,
     cold_pressure_welding,
     hot_pressure_welding,
     roll_welding,
     high_frequency_pressure_welding);
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE welding_type_resistance
= ENUMERATION OF
    (generic_resistance_welding,
     spot_welding,
     seam_welding,
     projection_welding,
     flash_welding,
     resistance_butt_welding,
     high_frequency_resistance_welding);
END_TYPE;
```

(* New for LPM/6 *)

```
TYPE welding_type_stud
= ENUMERATION OF
    (generic_stud_welding,
     resistance_stud_welding,
```

```

        drawn_arc_stud_welding,
        friction_stud_welding);
END_TYPE;

```

```

TYPE wireframe_model
= SELECT (shell_based_wireframe_model, edge_based_wireframe_model);
END_TYPE; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

TYPE year_number
= INTEGER;
WHERE
    WRTY1 : SELF > 1000;
END_TYPE; (* based on STEP Part 41 *)

```

(* ENTITY Declarations *)

(* Modified for LPM/6 *)

```

ENTITY action
SUPERTYPE OF (executed_action);
    name : label;
    description : OPTIONAL text;
    chosen_method : action_method;
DERIVE
    id : identifier := get_id_value (SELF);
WHERE
    WRA34 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
        'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; (* STEP Part 41 (modified in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

ENTITY action_directive;
    name : label;
    description : OPTIONAL text;
    analysis: text;
    comment : text;
    requests: SET [1:?] OF versioned_action_request;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

(* Modified for LPM/6 *)

```

ENTITY action_method;
    name : label;
    description : OPTIONAL text;
    consequence : text;
    purpose : text;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

(* Modified for LPM/6 *)

ENTITY address

SUPERTYPE OF (personal_address ANDOR organizational_address);

internal_location : OPTIONAL label;
 street_number : OPTIONAL label;
 street : OPTIONAL label;
 postal_box : OPTIONAL label;
 town : OPTIONAL label;
 region : OPTIONAL label;
 postal_code : OPTIONAL label;
 country : OPTIONAL label;
 facsimile_number : OPTIONAL label;
 telephone_number : OPTIONAL label;
 electronic_mail_address : OPTIONAL label;
 telex_number : OPTIONAL label;

DERIVE

name : label := get_name_value (SELF);
 url : identifier := get_id_value (SELF);

WHERE

WRA1 : EXISTS (internal_location) OR
 EXISTS (street_number) OR
 EXISTS (postal_box) OR
 EXISTS (town) OR
 EXISTS (region) OR
 EXISTS (postal_code) OR
 EXISTS (country) OR
 EXISTS (facsimile_number) OR
 EXISTS (telephone_number) OR
 EXISTS (electronic_mail_address) OR
 EXISTS (telex_number);

END_ENTITY; (* STEP Part 41 2nd edition *)

ENTITY analysis_method

SUPERTYPE OF (ONEOF

(analysis_method_dynamic,
 analysis_method_pseudo_dynamic,
 analysis_method_static) ANDOR
 analysis_method_documented);

analysis_name : label;
 analysis_assumptions : OPTIONAL text;

END_ENTITY;

ENTITY analysis_method_documented

SUBTYPE OF (analysis_method);

documented_constraints : SET [1:?] OF document_usage_constraint;

END_ENTITY;

```

ENTITY analysis_method_dynamic
SUBTYPE OF (analysis_method);
    analysis_type : dynamic_analysis_type;
END_ENTITY;

```

```

ENTITY analysis_method_pseudo_dynamic
SUBTYPE OF (analysis_method);
    analysis_type : label;
END_ENTITY;

```

```

ENTITY analysis_method_static
SUBTYPE OF (analysis_method);
    analysis_type : static_analysis_type;
END_ENTITY;

```

```

ENTITY analysis_model
SUPERTYPE OF (ONEOF
    (analysis_model_2D,
    analysis_model_3D) ANDOR
    analysis_model_located ANDOR
    analysis_model_child);
    model_name : label;
    model_description : OPTIONAL text;
    model_type : frame_type;
    method_of_analysis : OPTIONAL analysis_method;
    coordinate_space_dimension : dimension_count;
INVERSE
    component_elements : SET [1:?] OF element
        FOR parent_model;
    component_nodes : SET [2:?] OF node
        FOR parent_model;
END_ENTITY;

```

```

ENTITY analysis_model_2D
SUBTYPE OF (analysis_model);
WHERE
    WRA2 : SELF\analysis_model.coordinate_space_dimension = 2;
    WRA3 : (SELF\analysis_model.model_type = PLANE_FRAME) OR
        (SELF\analysis_model.model_type = PLANE_TRUSS) OR
        (SELF\analysis_model.model_type = GRILLAGE);
END_ENTITY;

```

```

ENTITY analysis_model_3D
SUBTYPE OF (analysis_model);
WHERE
    WRA4 : SELF\analysis_model.coordinate_space_dimension = 3;

```

```

    WRA5 : (SELF\analysis_model.model_type = SPACE_FRAME) OR
           (SELF\analysis_model.model_type = SPACE_TRUSS);
END_ENTITY;

```

```

ENTITY analysis_model_child
SUBTYPE OF (analysis_model);
    parent_model : analysis_model;
WHERE
    WRA6 : parent_model :<>: (SELF);
    WRA7 : SELF\analysis_model.coordinate_space_dimension <=
           parent_model.coordinate_space_dimension;
END_ENTITY;

```

```

ENTITY analysis_model_located
SUBTYPE OF (analysis_model);
    model_coord_sys : coord_system;
WHERE
    WRA8 : SELF\analysis_model.coordinate_space_dimension <=
           model_coord_sys.coord_system_dimensionality;
END_ENTITY;

```

```

ENTITY analysis_model_mapping;
    mapped_analysis_model : analysis_model;
    represented_assemblies : SET [1:?] OF assembly;
END_ENTITY;

```

```

ENTITY analysis_model_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    relating_model : analysis_model;
    related_model : analysis_model;
WHERE
    WRA9 : relating_model :<>: related_model;
END_ENTITY;

```

```

ENTITY analysis_result
ABSTRACT SUPERTYPE OF (ONEOF
    (analysis_result_node,
     analysis_result_element_node,
     analysis_result_element));
    analysis_result_name : label;
    sign_convention : OPTIONAL text;
    results_for_analysis : analysis_method;
UNIQUE
    URA1 : analysis_result_name;
END_ENTITY;

```

```
ENTITY analysis_result_element
ABSTRACT SUPERTYPE OF (ONEOF
    (analysis_result_element_curve,
    analysis_result_element_surface,
    analysis_result_element_point,
    analysis_result_element_volume))
SUBTYPE OF (analysis_result);
END_ENTITY;
```

```
ENTITY analysis_result_element_curve
SUBTYPE OF (analysis_result_element);
    result_for_element_curve : element_curve;
    x_increasing : BOOLEAN;
    result_values : reaction;
    result_position : length_measure_with_unit;
    position_label : OPTIONAL label;
END_ENTITY;
```

```
ENTITY analysis_result_element_node
SUBTYPE OF (analysis_result);
    result_for_element_node : element_node_connectivity;
    result_values : reaction;
END_ENTITY;
```

```
ENTITY analysis_result_element_point
SUBTYPE OF (analysis_result_element);
    result_for_element_point : element_point;
    result_values : reaction;
END_ENTITY;
```

```
ENTITY analysis_result_element_surface
ABSTRACT SUPERTYPE OF (ONEOF
    (analysis_result_element_surface_stresses,
    analysis_result_element_surface_tractions))
SUBTYPE OF (analysis_result_element);
    result_for_element_surface : element_surface;
    result_position : point;
    position_label : OPTIONAL label;
END_ENTITY;
```

```
ENTITY analysis_result_element_surface_stresses
SUBTYPE OF (analysis_result_element_surface);
    direct_stress_sigma_y : pressure_measure_with_unit;
    membrane_stress_tau_yz : pressure_measure_with_unit;
    direct_stress_sigma_z : pressure_measure_with_unit;
END_ENTITY;
```

ENTITY analysis_result_element_surface_tractions

SUBTYPE OF (analysis_result_element_surface);

thrust_tz : OPTIONAL force_per_length_measure_with_unit;

bending_traction_my : force_measure_with_unit;

thrust_ty : OPTIONAL force_per_length_measure_with_unit;

torsional_traction_mzy : force_measure_with_unit;

torsional_traction_myz : force_measure_with_unit;

shear_traction_qz : force_per_length_measure_with_unit;

shear_traction_qy : force_per_length_measure_with_unit;

bending_traction_mz : force_measure_with_unit;

END_ENTITY;

ENTITY analysis_result_element_volume

ABSTRACT SUPERTYPE OF (analysis_result_element_volume_stress_tensor)

SUBTYPE OF (analysis_result_element);

result_for_element_volume : element_volume;

result_position : point;

position_label : OPTIONAL label;

END_ENTITY;

ENTITY analysis_result_element_volume_stress_tensor

SUBTYPE OF (analysis_result_element_volume);

shear_stress_tau_zy : pressure_measure_with_unit;

shear_stress_tau_xz : pressure_measure_with_unit;

normal_stress_sigma_z : pressure_measure_with_unit;

normal_stress_sigma_y : pressure_measure_with_unit;

normal_stress_sigma_x : pressure_measure_with_unit;

shear_stress_tau_zx : pressure_measure_with_unit;

shear_stress_tau_yz : pressure_measure_with_unit;

shear_stress_tau_yx : pressure_measure_with_unit;

shear_stress_tau_xy : pressure_measure_with_unit;

END_ENTITY;

ENTITY analysis_result_node

SUBTYPE OF (analysis_result);

result_for_node : node;

result_values : reaction;

END_ENTITY;

ENTITY analysis_results_set

SUPERTYPE OF (ONEOF

(analysis_results_set_basic,

analysis_results_set_combined,

analysis_results_set_envelope,

analysis_results_set_redistributed));

results_set_name : label;

component_results : SET [1:?] OF analysis_result;

END_ENTITY;

ENTITY analysis_results_set_basic
 SUBTYPE OF (analysis_results_set);
 basic_load_case : load_case;
 END_ENTITY;

ENTITY analysis_results_set_combined
 SUBTYPE OF (analysis_results_set);
 loading_combination_ref : loading_combination;
 END_ENTITY;

ENTITY analysis_results_set_envelope
 SUBTYPE OF (analysis_results_set);
 max_or_min : maximum_or_minimum;
 component_combinations : SET [1:?] OF analysis_results_set_combined;
 END_ENTITY;

ENTITY analysis_results_set_redistributed
 SUBTYPE OF (analysis_results_set);
 redistribution_factors : LIST [1:?] OF ratio_measure_with_unit;
 END_ENTITY;

ENTITY applied_load
 ABSTRACT SUPERTYPE OF (ONEOF(applied_load_static, applied_load_dynamic));
 applied_load_name : label;
 END_ENTITY;

ENTITY applied_load_dynamic
 SUPERTYPE OF (ONEOF
 (applied_load_dynamic_acceleration,
 applied_load_dynamic_velocity))
 SUBTYPE OF (applied_load);
 initial_value : OPTIONAL applied_load_static;
 final_value : OPTIONAL applied_load_static;
 maximum_value : OPTIONAL applied_load_static;
 minimum_value : OPTIONAL applied_load_static;
 number_of_cycles : OPTIONAL count_measure;
 load_duration : OPTIONAL time_measure_with_unit;
 load_frequency : OPTIONAL frequency_measure_with_unit;
 WHERE
 WRA10 : EXISTS (initial_value) OR EXISTS (final_value) OR
 EXISTS (maximum_value) OR EXISTS (minimum_value) OR
 EXISTS (number_of_cycles) OR EXISTS (load_duration) OR EXISTS
 (load_frequency) OR
 ('STRUCTURAL_FRAME_SCHEMA.APPLIED_LOAD_DYNAMIC_ACCELERATION'
 IN TYPEOF (SELF)) OR

```

        ('STRUCTURAL_FRAME_SCHEMA.APPLIED_LOAD_DYNAMIC_VELOCITY' IN
        TYPEOF (SELF));
END_ENTITY;

```

ENTITY applied_load_dynamic_acceleration

SUBTYPE OF (applied_load_dynamic);

```

    preset_acceleration_ax : OPTIONAL linear_acceleration_measure_with_unit;
    preset_acceleration_ay : OPTIONAL linear_acceleration_measure_with_unit;
    preset_acceleration_az : OPTIONAL linear_acceleration_measure_with_unit;
    preset_acceleration_arx : OPTIONAL rotational_acceleration_measure_with_unit;
    preset_acceleration_ary : OPTIONAL rotational_acceleration_measure_with_unit;
    preset_acceleration_arz : OPTIONAL rotational_acceleration_measure_with_unit;

```

WHERE

```

    WRA11 : EXISTS (preset_acceleration_ax) OR
            EXISTS (preset_acceleration_ay) OR
            EXISTS (preset_acceleration_az) OR
            EXISTS (preset_acceleration_arx) OR
            EXISTS (preset_acceleration_ary) OR
            EXISTS (preset_acceleration_arz);

```

END_ENTITY;

ENTITY applied_load_dynamic_velocity

SUBTYPE OF (applied_load_dynamic);

```

    preset_velocity_vx : OPTIONAL linear_velocity_measure_with_unit;
    preset_velocity_vy : OPTIONAL linear_velocity_measure_with_unit;
    preset_velocity_vz : OPTIONAL linear_velocity_measure_with_unit;
    preset_velocity_vrx : OPTIONAL rotational_velocity_measure_with_unit;
    preset_velocity_vry : OPTIONAL rotational_velocity_measure_with_unit;
    preset_velocity_vrz : OPTIONAL rotational_velocity_measure_with_unit;

```

WHERE

```

    WRA12 : EXISTS (preset_velocity_vx) OR
            EXISTS (preset_velocity_vy) OR
            EXISTS (preset_velocity_vz) OR
            EXISTS (preset_velocity_vrx) OR
            EXISTS (preset_velocity_vry) OR
            EXISTS (preset_velocity_vrz);

```

END_ENTITY;

ENTITY applied_load_static

ABSTRACT SUPERTYPE OF (ONEOF

```

    (applied_load_static_displacement,
    applied_load_static_force,
    applied_load_static_pressure))

```

SUBTYPE OF (applied_load);

END_ENTITY;

ENTITY applied_load_static_displacement

SUBTYPE OF (applied_load_static);

preset_displacement_dx : OPTIONAL length_measure_with_unit;
 preset_displacement_dy : OPTIONAL length_measure_with_unit;
 preset_displacement_dz : OPTIONAL length_measure_with_unit;
 preset_displacement_rx : OPTIONAL plane_angle_measure_with_unit;
 preset_displacement_ry : OPTIONAL plane_angle_measure_with_unit;
 preset_displacement_rz : OPTIONAL plane_angle_measure_with_unit;

WHERE

WRA13 : EXISTS (preset_displacement_dx) OR
 EXISTS (preset_displacement_dy) OR
 EXISTS (preset_displacement_dz) OR
 EXISTS (preset_displacement_rx) OR
 EXISTS (preset_displacement_ry) OR
 EXISTS (preset_displacement_rz);

END_ENTITY;

ENTITY applied_load_static_force

SUBTYPE OF (applied_load_static);

applied_force_fx : OPTIONAL force_measure_with_unit;
 applied_force_fy : OPTIONAL force_measure_with_unit;
 applied_force_fz : OPTIONAL force_measure_with_unit;
 applied_moment_mx : OPTIONAL moment_measure_with_unit;
 applied_moment_my : OPTIONAL moment_measure_with_unit;
 applied_moment_mz : OPTIONAL moment_measure_with_unit;

WHERE

WRA14 : EXISTS (applied_force_fx) OR
 EXISTS (applied_force_fy) OR
 EXISTS (applied_force_fz) OR
 EXISTS (applied_moment_mx) OR
 EXISTS (applied_moment_my) OR
 EXISTS (applied_moment_mz);

END_ENTITY;

ENTITY applied_load_static_pressure

SUBTYPE OF (applied_load_static);

applied_pressure_px : OPTIONAL pressure_measure_with_unit;
 applied_pressure_py : OPTIONAL pressure_measure_with_unit;
 applied_pressure_pz : OPTIONAL pressure_measure_with_unit;

WHERE

WRA15 : EXISTS (applied_pressure_px) OR
 EXISTS (applied_pressure_py) OR
 EXISTS (applied_pressure_pz);

END_ENTITY;

ENTITY approval;

status : approval_status;

```

    level : label;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY approval_status;
    name : label;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY area_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRA16 : 'STRUCTURAL_FRAME_SCHEMA.AREA_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
    WRA17 : 'STRUCTURAL_FRAME_SCHEMA.AREA_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY; (* based STEP Part 41 *)

```

```

ENTITY area_unit
SUBTYPE OF (named_unit);
WHERE
    WRA18 : (SELF\named_unit.dimensions.length_exponent = 2.0) AND
        (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
        (SELF\named_unit.dimensions.time_exponent = 0.0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY assemble
SUBTYPE OF (structural_frame_process);
    resulting_assembly : located_assembly;
    components : SET [1:?] OF assembly_component_select;
    required_processes : SET [1:?] OF structural_frame_process;
WHERE
    WRA29 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN
        TYPEOF (resulting_assembly.descriptive_assembly);
    WRA30 : SIZEOF(QUERY(component <* components | component :=:
        resulting_assembly)) = 0;
    WRA31 : SIZEOF(QUERY(process <* required_processes | process :=: (SELF))) = 0;
END_ENTITY;

```

(* Modified for LPM/6 *)

```

ENTITY assembly
ABSTRACT SUPERTYPE OF (ONEOF
    (assembly_design,
    assembly_manufacturing) ANDOR
    assembly_with_shape)

```

```

SUBTYPE OF (structural_frame_product);
    assembly_sequence_number : OPTIONAL INTEGER;
    complexity : OPTIONAL complexity_level;
DERIVE
    uses : SET [0:?] OF located_assembly := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.LOCATED_ASSEMBLY.
        DESCRIPTIVE_ASSEMBLY'));
END_ENTITY;

```

```

ENTITY assembly_design
SUPERTYPE OF (ONEOF
    (assembly_design_structural_frame,
    assembly_design_structural_member,
    assembly_design_structural_connection) ANDOR
    assembly_design_child)
SUBTYPE OF (assembly);
    designed : BOOLEAN;
    checked : BOOLEAN;
    roles : SET [0:?] OF functional_role;
    governing_criteria : SET [0:?] OF design_criterion;
END_ENTITY;

```

```

ENTITY assembly_design_child
SUBTYPE OF (assembly_design);
    parent_assemblies : SET [1:?] OF assembly_design;
WHERE
    WRA27 : SIZEOF(QUERY(assembly <* parent_assemblies | assembly := (SELF)) ) = 0;
END_ENTITY;

```

```

ENTITY assembly_design_structural_connection
SUPERTYPE OF (ONEOF
    (assembly_design_structural_connection_internal,
    assembly_design_structural_connection_external))
SUBTYPE OF (assembly_design);
    struc_connection_type : OPTIONAL connection_type;
END_ENTITY;

```

```

ENTITY assembly_design_structural_connection_external
SUBTYPE OF (assembly_design_structural_connection);
    connected_member : assembly_design_structural_member;
END_ENTITY;

```

```

ENTITY assembly_design_structural_connection_internal
SUBTYPE OF (assembly_design_structural_connection);
    connected_members : SET [2:?] OF assembly_design_structural_member;
END_ENTITY;

```

```

ENTITY assembly_design_structural_frame
SUBTYPE OF (assembly_design);
    type_of_frame : frame_type;
    continuity : OPTIONAL frame_continuity;
    sway_frame : OPTIONAL BOOLEAN;
    braced_frame : OPTIONAL BOOLEAN;
    bracing_frame : OPTIONAL BOOLEAN;
    frame_members : SET [0:?] OF assembly_design_structural_member;
    frame_connections : SET [0:?] OF assembly_design_structural_connection;
END_ENTITY;

```

(* [Modified for LPM/6](#) - DERIVE added *)

```

ENTITY assembly_design_structural_member
SUPERTYPE OF (ONEOF
    (assembly_design_structural_member_cubic,
     assembly_design_structural_member_linear,
     assembly_design_structural_member_planar))
SUBTYPE OF (assembly_design);
    key_member : OPTIONAL BOOLEAN;
    structural_member_use : member_role;
    structural_member_class : member_class;
DERIVE
    restraints : SET [0:?] OF restraint := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.RESTRAINT.RESTRAINED_MBR'));
    effective_lengths : SET [0:?] OF effective_buckling_length := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.EFFECTIVE_BUCKLING_LENGTH.
        APPLICABLE_MEMBER'));
END_ENTITY;

```

```

ENTITY assembly_design_structural_member_cubic
SUBTYPE OF (assembly_design_structural_member);
    cubic_member_type : member_cubic_type;
    cubic_member_components : SET [0:?] OF assembly_design_structural_member;
WHERE
    WRA28 : SIZEOF(QUERY(member <* cubic_member_components | member := (SELF))
    ) = 0;
END_ENTITY;

```

(* [Modified for LPM/6](#) - subtypes added *)

```

ENTITY assembly_design_structural_member_linear
SUPERTYPE OF (ONEOF (
    assembly_design_structural_member_linear_beam,
    assembly_design_structural_member_linear_brace,
    assembly_design_structural_member_linear_cable,
    assembly_design_structural_member_linear_column) ANDOR
    assembly_design_structural_member_linear_campered)
SUBTYPE OF (assembly_design_structural_member);
    linear_member_type : member_linear_type;

```

END_ENTITY;

(* New for LPM/6 *)

```
ENTITY assembly_design_structural_member_linear_beam
SUBTYPE OF (assembly_design_structural_member_linear);
    beam_type : SET [0:?] OF member_beam_type;
    beam_role : SET [0:?] OF member_beam_role;
    unrestrained_beam : LOGICAL;
    deep_beam : LOGICAL;
WHERE
    WRA35 : SELF\assembly_design_structural_member_linear.linear_member_type =
        BEAM;
    WRA42 : SIZEOF(SELF\assembly_design_structural_member.restraints) > 1;
    WRA43 : SIZEOF(SELF\assembly_design_structural_member.effective_lengths) > 0;
END_ENTITY;
```

(* New for LPM/6 *)

```
ENTITY assembly_design_structural_member_linear_brace
SUBTYPE OF (assembly_design_structural_member_linear);
    brace_type : member_brace_type;
WHERE
    WRA36 : SELF\assembly_design_structural_member_linear.linear_member_type =
        BRACE;
END_ENTITY;
```

(* New for LPM/6 *)

```
ENTITY assembly_design_structural_member_linear_cable
SUBTYPE OF (assembly_design_structural_member_linear);
    cable_type : member_cable_type;
WHERE
    WRA37 : SELF\assembly_design_structural_member_linear.linear_member_type =
        CABLE;
END_ENTITY;
```

(* New for LPM/6 - see Issue 92 *)

```
ENTITY assembly_design_structural_member_linear_campered
SUPERTYPE OF (assembly_design_structural_member_linear_campered_absolute
    ANDOR assembly_design_structural_member_linear_campered_relative)
SUBTYPE OF (assembly_design_structural_member_linear);
    camber_description : OPTIONAL text;
END_ENTITY;
```

(* New for LPM/6 - see Issue 92 *)

```
ENTITY assembly_design_structural_member_linear_campered_absolute
SUBTYPE OF (assembly_design_structural_member_linear_campered);
    absolute_offset_position : positive_length_measure_with_unit;
    absolute_offset_y : length_measure_with_unit;
    absolute_offset_z : length_measure_with_unit;
```

END_ENTITY;

(* New for LPM/6 - see Issue 92 *)

ENTITY assembly_design_structural_member_linear_campered_relative
 SUBTYPE OF (assembly_design_structural_member_linear_campered);
 relative_offset_position : ratio_measure_with_unit;
 relative_offset_y : ratio_measure_with_unit;
 relative_offset_z : ratio_measure_with_unit;
 END_ENTITY;

(* New for LPM/6 *)

ENTITY assembly_design_structural_member_linear_column
 SUBTYPE OF (assembly_design_structural_member_linear);
 column_type : SET [1:?] OF member_column_type;
 slender_column : LOGICAL;
 WHERE
 WRA38 : SELF\assembly_design_structural_member_linear.linear_member_type =
 COLUMN;
 WRA44 : SIZEOF(SELF\assembly_design_structural_member.restraints) > 1;
 WRA45 : SIZEOF(SELF\assembly_design_structural_member.effective_lengths) > 0;
 END_ENTITY;

(* Modified for LPM/6 - subtypes added *)

ENTITY assembly_design_structural_member_planar
 SUPERTYPE OF (ONEOF (
 assembly_design_structural_member_planar_plate,
 assembly_design_structural_member_planar_slab,
 assembly_design_structural_member_planar_wall))
 SUBTYPE OF (assembly_design_structural_member);
 planar_member_type : member_planar_type;
 planar_member_components : SET [0:?] OF
 assembly_design_structural_member_linear;
 END_ENTITY;

(* New for LPM/6 *)

ENTITY assembly_design_structural_member_planar_plate
 SUBTYPE OF (assembly_design_structural_member_planar);
 plate_type : member_plate_type;
 stiffened_plate : LOGICAL;
 thick_plate : LOGICAL;
 WHERE
 WRA39 : SELF\assembly_design_structural_member_planar.planar_member_type =
 PLATE;
 END_ENTITY;

(* New for LPM/6 *)

ENTITY assembly_design_structural_member_planar_slab
 SUBTYPE OF (assembly_design_structural_member_planar);


```

    slab_type : member_slab_type;
WHERE
    WRA40 : SELF\assembly_design_structural_member_planar.planar_member_type =
        SLAB;
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY assembly_design_structural_member_planar_wall
SUBTYPE OF (assembly_design_structural_member_planar);
    wall_type : member_wall_type;
WHERE
    WRA41 : SELF\assembly_design_structural_member_planar.planar_member_type =
        WALL;
END_ENTITY;

```

```

ENTITY assembly_manufacturing
SUPERTYPE OF (assembly_manufacturing_child)
SUBTYPE OF (assembly);
    surface_treatment : OPTIONAL text;
    assembly_sequence : OPTIONAL text;
    assembly_use : OPTIONAL text;
    place_of_assembly : OPTIONAL shop_or_site;
END_ENTITY;

```

```

ENTITY assembly_manufacturing_child
SUBTYPE OF (assembly_manufacturing);
    parent_assembly : assembly_manufacturing;
WHERE
    WRA19 : parent_assembly :<>: (SELF);
END_ENTITY;

```

```

ENTITY assembly_map;
    represented_assembly : assembly;
    representing_elements : SET [1:?] OF element;
END_ENTITY;

```

```

ENTITY assembly_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    related_assembly : assembly;
    relating_assembly : assembly;
WHERE
    WRA20 : related_assembly :<>: relating_assembly;
END_ENTITY;

```

```

ENTITY assembly_with_shape
SUBTYPE OF (assembly);

```

```

    shape : shape_representation_with_units;
END_ENTITY;

```

```

(* New for LPM/6 - see Issue 22 *)

```

```

ENTITY assembly_with_bounding_box
SUBTYPE OF (assembly_with_shape);
DERIVE
    bounding_box : SET [1:?] OF representation_item :=
        (SELF\assembly_with_shape.shape\representation.items);
WHERE
    WRA32 : SIZEOF(bounding_box) = 1;
    WRA33 : SIZEOF(QUERY(tmp <* bounding_box |
        ('STRUCTURAL_FRAME_SCHEMA.BLOCK') IN TYPEOF(tmp))) = 1;
END_ENTITY;

```

```

(* Modified for LPM/6 *)

```

```

ENTITY axis1_placement
SUBTYPE OF (placement);
    axis : OPTIONAL direction;
DERIVE
    z : direction := NVL(normalise(axis), dummy_gri || direction([0.0,0.0,1.0]));
WHERE
    WRA21 : SELF\geometric_representation_item.dim = 3;
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

ENTITY axis2_placement_2d
SUBTYPE OF (placement);
    ref_direction : OPTIONAL direction;
DERIVE
    p : LIST [2:2] OF direction := build_2axes(ref_direction);
WHERE
    WRA22 : SELF\geometric_representation_item.dim = 2;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

(* Modified for LPM/6 *)

```

```

ENTITY axis2_placement_3d
SUBTYPE OF (placement);
    axis : OPTIONAL direction;
    ref_direction : OPTIONAL direction;
DERIVE
    p : LIST [3:3] OF direction := build_axes(axis,ref_direction);
WHERE
    WRA23 : SELF\placement.location.dim = 3;
    WRA24 : (NOT (EXISTS (axis))) OR (axis.dim = 3);
    WRA25 : (NOT (EXISTS (ref_direction))) OR (ref_direction.dim = 3);
    WRA26 : (NOT (EXISTS (axis))) OR (NOT (EXISTS (ref_direction))) OR
        (cross_product(axis,ref_direction).magnitude > 0.0);

```

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

ENTITY b_spline_curve
SUPERTYPE OF ((ONEOF
  (b_spline_curve_with_knots,
   uniform_curve,
   quasi_uniform_curve,
   bezier_curve)) ANDOR
  rational_b_spline_curve)
SUBTYPE OF (bounded_curve);
  degree : INTEGER;
  control_points_list : LIST [2:?] OF cartesian_point;
  curve_form : b_spline_curve_form;
  closed_curve : LOGICAL;
  self_intersect : LOGICAL;
DERIVE
  upper_index_on_control_points : INTEGER := (SIZEOF(control_points_list) - 1);
  control_points : ARRAY [0:upper_index_on_control_points] OF cartesian_point :=
    list_to_array(control_points_list, 0, upper_index_on_control_points);
WHERE
  WRB1 : ('STRUCTURAL_FRAME_SCHEMA.UNIFORM_CURVE' IN TYPEOF(self))
    OR ('STRUCTURAL_FRAME_SCHEMA.QUASI_UNIFORM_CURVE' IN
      TYPEOF(self))
    OR ('STRUCTURAL_FRAME_SCHEMA.BEZIER_CURVE' IN TYPEOF(self))
    OR ('STRUCTURAL_FRAME_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS' IN
      TYPEOF(self));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY b_spline_curve_with_knots
SUBTYPE OF (b_spline_curve);
  knot_multiplicities : LIST [2:?] OF INTEGER;
  knots : LIST [2:?] OF parameter_value;
  knot_spec : knot_type;
DERIVE
  upper_index_on_knots : INTEGER := SIZEOF(knots);
WHERE
  WRB2 : constraints_param_b_spline(degree, upper_index_on_knots,
    upper_index_on_control_points, knot_multiplicities, knots);
  WRB3 : SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY b_spline_surface
SUPERTYPE OF ((ONEOF
  (b_spline_surface_with_knots,
   uniform_surface,
   quasi_uniform_surface,
   bezier_surface)) ANDOR
  rational_b_spline_surface)

```

```

SUBTYPE OF (bounded_surface);
  u_degree : INTEGER;
  v_degree : INTEGER;
  control_points_list : LIST [2:?] OF LIST [2:?] OF cartesian_point;
  surface_form : b_spline_surface_form;
  u_closed : LOGICAL;
  v_closed : LOGICAL;
  self_intersect : LOGICAL;
DERIVE
  u_upper : INTEGER := SIZEOF(control_points_list) - 1;
  v_upper : INTEGER := SIZEOF(control_points_list[1]) - 1;
  control_points : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF cartesian_point :=
    make_array_of_array (control_points_list, 0, u_upper, 0, v_upper);
WHERE
  WRB4 : ('STRUCTURAL_FRAME_SCHEMA.UNIFORM_SURFACE' IN TYPEOF(SELF))
    OR ('STRUCTURAL_FRAME_SCHEMA.QUASI_UNIFORM_SURFACE' IN
      TYPEOF(SELF))
    OR ('STRUCTURAL_FRAME_SCHEMA.BEZIER_SURFACE' IN TYPEOF(SELF))
    OR ('STRUCTURAL_FRAME_SCHEMA.B_SPLINE_SURFACE_WITH_KNOTS' IN
      TYPEOF(SELF));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY b_spline_surface_with_knots
SUBTYPE OF (b_spline_surface);
  u_multiplicities : LIST [2:?] OF INTEGER;
  v_multiplicities : LIST [2:?] OF INTEGER;
  u_knots : LIST [2:?] OF parameter_value;
  v_knots : LIST [2:?] OF parameter_value;
  knot_spec : knot_type;
DERIVE
  knot_u_upper : INTEGER := SIZEOF(u_knots);
  knot_v_upper : INTEGER := SIZEOF(v_knots);
WHERE
  WRB5 : constraints_param_b_spline(SELF\b_spline_surface.u_degree,
    knot_u_upper, SELF\b_spline_surface.u_upper,
    u_multiplicities, u_knots);
  WRB6 : constraints_param_b_spline(SELF\b_spline_surface.v_degree,
    knot_v_upper, SELF\b_spline_surface.v_upper,
    v_multiplicities, v_knots);
  WRB7 : SIZEOF(u_multiplicities) = knot_u_upper;
  WRB8 : SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY b_spline_volume
SUPERTYPE OF (ONEOF(
  b_spline_volume_with_knots,

```

```

    uniform_volume,
    quasi_uniform_volume,
    bezier_volume) ANDOR
    rational_b_spline_volume)
SUBTYPE OF (volume);
    u_degree : INTEGER;
    v_degree : INTEGER;
    w_degree : INTEGER;
    control_points_list : LIST [2:?] OF LIST [2:?] OF LIST [2:?] OF cartesian_point;
DERIVE
    u_upper : INTEGER := SIZEOF(control_points_list) - 1;
    v_upper : INTEGER := SIZEOF(control_points_list[1]) - 1;
    w_upper : INTEGER := SIZEOF(control_points_list[1][1]) - 1;
    control_points : ARRAY [0:u_upper] OF ARRAY [0:v_upper]
        OF ARRAY [0:w_upper] OF cartesian_point
        := make_array_of_array_of_array
            (control_points_list,0,u_upper,0,v_upper,0,w_upper );
WHERE
    WRB17 : ('STRUCTURAL_FRAME_SCHEMA.BEZIER_VOLUME' IN TYPEOF(SELF))
        OR ('STRUCTURAL_FRAME_SCHEMA.UNIFORM_VOLUME' IN TYPEOF(SELF))
        OR ('STRUCTURAL_FRAME_SCHEMA.QUASI_UNIFORM_VOLUME' IN
            TYPEOF(SELF))
        OR ('STRUCTURAL_FRAME_SCHEMA.B_SPLINE_VOLUME_WITH_KNOTS' IN
            TYPEOF(SELF)) ;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY b_spline_volume_with_knots
SUBTYPE OF (b_spline_volume);
    u_multiplicities : LIST [2:?] OF INTEGER;
    v_multiplicities : LIST [2:?] OF INTEGER;
    w_multiplicities : LIST [2:?] OF INTEGER;
    u_knots : LIST [2:?] OF parameter_value;
    v_knots : LIST [2:?] OF parameter_value;
    w_knots : LIST [2:?] OF parameter_value;
DERIVE
    knot_u_upper : INTEGER := SIZEOF(u_knots);
    knot_v_upper : INTEGER := SIZEOF(v_knots);
    knot_w_upper : INTEGER := SIZEOF(w_knots);
WHERE
    WRB18 : constraints_param_b_spline(SELF\b_spline_volume.u_degree,
        knot_u_upper, SELF\b_spline_volume.u_upper,
        u_multiplicities, u_knots);
    WRB19 : constraints_param_b_spline(SELF\b_spline_volume.v_degree,
        knot_v_upper, SELF\b_spline_volume.v_upper,
        v_multiplicities, v_knots);
    WRB20 : constraints_param_b_spline(SELF\b_spline_volume.w_degree,

```

```

        knot_w_upper, SELF\b_spline_volume.w_upper,
        w_multiplicities, w_knots);
WRB21 : SIZEOF(u_multiplicities) = knot_u_upper;
WRB22 : SIZEOF(v_multiplicities) = knot_v_upper;
WRB23 : SIZEOF(w_multiplicities) = knot_w_upper;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY bend
SUBTYPE OF (structural_frame_process);
    method : bending_method;
END_ENTITY;

```

```

ENTITY bezier_curve
SUBTYPE OF (b_spline_curve);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY bezier_surface
SUBTYPE OF (b_spline_surface);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY bezier_volume
SUBTYPE OF (b_spline_volume);
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY block
SUBTYPE OF (geometric_representation_item);
    position : axis2_placement_3d;
    x : positive_length_measure;
    y : positive_length_measure;
    z : positive_length_measure;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY block_volume
SUBTYPE OF (volume);
    position : axis2_placement_3d;
    x : positive_length_measure;
    y : positive_length_measure;
    z : positive_length_measure;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY boolean_result
SUBTYPE OF (geometric_representation_item);
    operator : boolean_operator;
    first_operand : boolean_operand;
    second_operand : boolean_operand;

```

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```
ENTITY boundary_condition
ABSTRACT SUPERTYPE OF (ONEOF
    (boundary_condition_logical,
    boundary_condition_spring_linear,
    boundary_condition_spring_non_linear) ANDOR
    boundary_condition_skewed);
    boundary_condition_name : label;
    boundary_condition_description : OPTIONAL text;
INVERSE
    restrained_nodes : SET [1:?] OF node FOR restraints;
END_ENTITY;
```

```
ENTITY boundary_condition_logical
SUBTYPE OF (boundary_condition);
    bc_x_displacement_free : LOGICAL;
    bc_y_displacement_free : LOGICAL;
    bc_z_displacement_free : LOGICAL;
    bc_x_rotation_free : LOGICAL;
    bc_y_rotation_free : LOGICAL;
    bc_z_rotation_free : LOGICAL;
END_ENTITY;
```

```
ENTITY boundary_condition_skewed
SUBTYPE OF (boundary_condition);
    x_skew_angle : OPTIONAL plane_angle_measure_with_unit;
    y_skew_angle : OPTIONAL plane_angle_measure_with_unit;
    z_skew_angle : OPTIONAL plane_angle_measure_with_unit;
WHERE
    WRB9 : EXISTS (x_skew_angle) OR EXISTS (y_skew_angle) OR EXISTS
        (z_skew_angle);
END_ENTITY;
```

```
ENTITY boundary_condition_spring_linear
SUPERTYPE OF (boundary_condition_warping)
SUBTYPE OF (boundary_condition);
    bc_x_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    bc_y_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    bc_z_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    bc_x_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
    bc_y_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
    bc_z_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
WHERE
    WRB10 : EXISTS (bc_x_displacement) OR
        EXISTS (bc_y_displacement) OR
        EXISTS (bc_z_displacement) OR
```

```

        EXISTS (bc_x_rotation) OR
        EXISTS (bc_y_rotation) OR
        EXISTS (bc_z_rotation);
END_ENTITY;

```

```

ENTITY boundary_condition_spring_non_linear
SUBTYPE OF (boundary_condition);
    change_values : LIST [2:?] OF measure_with_unit;
    values : LIST [2:?] OF boundary_condition_spring_linear;
DERIVE
    number_of_values : INTEGER := SIZEOF(change_values);
WHERE
    WRB12 : SIZEOF(values) = SIZEOF(change_values);
END_ENTITY;

```

```

ENTITY boundary_condition_warping
SUBTYPE OF (boundary_condition_spring_linear);
    bc_warping : rotational_stiffness_measure_with_unit;
END_ENTITY;

```

```

ENTITY boundary_curve
SUBTYPE OF (composite_curve_on_surface);
WHERE
    WRB13 : SELF\composite_curve.closed_curve;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY bounded_curve
SUPERTYPE OF (ONEOF
    (composite_curve,
    polyline,
    b_spline_curve,
    trimmed_curve,
    bounded_pcurve,
    bounded_surface_curve))
SUBTYPE OF (curve);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY bounded_pcurve
SUBTYPE OF (pcurve, bounded_curve);
WHERE
    WRB14 : ('STRUCTURAL_FRAME_SCHEMA.BOUNDED_CURVE' IN
        TYPEOF(SELF\pcurve.reference_to_curve.items[1]));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY bounded_surface
SUPERTYPE OF (ONEOF
    (b_spline_surface,

```



```

        rectangular_trimmed_surface,
        curve_bounded_surface,
        rectangular_composite_surface))
SUBTYPE OF (surface);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY bounded_surface_curve
SUBTYPE OF (surface_curve, bounded_curve);
WHERE
    WRB15 : ('STRUCTURAL_FRAME_SCHEMA.BOUNDED_CURVE' IN
        TYPEOF(SELF\surface_curve.curve_3d));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY box_domain;
    corner : cartesian_point;
    xlength : positive_length_measure;
    ylength : positive_length_measure;
    zlength : positive_length_measure;
WHERE
    WRB16 : SIZEOF(QUERY(item <* USEDIN(SELF,")|
        NOT ('STRUCTURAL_FRAME_SCHEMA.BOXED_HALF_SPACE'
            IN TYPEOF(item)))) = 0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY boxed_half_space
SUBTYPE OF (half_space_solid);
    enclosure : box_domain;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY braze
SUBTYPE OF (structural_frame_process);
    braze_type : brazing_type;
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY brep_2d
SUBTYPE OF (solid_model);
    extent : face;
WHERE
    WRB25 : SIZEOF (['STRUCTURAL_FRAME_SCHEMA.FACE_SURFACE',
        'STRUCTURAL_FRAME_SCHEMA.SUBFACE',
        'STRUCTURAL_FRAME_SCHEMA.ORIENTED_FACE'] *
        TYPEOF (SELF.extent)) = 0;
    WRB26 : SIZEOF (QUERY (bnds <* extent.bounds |
        NOT ('STRUCTURAL_FRAME_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound))) )
        = 0;

```

```

WRB27 : SIZEOF (QUERY (bnds <* extent.bounds |
    'STRUCTURAL_FRAME_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF(bnds))) = 1;
WRB28 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* extent.bounds |
    'STRUCTURAL_FRAME_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
    NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list | NOT
        (('STRUCTURAL_FRAME_SCHEMA.EDGE_CURVE' IN TYPEOF(oe.edge_element))
        AND (oe.edge_element\geometric_representation_item.dim = 2)))) = 0))) = 0;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY brep_with_voids
SUBTYPE OF (manifold_solid_brep);
    voids : SET [1:?] OF oriented_closed_shell;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY building
SUPERTYPE OF (building_with_shape)
SUBTYPE OF (structural_frame_item);
    building_class : OPTIONAL label;
    owner : OPTIONAL person_and_organization;
    building_structures : OPTIONAL LIST [1:?] OF structure;
UNIQUE
    URB1 : SELF\structural_frame_item.item_number,
    SELF\structural_frame_item.item_name;
END_ENTITY;

```

```

ENTITY building_complex
SUBTYPE OF (structural_frame_item);
    building_site : site;
    buildings : LIST [1:?] OF building;
UNIQUE
    URB2 : SELF\structural_frame_item.item_number,
    SELF\structural_frame_item.item_name;
END_ENTITY;

```

```

ENTITY building_with_shape
SUBTYPE OF (building);
    shape : shape_representation_with_units;
END_ENTITY;

```

```

ENTITY calendar_date
SUBTYPE OF (date);
    day_component : day_in_month_number;
    month_component : month_in_year_number;
WHERE
    WRC1: valid_calendar_date (SELF);

```

END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

(* Modified for LPM/6 *)

ENTITY cartesian_point

SUPERTYPE OF (ONEOF(cylindrical_point, polar_point, spherical_point))

SUBTYPE OF (point);

coordinates : LIST [1:3] OF length_measure;

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

ENTITY cartesian_transformation_operator

SUPERTYPE OF (ONEOF

(cartesian_transformation_operator_3d, cartesian_transformation_operator_2d))

SUBTYPE OF (geometric_representation_item, functionally_defined_transformation);

axis1 : OPTIONAL direction;

axis2 : OPTIONAL direction;

local_origin : cartesian_point;

scale : OPTIONAL REAL;

DERIVE

scl : REAL := NVL(scale, 1.0);

WHERE

WRC2 : scl > 0.0;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY cartesian_transformation_operator_2d

SUBTYPE OF (cartesian_transformation_operator);

DERIVE

u : LIST [2:2] OF direction :=

base_axis(2,SELF\cartesian_transformation_operator.axis1,
SELF\cartesian_transformation_operator.axis2,?);

WHERE

WRC3 : SELF\geometric_representation_item.dim = 2;

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

ENTITY cartesian_transformation_operator_3d

SUBTYPE OF (cartesian_transformation_operator);

axis3 : OPTIONAL direction;

DERIVE

u : LIST [3:3] OF direction :=

base_axis(3,SELF\cartesian_transformation_operator.axis1,
SELF\cartesian_transformation_operator.axis2,axis3);

WHERE

WRC4 : SELF\geometric_representation_item.dim = 3;

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

ENTITY certification;

name : label;

purpose : label;

```

    kind : certification_type;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY certification_type;
    description : text;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY chemical_mechanism
SUBTYPE OF (structural_frame_product);
    layer_thickness : positive_length_measure_with_unit;
    layer_design_strength : OPTIONAL pressure_measure_with_unit;
    layer_type : chemical_mechanism_type;
END_ENTITY;

```

```

ENTITY circle
SUBTYPE OF (conic);
    radius : positive_length_measure;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

(* New for LPM/6 *)
ENTITY circular_area
SUBTYPE OF (primitive_2d);
    centre: cartesian_point;
    radius: positive_length_measure;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

(* New for LPM/6 *)
ENTITY clothoid
SUBTYPE OF (curve);
    position : axis2_placement;
    clothoid_constant : length_measure;
END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

```

```

ENTITY closed_shell
SUBTYPE OF (connected_face_set);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY coating
SUBTYPE OF (structural_frame_product);
    primary_purpose : coating_purpose;
END_ENTITY;

```

```

ENTITY composite_curve
SUPERTYPE OF (composite_curve_on_surface)
SUBTYPE OF (bounded_curve);
    segments : LIST [1:?] OF composite_curve_segment;
    self_intersect : LOGICAL;

```

DERIVE

n_segments : INTEGER := SIZEOF(segments);

closed_curve : LOGICAL := segments[n_segments].transition <> discontinuous;

WHERE

WRC5 : ((NOT closed_curve) AND (SIZEOF(QUERY(temp <* segments |
temp.transition = discontinuous)) = 1)) OR
((closed_curve) AND (SIZEOF(QUERY(temp <* segments |
temp.transition = discontinuous)) = 0));

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY composite_curve_on_surface

SUBTYPE OF (composite_curve);

DERIVE

basis_surface : SET [0:2] OF surface := get_basis_surface(SELF);

WHERE

WRC6 : SIZEOF(basis_surface) > 0;

WRC7 : constraints_composite_curve_on_surface(SELF);

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* [Modified for LPM/6](#) *)

ENTITY composite_curve_segment

SUBTYPE OF (founded_item);

transition : transition_code;

same_sense : BOOLEAN;

parent_curve : curve;

INVERSE

using_curves : BAG [1:?] OF composite_curve FOR segments;

WHERE

WRC8 : ('STRUCTURAL_FRAME_SCHEMA.BOUNDED_CURVE' IN
TYPEOF(parent_curve));

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

ENTITY conic

SUPERTYPE OF (ONEOF(circle, ellipse, hyperbola, parabola))

SUBTYPE OF (curve);

position : axis2_placement;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY conical_surface

SUBTYPE OF (elementary_surface);

radius : length_measure;

semi_angle : plane_angle_measure;

WHERE

WRC9 : radius >= 0.0;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

ENTITY connected_edge_set
SUBTYPE OF (topological_representation_item);
    ces_edges : SET [1:?] OF edge;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY connected_face_set
SUPERTYPE OF (ONEOF(open_shell, closed_shell))
SUBTYPE OF (topological_representation_item);
    cfs_faces : SET [1:?] OF face;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY context_dependent_unit
SUBTYPE OF (named_unit);
    name : label;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY contract;
    name : label;
    purpose : label;
    kind : contract_type;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY contract_type;
    description : text;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY conversion_based_unit
SUBTYPE OF (named_unit);
    name : label;
    conversion_factor : measure_with_unit;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

(* **New for LPM/6** *)

```

ENTITY convex_hexahedron
SUBTYPE OF (faceted_primitive);
WHERE
    WRC27 : SIZEOF(points) = 8 ;
    WRC28 : above_plane(points[1], points[2], points[3], points[4]) = 0.0;
    WRC29 : above_plane(points[5], points[8], points[7], points[6]) = 0.0;
    WRC30 : above_plane(points[1], points[4], points[8], points[5]) = 0.0;
    WRC31 : above_plane(points[4], points[3], points[7], points[8]) = 0.0;
    WRC32 : above_plane(points[3], points[2], points[6], points[7]) = 0.0;
    WRC33 : above_plane(points[1], points[5], points[6], points[2]) = 0.0;
    WRC34 : same_side([points[1], points[2], points[3]],
        [points[5], points[6], points[7], points[8]]);
    WRC35 : same_side([points[1], points[4], points[8]],
        [points[3], points[7], points[6], points[2]]);

```

```

WRC36 : same_side([points[1], points[2], points[5]],
  [points[3], points[7], points[8], points[4]]);
WRC37 : same_side([points[5], points[6], points[7]],
  [points[1], points[2], points[3], points[4]]);
WRC38 : same_side([points[3], points[7], points[6]],
  [points[1], points[4], points[8], points[5]]);
WRC39 : same_side([points[3], points[7], points[8]],
  [points[1], points[5], points[6], points[2]]);
END_ENTITY; (* STEP Part 41 (new in 2nd edition) *)

```

```

ENTITY coord_system
ABSTRACT SUPERTYPE OF (ONEOF
  (coord_system_cartesian_2d,
  coord_system_cartesian_3d,
  coord_system_spherical,
  coord_system_cylindrical) ANDOR
  coord_system_child);
coord_system_name : label;
coord_system_use : label;
sign_convention : OPTIONAL text;
coord_system_dimensionality : dimension_count;
END_ENTITY;

```

```

ENTITY coord_system_cartesian_2d
SUBTYPE OF (coord_system);
  axes_definition : axis2_placement_2d;
DERIVE
  origin_1 : REAL := axes_definition.location\cartesian_point.coordinates[1];
  origin_2 : REAL := NVL(axes_definition.location\cartesian_point.coordinates[2], 0.0);
WHERE
  WRC10 : SELF\coord_system.coord_system_dimensionality = 2;
  WRC11 : SIZEOF (axes_definition.location\cartesian_point.coordinates) = 2;
END_ENTITY;

```

```

ENTITY coord_system_cartesian_3d
SUBTYPE OF (coord_system);
  axes_definition : axis2_placement_3d;
DERIVE
  origin_x : REAL := axes_definition.location\cartesian_point.coordinates[1];
  origin_y : REAL := NVL(axes_definition.location\cartesian_point.coordinates[2], 0.0);
  origin_z : REAL := NVL(axes_definition.location\cartesian_point.coordinates[3], 0.0);
WHERE
  WRC12 : SELF\coord_system.coord_system_dimensionality = 3;
  WRC13 : SIZEOF (axes_definition.location\cartesian_point.coordinates) = 3;
END_ENTITY;

```

```

ENTITY coord_system_child
SUBTYPE OF (coord_system);
    parent_coord_system : coord_system;
WHERE
    WRC14 : parent_coord_system :<>: (SELF);
    WRC15 : SELF\coord_system.coord_system_dimensionality <=
        parent_coord_system.coord_system_dimensionality;
END_ENTITY;

```

```

ENTITY coord_system_cylindrical
SUBTYPE OF (coord_system);
    origin : cylindrical_point;
    axes_definition : LIST [2:3] OF direction;
END_ENTITY;

```

```

ENTITY coord_system_spherical
SUBTYPE OF (coord_system);
    origin : spherical_point;
    axes_definition : LIST [3:3] OF direction;
END_ENTITY;

```

(* [Modified for LPM/6](#) *)

```

ENTITY coordinated_universal_time_offset;
    hour_offset : hour_in_day;
    minute_offset : OPTIONAL minute_in_hour;
    sense : ahead_or_behind;
DERIVE
    actual_minute_offset: INTEGER := NVL(minute_offset,0);
WHERE
    WRC23: { 0 <= hour_offset < 24 };
    WRC24: { 0 <= actual_minute_offset <= 59 };
    WRC25: NOT (((hour_offset <> 0) OR (actual_minute_offset <>0)) AND (sense = exact));
END_ENTITY; (* STEP Part 41 2nd edition *)

```

```

ENTITY csg_solid
SUBTYPE OF (solid_model);
    tree_root_expression : csg_select;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY currency_measure_with_unit
SUPERTYPE OF (ONEOF(currency_rate_with_unit));
    amount : REAL;
    unit : currency_unit;
END_ENTITY;

```

```

ENTITY currency_rate_with_unit
SUBTYPE OF (currency_measure_with_unit);

```



```

    per_quantity : measure_with_unit;
END_ENTITY;

```

```

ENTITY currency_unit;
    name : label;
    description : OPTIONAL text;
END_ENTITY;

```

(* [Modified for LPM/6](#) *)

```

ENTITY curve
SUPERTYPE OF (ONEOF
    (line,
    clothoid,
    conic,
    pcurve,
    surface_curve,
    offset_curve_2d,
    offset_curve_3d,
    curve_replica) ANDOR
    bounded_curve)
SUBTYPE OF (geometric_representation_item);
END_ENTITY; (* STEP Part 42 (expanded in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY curve_bounded_surface
SUBTYPE OF (bounded_surface);
    basis_surface : surface;
    boundaries : SET [1:?] OF boundary_curve;
    implicit_outer : BOOLEAN;
WHERE
    WR16: (NOT implicit_outer) OR
        (SIZEOF (QUERY (temp <* boundaries |
        'STRUCTURAL_FRAME_SCHEMA.OUTER_BOUNDARY_CURVE' IN
        TYPEOF(temp))) = 0);
    WR17: (NOT(implicit_outer)) OR
        ('STRUCTURAL_FRAME_SCHEMA.BOUNDED_SURFACE' IN
        TYPEOF(basis_surface));
    WR18: SIZEOF(QUERY(temp <* boundaries |
        'STRUCTURAL_FRAME_SCHEMA.OUTER_BOUNDARY_CURVE' IN
        TYPEOF(temp))) <= 1;
    WR19: SIZEOF(QUERY(temp <* boundaries |
        (temp\composite_curve_on_surface.basis_surface [1] <> SELF.basis_surface))) = 0;
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

ENTITY curve_replica
SUBTYPE OF (curve);
    parent_curve : curve;
    transformation : cartesian_transformation_operator;

```

WHERE

WRC20 : transformation.dim = parent_curve.dim;
 WRC21 : acyclic_curve_replica (SELF, parent_curve);
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY cut

SUBTYPE OF (structural_frame_process);
 cutting_method : cutting_type;
 END_ENTITY;

(* New for LPM/6 *)

ENTITY cyclide_segment_solid
 SUBTYPE OF (geometric_representation_item);
 position : axis2_placement_3d;
 radius1 : positive_length_measure;
 radius2 : positive_length_measure;
 cone_angle1 : plane_angle_measure;
 cone_angle2 : plane_angle_measure;
 turn_angle : plane_angle_measure;
 END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

(* Modified for LPM/6 - CIS ENTITY replaced by STEP entity *)

ENTITY cylindrical_point
 SUBTYPE OF (cartesian_point);
 r : length_measure;
 theta : plane_angle_measure;
 z : length_measure;
 DERIVE
 SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure :=
 [r*cos(theta), r*sin(theta), z];
 WHERE
 WRC26: r >= 0.0;
 END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

ENTITY cylindrical_surface
 SUBTYPE OF (elementary_surface);
 radius : positive_length_measure;
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

ENTITY cylindrical_volume
 SUBTYPE OF (volume);
 position : axis2_placement_3d;
 radius : positive_length_measure;
 height : positive_length_measure;
 END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

ENTITY date

SUPERTYPE OF (ONEOF(calendar_date));

year_component : year_number;

END_ENTITY; (* STEP Part 41 trimmed (2nd edition unchanged) *)

ENTITY date_and_time;

date_component : calendar_date;

time_component : local_time;

END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

ENTITY definitional_representation

SUBTYPE OF (representation);

WHERE

WRD1 : 'STRUCTURAL_FRAME_SCHEMA.

PARAMETRIC_REPRESENTATION_CONTEXT' IN TYPEOF(
SELF\representation.context_of_items);

END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

ENTITY degenerate_pcurve

SUBTYPE OF (point);

basis_surface : surface;

reference_to_curve : definitional_representation;

WHERE

WRD2 : SIZEOF(reference_to_curve\representation.items) = 1;

WRD3 : 'STRUCTURAL_FRAME_SCHEMA.CURVE' IN TYPEOF(
(reference_to_curve\representation.items[1]);

WRD4 : reference_to_curve\representation.

items[1]\geometric_representation_item.dim =2;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY degenerate_toroidal_surface

SUBTYPE OF (toroidal_surface);

select_outer : BOOLEAN;

WHERE

WRD5 : major_radius < minor_radius;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY derived_measure_with_unit

SUPERTYPE OF (ONEOF

(force_per_length_measure_with_unit,

inertia_measure_with_unit,

linear_acceleration_measure_with_unit,

linear_stiffness_measure_with_unit,

linear_velocity_measure_with_unit,

mass_per_length_measure_with_unit,

modulus_measure_with_unit,

```

        moment_measure_with_unit,
        rotational_acceleration_measure_with_unit,
        rotational_stiffness_measure_with_unit,
        rotational_velocity_measure_with_unit))
SUBTYPE OF (measure_with_unit);
WHERE
    WRD6 : 'STRUCTURAL_FRAME_SCHEMA.DERIVED_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
END_ENTITY;

```

(* Modified for LPM/6 *)

```

ENTITY derived_unit
SUPERTYPE OF (ONEOF
    (force_per_length_unit,
    inertia_unit,
    linear_acceleration_unit,
    linear_stiffness_unit,
    linear_velocity_unit,
    mass_per_length_unit,
    modulus_unit,
    moment_unit,
    rotational_acceleration_unit,
    rotational_stiffness_unit,
    rotational_velocity_unit));
    elements : SET [1:?] OF derived_unit_element;
DERIVE
    name : label := get_name_value (SELF);
WHERE
    WRD7 : ( SIZEOF ( elements ) > 1 ) OR
        (( SIZEOF ( elements ) = 1 ) AND ( elements[1].exponent <> 1.0 ));
    WRD13 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
        'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; (* expanded STEP Part 41 (modified in 2nd edition) *)

```

```

ENTITY derived_unit_element;
    unit : named_unit;
    exponent : REAL;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

(* New for LPM/6 *)

```

ENTITY description_attribute;
    attribute_value : text;
    described_item : description_attribute_select;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

```

ENTITY design_criterion
SUPERTYPE OF (design_criterion_documented);

```

```

    criterion_name : label;
    criterion_description : text;
    design_assumptions : OPTIONAL text;
INVERSE
    governed_assemblies : SET [1:?] OF assembly_design FOR governing_criteria;
END_ENTITY;

```

```

ENTITY design_criterion_documented
SUBTYPE OF (design_criterion);
    documented_reference : document_usage_constraint;
END_ENTITY;

```

```

ENTITY design_joint_system;
    design_joint_system_name : label;
    design_joint_system_spec : joint_system;
    parent_assemblies : LIST [1:?] OF assembly_design;
    locations : OPTIONAL LIST [1:?] OF coord_system;
    connected_parts : SET [0:?] OF design_part;
WHERE
    WRD8 : NOT(EXISTS(locations) AND (SIZEOF(locations) <>
        SIZEOF(parent_assemblies)));
END_ENTITY;

```

```

ENTITY design_part;
    design_part_name : label;
    design_part_spec : part;
    parent_assemblies : LIST [1:?] OF assembly_design;
    locations : OPTIONAL LIST [1:?] OF coord_system;
WHERE
    WRD9 : NOT(EXISTS(locations) AND (SIZEOF(locations) <>
        SIZEOF(parent_assemblies)));
END_ENTITY;

```

```

ENTITY design_result
ABSTRACT SUPERTYPE OF (ONEOF
    (design_result_connection,
    design_result_joint_system,
    design_result_member,
    design_result_part) ANDOR
    design_result_mapped ANDOR
    design_result_resolved);
    design_result_name : label;
    design_resistance : resistance;
END_ENTITY;

```

```

ENTITY design_result_connection
SUBTYPE OF (design_result);

```

```

    result_for_connection : assembly_design_structural_connection;
    result_position : OPTIONAL point;
    position_label : OPTIONAL label;
END_ENTITY;

```

```

ENTITY design_result_joint_system
SUBTYPE OF (design_result);
    result_for_joint_system : design_joint_system;
END_ENTITY;

```

```

ENTITY design_result_mapped
SUBTYPE OF (design_result);
    origin_of_forces : analysis_results_set;
END_ENTITY;

```

```

ENTITY design_result_member
SUBTYPE OF (design_result);
    result_for_member : assembly_design_structural_member;
    result_position : point;
    position_label : label;
END_ENTITY;

```

```

ENTITY design_result_part
SUBTYPE OF (design_result);
    result_for_part : design_part;
END_ENTITY;

```

```

ENTITY design_result_resolved
SUBTYPE OF (design_result);
    design_forces : reaction_force;
    design_factor : REAL;
END_ENTITY;

```

```

ENTITY dimensional_exponents;
    length_exponent : REAL;
    mass_exponent : REAL;
    time_exponent : REAL;
    electric_current_exponent : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent : REAL;
    luminous_intensity_exponent : REAL;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY directed_action
SUBTYPE OF (executed_action);
    directive : action_directive;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY direction
SUBTYPE OF (geometric_representation_item);
    direction_ratios : LIST [2:3] OF REAL;
WHERE
    WRD10 : SIZEOF(QUERY(tmp <* direction_ratios | tmp <> 0.0)) > 0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) - See Issue 15 *)

```

ENTITY dispatch
SUBTYPE OF (structural_frame_process);
    transported_products : LIST [1:?] OF product_item_select;
    dispatch_address : organizational_address;
    delivery_address : organizational_address;
    dispatch_date : calendar_date;
    delivery_date : calendar_date;
END_ENTITY;

```

(* [Modified for LPM/6](#) *)

```

ENTITY document
SUPERTYPE OF (document_with_class);
    id : identifier;
    name : label;
    description : text;
    kind : document_type;
INVERSE
    representation_types : SET[0:?] OF document_representation_type
        FOR represented_document;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY document_relationship
SUPERTYPE OF (document_usage);
    name : label;
    description : OPTIONAL text;
    relating_document : document;
    related_document : document;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

(* [New for LPM/6](#) *)

```

ENTITY document_representation_type;
    name : label;
    represented_document : document;
END_ENTITY; (* STEP Part 41 (added in 2nd edition) *)

```

```

ENTITY document_standard
SUBTYPE OF (document_with_class);

```

DERIVE

clauses : SET [1:?] OF document_usage_constraint := bag_to_set (USEDIN(SELF,
'STRUCTURAL_FRAME_SCHEMA.DOCUMENT_USAGE_CONSTRAINT.SOURCE')
);

INVERSE

relevant_clauses : SET [1:?] OF document_usage_constraint FOR source;

WHERE

WRD11 : ((SELF\document.kind.product_data_type = 'Standard Specification') OR
(SELF\document.kind.product_data_type = 'Code of Practice'));

END_ENTITY;

ENTITY document_type;

product_data_type : label;

END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

ENTITY document_usage

SUBTYPE OF (document_relationship);

UNIQUE

URD2 : SELF\document_relationship.name,
SELF\document_relationship.relating_document,
SELF\document_relationship.related_document;

WHERE

WRD12 : acyclic_document_relationship(SELF,
[SELF\document_relationship.related_document],
'STRUCTURAL_FRAME_SCHEMA.DOCUMENT_RELATIONSHIP.
RELATED_DOCUMENT');

END_ENTITY;

ENTITY document_usage_constraint;

source : document;

subject_element : label;

subject_element_value : text;

END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

ENTITY document_with_class

SUPERTYPE OF (document_standard)

SUBTYPE OF (document);

class : identifier;

END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

(* New for LPM/6 *)

ENTITY eccentric_cone

SUBTYPE OF (geometric_representation_item);

position : axis2_placement_3d;

semi_axis_1 : positive_length_measure;

semi_axis_2 : positive_length_measure;

height : positive_length_measure;


```

    x_offset : length_measure;
    y_offset : length_measure;
    ratio : REAL;
WHERE
    WRE23 : ratio >= 0.0;
END_ENTITY; (* STEP Part 42 (New for 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY eccentric_conical_volume
SUBTYPE OF (volume);
    position : axis2_placement_3d;
    semi_axis_1 : positive_length_measure;
    semi_axis_2 : positive_length_measure;
    height : positive_length_measure;
    x_offset : length_measure;
    y_offset : length_measure;
    ratio : REAL;
WHERE
    WRE22 : ratio >= 0.0;
END_ENTITY; (* STEP Part 42 (New for 2nd edition) *)

```

(* Modified for LPM/6 *)

```

ENTITY edge
SUPERTYPE OF (ONEOF(oriented_edge, edge_curve, subedge))
SUBTYPE OF (topological_representation_item);
    edge_start : vertex;
    edge_end : vertex;
END_ENTITY; (* STEP Part 42 (modified for 2nd edition) *)

```

```

ENTITY edge_based_wireframe_model
SUBTYPE OF (geometric_representation_item);
    ebwm_boundary : SET [1:?] OF connected_edge_set;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY edge_curve
SUBTYPE OF (edge, geometric_representation_item);
    edge_geometry : curve;
    same_sense: BOOLEAN;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY edge_loop
SUBTYPE OF (loop, path);
DERIVE
    ne : INTEGER := SIZEOF(SELF\path.edge_list);
WHERE
    WRE1 : (SELF\path.edge_list[1].edge_start) :=:
        (SELF\path.edge_list[ne].edge_end);

```

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY effective_buckling_length;
 effective_length_name : label;
 effective_length_factor : OPTIONAL REAL;
 effective_length_use : OPTIONAL text;
 effective_length_direction : OPTIONAL buckling_direction;
 applicable_member : assembly_design_structural_member;
 END_ENTITY;

ENTITY element
 SUPERTYPE OF (ONEOF
 (element_volume,
 element_surface,
 element_curve,
 element_point) ANDOR
 element_with_material);
 element_name : label;
 element_description : OPTIONAL text;
 parent_model : analysis_model;
 element_dimensionality : INTEGER;
 INVERSE
 connectivity : SET [1:?] OF element_node_connectivity FOR connecting_element;
 UNIQUE
 URE1 : element_name, parent_model;
 WHERE
 WRE2 : element_dimensionality <= parent_model.coordinate_space_dimension;
 WRE21 : (element_dimensionality >= 0) AND (element_dimensionality <= 3);
 END_ENTITY;

ENTITY element_curve
 ABSTRACT SUPERTYPE OF (ONEOF(element_curve_simple, element_curve_complex))
 SUBTYPE OF (element);
 element_subdivision : OPTIONAL INTEGER;
 DERIVE
 connectivities : SET [2:2] OF element_node_connectivity := bag_to_set (USEDIN(SELF,
 'STRUCTURAL_FRAME_SCHEMA.ELEMENT_NODE_CONNECTIVITY.
 CONNECTING_ELEMENT'));
 WHERE
 WRE3 : SELF\element.element_dimensionality = 1;
 WRE4 : connectivities[1] :<>: connectivities[2];
 WRE5 : connectivities[1].connecting_node :<>: connectivities[2].connecting_node;
 END_ENTITY;

ENTITY element_curve_complex
 SUBTYPE OF (element_curve);
 cross_sections : LIST [2:?] OF section_profile;

```

    points_defining_element_axis : LIST [2:?] OF point_on_curve;
    element_orientations : LIST [2:?] OF orientation_select;
DERIVE
    number_of_sections : INTEGER := SIZEOF (cross_sections);
    curve_defining_element : curve :=
        points_defining_element_axis[1]\point_on_curve.basis_curve;
WHERE
    WRE6 : ( (SIZEOF (points_defining_element_axis) = number_of_sections) AND
        (SIZEOF (element_orientations) = number_of_sections) );
    WRE7 : SIZEOF(QUERY(temp <* points_defining_element_axis |
        (temp\point_on_curve.basis_curve) :<>: curve_defining_element)) = 0;
END_ENTITY;

ENTITY element_curve_simple
SUBTYPE OF (element_curve);
    cross_section : section_profile;
    element_orientation : orientation_select;
END_ENTITY;

ENTITY element_eccentricity;
    element_eccentricity_name : label;
    x_eccentricity : OPTIONAL length_measure_with_unit;
    y_eccentricity : OPTIONAL length_measure_with_unit;
    z_eccentricity : OPTIONAL length_measure_with_unit;
INVERSE
    eccentric_connectivities: SET [1:?] OF element_node_connectivity FOR eccentricity;
WHERE
    WRE8 : EXISTS (x_eccentricity) OR
        EXISTS (y_eccentricity) OR
        EXISTS (z_eccentricity);
END_ENTITY;

ENTITY element_mapping;
    mapped_element : element;
    represented_part : part_select;
END_ENTITY;

ENTITY element_node_connectivity;
    connectivity_number : INTEGER;
    connectivity_name : label;
    connecting_node : node;
    connecting_element : element;
    eccentricity : OPTIONAL element_eccentricity;
    fixity : OPTIONAL release;
UNIQUE
    URE2 : connecting_node, connecting_element;
WHERE

```

```

WRE9 : NOT( (connectivity_number > 2) AND
  (connecting_element.element_dimensionality < 2) );
WRE10 : NOT( (connectivity_name <> 'Start Node') AND (connectivity_number = 1) );
WRE11 : NOT( (connectivity_name <> 'End Node') AND (connectivity_number = 2) AND
  (connecting_element.element_dimensionality = 1) );
WRE12 : connecting_node.parent_model :=: connecting_element.parent_model;
END_ENTITY;

```

```

ENTITY element_point
ABSTRACT SUPERTYPE OF (ONEOF
  (element_point_grounded_damper,
   element_point_stationary_mass,
   element_point_grounded_spring))
SUBTYPE OF (element);
DERIVE
  connectivities : BAG [1:1] OF element_node_connectivity := USEDIN(SELF,
    STRUCTURAL_FRAME_SCHEMA.ELEMENT_NODE_CONNECTIVITY.
    CONNECTING_ELEMENT');
WHERE
  WRE13 : SELF\element.element_dimensionality = 0;
END_ENTITY;

```

```

ENTITY element_point_grounded_damper
SUBTYPE OF (element_point);
  damping_coefficients : ARRAY [1:6] OF REAL;
END_ENTITY;

```

```

ENTITY element_point_grounded_spring
SUBTYPE OF (element_point);
  stiffness_coefficients : ARRAY [1:6] OF REAL;
END_ENTITY;

```

```

ENTITY element_point_stationary_mass
SUBTYPE OF (element_point);
  masses : ARRAY [1:3] OF REAL;
  moments_of_inertia : ARRAY [1:3] OF ARRAY [1:3] OF REAL;
END_ENTITY;

```

```

ENTITY element_surface
SUPERTYPE OF (ONEOF(element_surface_simple, element_surface_complex))
SUBTYPE OF (element);
  thickness : positive_length_measure_with_unit;
DERIVE
  connectivities : SET [3:?] OF element_node_connectivity := bag_to_set
    (USEDIN(SELF,STRUCTURAL_FRAME_SCHEMA.
      ELEMENT_NODE_CONNECTIVITY.CONNECTING_ELEMENT'));
WHERE
  WRE14 : SELF\element.element_dimensionality = 2;

```

```
WRE15 : SELF\element.parent_model.coordinate_space_dimension > 1;
END_ENTITY;
```

```
ENTITY element_surface_complex
SUPERTYPE OF (ONEOF(element_surface_plane, element_surface_profiled))
SUBTYPE OF (element_surface);
    surface_definition : surface;
END_ENTITY;
```

```
ENTITY element_surface_plane
SUBTYPE OF (element_surface_complex);
WHERE
    WRE16 : 'STRUCTURAL_FRAME_SCHEMA.PLANE' IN
        TYPEOF (SELF\element_surface_complex.surface_definition);
END_ENTITY;
```

```
ENTITY element_surface_profiled
SUBTYPE OF (element_surface_complex);
    profile : curve;
END_ENTITY;
```

```
ENTITY element_surface_simple
SUBTYPE OF (element_surface);
    shape : element_surface_shape;
    assumption : plane_stress_or_strain;
END_ENTITY;
```

```
ENTITY element_volume
ABSTRACT SUPERTYPE OF (ONEOF(element_volume_simple,
    element_volume_complex))
SUBTYPE OF (element);
DERIVE
    connectivities : SET [4:?] OF element_node_connectivity := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.ELEMENT_NODE_CONNECTIVITY.
        CONNECTING_ELEMENT'));
WHERE
    WRE17 : SELF\element.element_dimensionality = 3;
    WRE18 : SELF\element.parent_model.coordinate_space_dimension = 3;
END_ENTITY;
```

```
ENTITY element_volume_complex
SUBTYPE OF (element_volume);
    shape : shape_representation_with_units;
END_ENTITY;
```

```
ENTITY element_volume_simple
SUBTYPE OF (element_volume);
    shape : element_volume_shape;
```

END_ENTITY;

ENTITY element_with_material

SUBTYPE OF (element);

material_definition : material;

END_ENTITY;

ENTITY elementary_surface

SUPERTYPE OF (ONEOF

(plane,
cylindrical_surface,
conical_surface,
spherical_surface,
toroidal_surface))

SUBTYPE OF (surface);

position : axis2_placement_3d;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY ellipse

SUBTYPE OF (conic);

semi_axis_1 : positive_length_measure;

semi_axis_2 : positive_length_measure;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

ENTITY ellipsoid

SUBTYPE OF (geometric_representation_item);

position : axis2_placement_3d;

semi_axis_1 : positive_length_measure;

semi_axis_2 : positive_length_measure;

semi_axis_3 : positive_length_measure;

END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

(* New for LPM/6 *)

ENTITY ellipsoid_volume

SUBTYPE OF (volume);

position : axis2_placement_3d;

semi_axis_1 : positive_length_measure;

semi_axis_2 : positive_length_measure;

semi_axis_3 : positive_length_measure;

END_ENTITY;

(* New for LPM/6 *)

ENTITY elliptic_area

SUBTYPE OF (primitive_2d);

position : axis2_placement_2d;

semi_axis_1 : positive_length_measure;

```

    semi_axis_2 : positive_length_measure;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY evaluated_degenerate_pcurve
SUBTYPE OF (degenerate_pcurve);
    equivalent_point : cartesian_point;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY executed_action
SUBTYPE OF (action);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY extruded_area_solid
SUBTYPE OF (swept_area_solid);
    extruded_direction : direction;
    depth : positive_length_measure;
WHERE
    WRE19 : dot_product(
        (SELF\swept_area_solid.swept_area.basis_surface\
        elementary_surface.position.p[3]), extruded_direction) <> 0.0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY extruded_face_solid
SUBTYPE OF (swept_face_solid);
    extruded_direction : direction;
    depth : positive_length_measure;
WHERE
    WRE20 : dot_product(
        (SELF\swept_face_solid.swept_face.face_geometry\
        elementary_surface.position.p[3]), extruded_direction) <> 0.0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY face
SUPERTYPE OF (ONEOF(face_surface, oriented_face, subface))
SUBTYPE OF (topological_representation_item);
    bounds : SET [1:?] OF face_bound;
WHERE
    WRF1 : NOT (mixed_loop_type_set(list_to_set(list_face_loops(SELF))));
    WRF2 : SIZEOF(QUERY(temp <* bounds | 'STRUCTURAL_FRAME_SCHEMA.
        FACE_OUTER_BOUND' IN TYPEOF(temp))) <= 1;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY face_based_surface_model
SUBTYPE OF (geometric_representation_item);
    fbasm_faces : SET [1:?] OF connected_face_set;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```
ENTITY face_bound
SUBTYPE OF (topological_representation_item);
    bound : loop;
    orientation : BOOLEAN;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```
ENTITY face_outer_bound
SUBTYPE OF (face_bound);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

(* **Modified for LPM/6** *)

```
ENTITY face_surface
SUBTYPE OF (face, geometric_representation_item);
    face_geometry : surface;
    same_sense: BOOLEAN;
WHERE
    WRF30 : NOT ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_SURFACE' IN
        TYPEOF(face_geometry));
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)
```

```
ENTITY faceted_brep
SUBTYPE OF (manifold_solid_brep);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

(* **New for LPM/6** *)

```
ENTITY faceted_primitive
SUPERTYPE OF (ONEOF(tetrahedron, convex_hexahedron))
SUBTYPE OF (geometric_representation_item) ;
    points : LIST[4:?] OF UNIQUE cartesian_point ;
WHERE
    WRF31 : points[1].dim = 3 ;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)
```

```
ENTITY fastener
SUPERTYPE OF (ONEOF(fastener_simple, fastener_complex))
SUBTYPE OF (structural_frame_product);
    fastener_grade : OPTIONAL label;
END_ENTITY;
```

```
ENTITY fastener_complex
SUBTYPE OF (fastener);
    fastener_shape : shape_representation_with_units;
END_ENTITY;
```

```
ENTITY fastener_mechanism
SUBTYPE OF (structural_frame_product);
```



```
sequence : OPTIONAL text;
fasteners : LIST [1:?] OF fastener;
END_ENTITY;
```

(* New for LPM/6 - see Issue 85 *)

```
ENTITY fastener_mechanism_with_position
SUBTYPE OF (fastener_mechanism);
    fastener_positions : LIST [1:?] OF length_measure_with_unit;
WHERE
    WRF32 : SIZEOF(fastener_positions) = SIZEOF(SELF\fastener_mechanism.fasteners);
END_ENTITY;
```

```
ENTITY fastener_simple
SUPERTYPE OF (ONEOF
    (fastener_simple_bolt,
    fastener_simple_nut,
    fastener_simple_washer,
    fastener_simple_stud,
    fastener_simple_nail,
    fastener_simple_pin,
    fastener_simple_screw,
    fastener_simple_shear_connector) ANDOR
    fastener_simple_countersunk ANDOR
    fastener_simple_curved)
SUBTYPE OF (fastener);
    nominal_diameter : positive_length_measure_with_unit;
    nominal_length : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;
```

(* Modified for LPM/6 *)

```
ENTITY fastener_simple_bolt
SUPERTYPE OF (ONEOF
    (fastener_simple_bolt_circular_head,
    fastener_simple_bolt_hexagonal_head,
    fastener_simple_bolt_square_head))
SUBTYPE OF (fastener_simple);
    length_of_shank : OPTIONAL positive_length_measure_with_unit;
    bolt_preload : OPTIONAL force_measure_with_unit;
    full_section_area : OPTIONAL area_measure_with_unit;
    reduced_section_area : OPTIONAL area_measure_with_unit;
DERIVE
    bolt_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;
END_ENTITY;
```

```
ENTITY fastener_simple_bolt_circular_head
SUBTYPE OF (fastener_simple_bolt);
    bolt_head_height : positive_length_measure_with_unit;
```

```

    head_diameter : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_bolt_hexagonal_head
SUBTYPE OF (fastener_simple_bolt);
    bolt_head_height : positive_length_measure_with_unit;
    distance_across_vertices : OPTIONAL positive_length_measure_with_unit;
    distance_across_flats : OPTIONAL positive_length_measure_with_unit;
WHERE
    WRF3 : EXISTS (distance_across_vertices) OR EXISTS (distance_across_flats);
    WRF4 : NOT( (distance_across_flats.value_component >
        distance_across_vertices.value_component) AND
        (EXISTS (distance_across_vertices) AND EXISTS (distance_across_flats)) );
END_ENTITY;

```

```

ENTITY fastener_simple_bolt_square_head
SUBTYPE OF (fastener_simple_bolt);
    bolt_head_height : positive_length_measure_with_unit;
    distance_across_flats : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_countersunk
SUBTYPE OF (fastener_simple);
    countersink_angle : plane_angle_measure_with_unit;
    countersink_depth : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_curved
SUBTYPE OF (fastener_simple);
    curve_definition : curve;
END_ENTITY;

```

```

ENTITY fastener_simple_nail
SUBTYPE OF (fastener_simple);
    nail_type : OPTIONAL text;
    nail_drive_type : OPTIONAL text;
    nail_head_shape : OPTIONAL text;
    nail_point_type : OPTIONAL text;
END_ENTITY;

```

(* [Modified for LPM/6](#) *)

```

ENTITY fastener_simple_nut
SUPERTYPE OF (ONEOF
    (fastener_simple_nut_circular,
    fastener_simple_nut_hexagonal,
    fastener_simple_nut_square) ANDOR
    fastener_simple_nut_closed)

```

```

SUBTYPE OF (fastener_simple);
DERIVE
    nut_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;
WHERE
    WRF28 : NOT('STRUCTURAL_FRAME_SCHEMA.FASTENER_SIMPLE_CURVED' IN
        TYPEOF(SELF));
END_ENTITY;

```

```

ENTITY fastener_simple_nut_circular
SUBTYPE OF (fastener_simple_nut);
    outside_diameter : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_nut_closed
SUBTYPE OF (fastener_simple_nut);
    nut_depth : positive_length_measure_with_unit;
WHERE
    WRF5 : SELF\fastener_simple.nominal_length.value_component >
        nut_depth.value_component;
END_ENTITY;

```

```

ENTITY fastener_simple_nut_hexagonal
SUBTYPE OF (fastener_simple_nut);
    distance_across_vertices : OPTIONAL positive_length_measure_with_unit;
    distance_across_flats : OPTIONAL positive_length_measure_with_unit;
WHERE
    WRF6 : EXISTS (distance_across_vertices) OR EXISTS (distance_across_flats);
    WRF7 : NOT( (distance_across_flats.value_component >
        distance_across_vertices.value_component) AND
        (EXISTS (distance_across_vertices) AND EXISTS (distance_across_flats)) );
END_ENTITY;

```

```

ENTITY fastener_simple_nut_square
SUBTYPE OF (fastener_simple_nut);
    distance_across_flats : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_pin
SUBTYPE OF (fastener_simple);
    pin_type : OPTIONAL text;
END_ENTITY;

```

```

ENTITY fastener_simple_screw
SUPERTYPE OF (ONEOF
    (fastener_simple_screw_machine,
    fastener_simple_screw_tapered) ANDOR
    fastener_simple_screw_self_drilling ANDOR

```

```

        fastener_simple_screw_self_tapping)
SUBTYPE OF (fastener_simple);
    screw_type : OPTIONAL text;
    screw_drive_type : OPTIONAL text;
    screw_point_type : OPTIONAL text;
    screw_head_height : OPTIONAL positive_length_measure_with_unit;
    full_section_area : OPTIONAL area_measure_with_unit;
    reduced_section_area : OPTIONAL area_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_screw_machine
SUBTYPE OF (fastener_simple_screw);
WHERE
    WRF29 : NOT ('STRUCTURAL_FRAME_SCHEMA.FASTENER_SIMPLE_CURVED' IN
        TYPEOF(SELF));
END_ENTITY;

```

```

ENTITY fastener_simple_screw_self_drilling
SUBTYPE OF (fastener_simple_screw);
    hole_cutting_method : cutting_type;
    pilot_hole_diameter : OPTIONAL positive_length_measure_with_unit;
    drill_diameter : OPTIONAL positive_length_measure_with_unit;
WHERE
    WRF8 : NOT( (pilot_hole_diameter.value_component > drill_diameter.value_component)
        AND (EXISTS (pilot_hole_diameter) AND EXISTS (drill_diameter)) );
END_ENTITY;

```

```

ENTITY fastener_simple_screw_self_tapping
SUBTYPE OF (fastener_simple_screw);
    thread_cutting_method : cutting_type;
    pilot_hole_diameter : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_screw_tapered
SUBTYPE OF (fastener_simple_screw);
    absolute_taper : OPTIONAL length_measure_with_unit;
    relative_taper : OPTIONAL ratio_measure_with_unit;
WHERE
    WRF9 : EXISTS (absolute_taper) OR EXISTS (relative_taper);
END_ENTITY;

```

```

ENTITY fastener_simple_shear_connector
SUBTYPE OF (fastener_simple);
    head_shape : OPTIONAL text;
    connector_type : OPTIONAL text;
    connection_method : OPTIONAL text;
END_ENTITY;

```

(* Modified for LPM/6 - see Issue 14 *)

```

ENTITY fastener_simple_stud
SUBTYPE OF (fastener_simple);
    thread_length_1 : positive_length_measure_with_unit;
    thread_length_2 : OPTIONAL positive_length_measure_with_unit;
    length_of_shank : OPTIONAL positive_length_measure_with_unit;
    full_section_area : OPTIONAL area_measure_with_unit;
    reduced_section_area : OPTIONAL area_measure_with_unit;
DERIVE
    thread_length_value_1 : REAL := thread_length_1.value_component;
    thread_length_value_2 : REAL := NVL(thread_length_2.value_component, 0.0);
    length_of_shank_value : REAL := NVL(length_of_shank.value_component, 0.0);
WHERE
    WRF33 : NOT (EXISTS(SELF\fastener_simple.nominal_length) AND (
        (thread_length_value_1 + thread_length_value_2 + length_of_shank_value) >
        (SELF\fastener_simple.nominal_length.value_component) ) );
END_ENTITY;

```

(* Modified for LPM/6 *)

```

ENTITY fastener_simple_washer
SUPERTYPE OF (ONEOF
    (fastener_simple_washer_tapered, fastener_simple_washer_load_indicating))
SUBTYPE OF (fastener_simple);
    washer_shape : OPTIONAL text;
    inside_diameter : OPTIONAL positive_length_measure_with_unit;
    external_dimension : OPTIONAL positive_length_measure_with_unit;
DERIVE
    washer_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;
WHERE
    WRF10 : NOT( (EXISTS (inside_diameter) AND EXISTS (external_dimension)) AND
        (inside_diameter.value_component > external_dimension.value_component) );
    WRF11 : NOT( (EXISTS (inside_diameter)) AND
        (inside_diameter.value_component <
        (SELF\fastener_simple.nominal_diameter.value_component)) );
END_ENTITY;

```

```

ENTITY fastener_simple_washer_load_indicating
SUBTYPE OF (fastener_simple_washer);
    final_gap : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY fastener_simple_washer_tapered
SUBTYPE OF (fastener_simple_washer);
    taper : ratio_measure_with_unit;
END_ENTITY;

```

(* Modified for LPM/6 *)

ENTITY feature

SUPERTYPE OF (ONEOF

(feature_cutting_plane,
feature_edge_chamfer,
feature_surface,
feature_thread,
feature_volume))

SUBTYPE OF (structural_frame_item);

DERIVE

uses : SET [0:?] OF located_feature := bag_to_set (USEDIN(SELF,
'STRUCTURAL_FRAME_SCHEMA.LOCATED_FEATURE.'
DESCRIPTIVE_FEATURE'));

END_ENTITY;

ENTITY feature_cutting_plane

SUBTYPE OF (feature);

plane_definition : plane;

END_ENTITY;

ENTITY feature_edge_chamfer

ABSTRACT SUPERTYPE OF (ONEOF

(feature_edge_chamfer_straight,
feature_edge_chamfer_fillet,
feature_edge_chamfer_rounding))

SUBTYPE OF (feature);

follow_round : BOOLEAN;

END_ENTITY;

ENTITY feature_edge_chamfer_fillet

SUBTYPE OF (feature_edge_chamfer);

edge_fillet_radius : positive_length_measure_with_unit;

END_ENTITY;

ENTITY feature_edge_chamfer_rounding

SUBTYPE OF (feature_edge_chamfer);

edge_rounding_radius : positive_length_measure_with_unit;

END_ENTITY;

ENTITY feature_edge_chamfer_straight

SUBTYPE OF (feature_edge_chamfer);

edge_chamfer_width : positive_length_measure_with_unit;

edge_chamfer_depth : positive_length_measure_with_unit;

END_ENTITY;

(* Modified for LPM/6 *)

```
ENTITY feature_surface
ABSTRACT SUPERTYPE OF (ONEOF
    (feature_surface_complex,
    feature_surface_point,
    feature_surface_simple) ANDOR
    feature_surface_name_tag ANDOR
    feature_surface_treatment ANDOR
    feature_surface_with_layout)
SUBTYPE OF (feature);
END_ENTITY;
```

```
ENTITY feature_surface_complex
SUBTYPE OF (feature_surface);
    feature_boundary : bounded_surface;
END_ENTITY;
```

```
ENTITY feature_surface_name_tag
SUBTYPE OF (feature_surface);
    name_tag_items : LIST [1:?] OF text;
END_ENTITY;
```

(* New for LPM/6 *)

```
ENTITY feature_surface_point
SUBTYPE OF (feature_surface);
    feature_point : point;
END_ENTITY;
```

(* New for LPM/6 *)

```
ENTITY feature_surface_point_mark
SUBTYPE OF (feature_surface);
    marking_process : surface_treatment_hard_stamp;
END_ENTITY;
```

```
ENTITY feature_surface_simple
SUBTYPE OF (feature_surface);
    feature_boundary : LIST [3:?] OF point;
END_ENTITY;
```

```
ENTITY feature_surface_treatment
SUBTYPE OF (feature_surface);
    treatment_definition : surface_treatment;
END_ENTITY;
```

```
ENTITY feature_surface_with_layout
SUBTYPE OF (feature_surface);
    layout : SET [2:?] OF point;
```

END_ENTITY;

ENTITY feature_thread

SUBTYPE OF (feature);

thread_pitch : positive_length_measure_with_unit;

thread_length : positive_length_measure_with_unit;

thread_profile : OPTIONAL shape_representation_with_units;

right_handed : BOOLEAN;

number_of_threads : INTEGER;

WHERE

WRF12 : number_of_threads > 0;

END_ENTITY;

(* [Modified for LPM/6](#) - see Issue 84 *)

ENTITY feature_volume

ABSTRACT SUPERTYPE OF (ONEOF (feature_volume_complex,

feature_volume_hole,

feature_volume_prismatic) ANDOR

feature_volume_curved ANDOR

feature_volume_with_depth ANDOR

feature_volume_with_limit ANDOR

feature_volume_with_layout ANDOR

feature_volume_with_process)

SUBTYPE OF (feature);

END_ENTITY;

ENTITY feature_volume_complex

SUBTYPE OF (feature_volume);

feature_shape : shape_representation_with_units;

END_ENTITY;

ENTITY feature_volume_curved

SUBTYPE OF (feature_volume);

feature_trace : curve;

END_ENTITY;

ENTITY feature_volume_curved_line

SUBTYPE OF (feature_volume_curved);

WHERE

WRF13 : 'STRUCTURAL_FRAME_SCHEMA.LINE' IN TYPEOF

(SELF.feature_volume_curved.feature_trace);

END_ENTITY;

ENTITY feature_volume_hole

ABSTRACT SUPERTYPE OF (ONEOF

(feature_volume_hole_circular,

feature_volume_hole_rectangular,


```

        feature_volume_hole_slotted))
SUBTYPE OF (feature_volume);
END_ENTITY;

```

```

ENTITY feature_volume_hole_circular
SUPERTYPE OF (feature_volume_hole_circular_threaded)
SUBTYPE OF (feature_volume_hole);
    hole_radius : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY feature_volume_hole_circular_threaded
SUBTYPE OF (feature_volume_hole_circular);
    thread_definition : feature_thread;
END_ENTITY;

```

```

ENTITY feature_volume_hole_rectangular
SUBTYPE OF (feature_volume_hole);
    hole_length : positive_length_measure_with_unit;
    hole_height : positive_length_measure_with_unit;
    fillet_radius : OPTIONAL positive_length_measure_with_unit;
DERIVE
    fillet_radius_value : REAL := NVL(fillet_radius.value_component, 0.0);
    hole_length_value : REAL := hole_length.value_component;
    hole_height_value : REAL := hole_height.value_component;
WHERE
    WRF14 : fillet_radius_value < (hole_length_value/2);
    WRF15 : fillet_radius_value < (hole_height_value/2);
END_ENTITY;

```

```

ENTITY feature_volume_hole_slotted
SUPERTYPE OF (feature_volume_hole_slotted_curved)
SUBTYPE OF (feature_volume_hole);
    slot_height : positive_length_measure_with_unit;
    slot_length : positive_length_measure_with_unit;
WHERE
    WRF16 : slot_length.value_component > slot_height.value_component;
END_ENTITY;

```

```

ENTITY feature_volume_hole_slotted_curved
SUBTYPE OF (feature_volume_hole_slotted);
    curve_radius : positive_length_measure_with_unit;
    sector_angle : plane_angle_measure_with_unit;
DERIVE
    slot_radius : REAL :=
        (SELF.feature_volume_hole_slotted.slot_height.value_component)/2.0;
END_ENTITY;

```

```

ENTITY feature_volume_prismatic
ABSTRACT SUPERTYPE OF ( ONEOF
    (feature_volume_prismatic_chamfer,
    feature_volume_prismatic_flange_notch,
    feature_volume_prismatic_flange_chamfer,
    feature_volume_prismatic_notch,
    feature_volume_prismatic_skewed_end))

```

```

SUBTYPE OF (feature_volume);
    top_or_bottom_edge : top_or_bottom;
    start_or_end : start_or_end_face;
    original_face : BOOLEAN;
END_ENTITY;

```

```

ENTITY feature_volume_prismatic_chamfer
SUBTYPE OF (feature_volume_prismatic);
    chamfer_length : positive_length_measure_with_unit;
    chamfer_depth : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY feature_volume_prismatic_flange_chamfer
SUBTYPE OF (feature_volume_prismatic);
    left_or_right_hand : left_or_right;
    flange_chamfer_length : positive_length_measure_with_unit;
    flange_chamfer_width : positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY feature_volume_prismatic_flange_notch
SUBTYPE OF (feature_volume_prismatic);
    left_or_right_hand : left_or_right;
    flange_notch_length : positive_length_measure_with_unit;
    flange_notch_width : positive_length_measure_with_unit;
    flange_notch_radius : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY feature_volume_prismatic_notch
SUBTYPE OF (feature_volume_prismatic);
    notch_length : positive_length_measure_with_unit;
    notch_depth : positive_length_measure_with_unit;
    notch_radius : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY feature_volume_prismatic_skewed_end
SUBTYPE OF (feature_volume_prismatic);
    skew_angle_1 : plane_angle_measure_with_unit;
    skew_angle_2 : plane_angle_measure_with_unit;
END_ENTITY;

```

(* New for LPM/6 - see Issue 84 *)

ENTITY feature_volume_with_depth

SUBTYPE OF (feature_volume);

penetration_depth : positive_length_measure_with_unit;

WHERE

WRF34 : NOT('STRUCTURAL_FRAME_SCHEMA.FEATURE_VOLUME_COMPLEX' IN
TYPEOF(SELF));

END_ENTITY;

ENTITY feature_volume_with_layout

SUBTYPE OF (feature_volume);

layout : SET [2:?] OF point;

END_ENTITY;

(* New for LPM/6 - see Issue 84 *)

ENTITY feature_volume_with_limit

SUBTYPE OF (feature_volume);

penetration_limit : INTEGER;

WHERE

WRF35 : NOT('STRUCTURAL_FRAME_SCHEMA.FEATURE_VOLUME_COMPLEX' IN
TYPEOF(SELF));

WRF36 : penetration_limit > 0;

END_ENTITY;

ENTITY feature_volume_with_process

SUBTYPE OF (feature_volume);

process_definition : cut;

END_ENTITY;

(* New for LPM/6 *)

ENTITY fixed_reference_swept_surface

SUBTYPE OF (swept_surface);

directrix : curve;

fixed_reference : direction;

END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

ENTITY flavour

SUBTYPE OF (group_assignment);

items : SET [2:?] OF item_reference;

END_ENTITY;

ENTITY force_measure_with_unit

SUBTYPE OF (measure_with_unit);

WHERE

WRF17 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN
TYPEOF (SELF\measure_with_unit.unit_component);

WRF18 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_MEASURE' IN

```

        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```

```

ENTITY force_per_length_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
    WRF19 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_PER_LENGTH_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
    WRF20 : 'STRUCTURAL_FRAME_SCHEMA.FORCE_PER_LENGTH_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```

```

ENTITY force_per_length_unit
SUBTYPE OF (derived_unit);
WHERE
    WRF21 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRF22 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRF23 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);
END_ENTITY;

```

```

ENTITY force_unit
SUBTYPE OF (named_unit);
WHERE
    WRF24 : (SELF\named_unit.dimensions.length_exponent = 1.0) AND
        (SELF\named_unit.dimensions.mass_exponent = 1.0) AND
        (SELF\named_unit.dimensions.time_exponent = -2.0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY founded_item;
END_ENTITY; (* STEP Part 43 2nd Edition *)

```

```

ENTITY frequency_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRF25 : 'STRUCTURAL_FRAME_SCHEMA.FREQUENCY_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
    WRF26 : 'STRUCTURAL_FRAME_SCHEMA.FREQUENCY_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);

```

END_ENTITY;

ENTITY frequency_unit

SUBTYPE OF (named_unit);

WHERE

WRF27 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND
 (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
 (SELF\named_unit.dimensions.time_exponent = -1.0) AND
 (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
 (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
 (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
 (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);

END_ENTITY;

ENTITY functional_role

SUPERTYPE OF (functional_role_documented);

functional_role_name : label;

functional_role_description : text;

INVERSE

role_for_assemblies : SET [1:?] OF assembly_design FOR roles;

END_ENTITY;

ENTITY functional_role_documented

SUBTYPE OF (functional_role);

document_reference : document_usage_constraint;

END_ENTITY;

ENTITY functionally_defined_transformation;

name : label;

description : text;

END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

ENTITY geographical_location

ABSTRACT SUPERTYPE OF (ONEOF(global_location, map_location));

height_above_datum : length_measure_with_unit;

datum_name : label;

END_ENTITY;

ENTITY geometric_curve_set

SUBTYPE OF (geometric_set);

WHERE

WRG1 : SIZEOF(QUERY(temp <* SELF\geometric_set.elements |
 'STRUCTURAL_FRAME_SCHEMA.SURFACE' IN TYPEOF(temp))) = 0;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

ENTITY geometric_representation_context
SUBTYPE OF (representation_context);
    coordinate_space_dimension : dimension_count;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY geometric_representation_item
SUPERTYPE OF (ONEOF
    (point,
    direction,
    vector,
    placement,
    cartesian_transformation_operator,
    curve,
    surface,
    edge_curve,
    face_surface,
    poly_loop,
    vertex_point,
    solid_model,
    boolean_result,
    sphere,
    right_circular_cone,
    right_circular_cylinder,
    torus,
    block,
    primitive_2d,
    right_angular_wedge,
    eccentric_cone,
    ellipsoid,
    faceted_primitive,
    rectangular_pyramid,
    cyclide_segment_solid,
    volume,
    half_space_solid,
    half_space_2d,
    shell_based_surface_model,
    face_based_surface_model,
    shell_based_wireframe_model,
    edge_based_wireframe_model,
    geometric_set))
SUBTYPE OF (representation_item);
DERIVE
    dim : dimension_count := dimension_of(SELF);
WHERE
    WRG2 : SIZEOF (QUERY (using_rep <= using_representations (SELF) | NOT

```

```

        ('STRUCTURAL_FRAME_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
         IN TYPEOF (using_rep.context_of_items))) = 0;
END_ENTITY; (* STEP Part 42 (Modified in 2nd edition) *)

```

```

ENTITY geometric_set
SUPERTYPE OF (ONEOF(geometric_curve_set, geometric_set_replica))
SUBTYPE OF (geometric_representation_item);
    elements : SET [1:?] OF geometric_set_select;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY geometric_set_replica
SUBTYPE OF (geometric_set);
    parent_set : geometric_set;
    transformation : cartesian_transformation_operator;
DERIVE
    SELF\geometric_set.elements : SET [1:?] OF geometric_set_select :=
        build_transformed_set(transformation, parent_set);
WHERE
    WRG3 : acyclic_set_replica(SELF, parent_set);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY global_location
SUBTYPE OF (geographical_location);
    latitude_degrees : degrees_rotation;
    latitude_minutes : minutes_rotation;
    latitude_seconds : OPTIONAL seconds_rotation;
    longitude_degrees : degrees_rotation;
    longitude_minutes : minutes_rotation;
    longitude_seconds : OPTIONAL seconds_rotation;
END_ENTITY;

```

```

ENTITY global_uncertainty_assigned_context
    SUBTYPE OF (representation_context);
        uncertainty : SET [1:?] OF uncertainty_measure_with_unit;
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

```

```

ENTITY global_unit_assigned_context
    SUBTYPE OF (representation_context);
        units : SET [1:?] OF unit;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY grid
SUPERTYPE OF (ONEOF
    (grid_of_building,
     grid_of_site,
     grid_of_structure) ANDOR ONEOF
    (grid_orthogonal,

```

```

        grid_skewed,
        grid_radial));
    grid_name : label;
    grid_description : OPTIONAL text;
    grid_use : OPTIONAL text;
    DERIVE
        gridlines : SET [1:?] OF gridline := bag_to_set
            (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.GRIDLINE.PARENT_GRID'));
        grid_levels : SET [0:?] OF grid_level := bag_to_set
            (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.GRID_LEVEL.PARENT_GRID'));
    INVERSE
        constituent_lines : SET [1:?] OF gridline FOR parent_grid;
    END_ENTITY;

```

```

    ENTITY grid_intersection
    SUPERTYPE OF (grid_intersection_resolved);
        grid_intersection_name : label;
        gridlines : SET [2:2] OF gridline;
        level : OPTIONAL grid_level;
    WHERE
        WRG4 : gridlines[1].parent_grid := gridlines[2].parent_grid;
        WRG5 : NOT (EXISTS(level) AND (level.parent_grid :<>: gridlines[1].parent_grid));
    END_ENTITY;

```

```

    ENTITY grid_intersection_resolved
    SUBTYPE OF (grid_intersection);
        resolution_point : geographical_location;
    END_ENTITY;

```

```

    ENTITY grid_level
    SUBTYPE OF (plane);
        parent_grid : grid;
    UNIQUE
        URG1 : SELFelementary_surface.position;
    END_ENTITY;

```

```

    ENTITY grid_of_building
    SUBTYPE OF (grid);
        grid_for_building : building;
    END_ENTITY;

```

```

    ENTITY grid_of_site
    SUBTYPE OF (grid);
        grid_for_site : site;
    END_ENTITY;

```



```
ENTITY grid_of_structure
SUBTYPE OF (grid);
    grid_for_structure : structure;
END_ENTITY;
```

```
ENTITY grid_offset;
    intersection : grid_intersection;
    offset : LIST [2:3] OF length_measure_with_unit;
END_ENTITY;
```

```
ENTITY grid_orthogonal
SUBTYPE OF (grid);
    spacing_1 : LIST [0:?] OF positive_length_measure_with_unit;
    spacing_2 : LIST [0:?] OF positive_length_measure_with_unit;
END_ENTITY;
```

```
ENTITY grid_radial
SUBTYPE OF (grid);
    spacing_1 : LIST [0:?] OF plane_angle_measure_with_unit;
END_ENTITY;
```

```
ENTITY grid_skewed
SUBTYPE OF (grid);
    skew_angle : plane_angle_measure_with_unit;
    spacing_1 : LIST [0:?] OF positive_length_measure_with_unit;
    spacing_2 : LIST [0:?] OF positive_length_measure_with_unit;
END_ENTITY;
```

```
ENTITY gridline
SUBTYPE OF (plane);
    parent_grid : grid;
    preceding_line : OPTIONAL gridline;
INVERSE
    succeeding_line : SET [0:1] OF gridline FOR preceding_line;
UNIQUE
    URG2 : SELFelementary_surface.position;
WHERE
    WRG7 : NOT( EXISTS(preceding_line) AND (preceding_line :=: (SELF)) );
END_ENTITY;
```

(* [Modified for LPM/6](#) *)

```
ENTITY group
SUPERTYPE OF (media_file);
    group_name : label;
    group_description : text;
DERIVE
    id : identifier := get_id_value (SELF);
```

WHERE

WRG8 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
 'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
 END_ENTITY; (* STEP Part 41 2nd edition *)

(* [Modified for LPM/6](#) - see Issue 90 and 94 *)

ENTITY group_assignment

ABSTRACT SUPERTYPE OF (ONEOF

(flavour,
 group_of_analysis_data,
 group_of_design_data,
 group_of_physical_data,
 group_of_project_definition_data,
 group_of_structural_data,
 group_of_generic_data,
 managed_data_group) ANDOR
 group_assignment_actioned ANDOR
 group_assignment_approved ANDOR
 media_content ANDOR
 project_data_group);

assigned_group : group;

DERIVE

role : object_role := get_role (SELF);

WHERE

WRG9 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
 'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
 END_ENTITY; (* STEP Part 41 2nd edition *)

ENTITY group_assignment_actioned

SUBTYPE OF (group_assignment);

assigned_action : action;

UNIQUE

URG3 : SELF\group_assignment.assigned_group, assigned_action;

END_ENTITY;

ENTITY group_assignment_approved

SUBTYPE OF (group_assignment);

assigned_approval : approval;

UNIQUE

URG4 : SELF\group_assignment.assigned_group, assigned_approval;

END_ENTITY;

ENTITY group_of_analysis_data

SUBTYPE OF (group_assignment);

items : SET [2:?] OF select_analysis_item;

END_ENTITY;

```
ENTITY group_of_design_data
SUBTYPE OF (group_assignment);
    items : SET [2:?] OF select_design_item;
END_ENTITY;
```

```
ENTITY group_of_generic_data
SUBTYPE OF (group_assignment);
    items : SET [1:?] OF select_generic_item;
END_ENTITY;
```

```
ENTITY group_of_physical_data
SUBTYPE OF (group_assignment);
    items : SET [2:?] OF select_physical_item;
END_ENTITY;
```

```
ENTITY group_of_project_definition_data
SUBTYPE OF (group_assignment);
    items : SET [2:?] OF select_project_definition_item;
END_ENTITY;
```

```
ENTITY group_of_structural_data
SUBTYPE OF (group_assignment);
    items : SET [1:?] OF select_structural_item;
END_ENTITY;
```

(* **Modified for LPM/6** *)

```
ENTITY group_relationship
SUPERTYPE OF (group_usage);
    name : label;
    description : OPTIONAL text;
    relating_group : group;
    related_group : group;
END_ENTITY; (* STEP Part 41 (modified in 2nd edition) expanded *)
```

```
ENTITY group_usage
SUBTYPE OF (group_relationship);
UNIQUE
    URG5 : SELF\group_relationship.name,
        SELF\group_relationship.relatng_group,
        SELF\group_relationship.related_group;
WHERE
    WRG6 : acyclic_group_relationship(SELF, [SELF\group_relationship.related_group],
        'STRUCTURAL_FRAME_SCHEMA.GROUP_RELATIONSHIP.RELATED_GROUP');
END_ENTITY;
```

(* New for LPM/6 *)

ENTITY half_space_2d

SUBTYPE OF (geometric_representation_item);

base_curve: curve;

agreement_flag: BOOLEAN;

END_ENTITY; (* STEP Part 42 (New in 2nd edition) *)

ENTITY half_space_solid

SUPERTYPE OF (boxed_half_space)

SUBTYPE OF (geometric_representation_item);

base_surface : surface;

agreement_flag : BOOLEAN;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

ENTITY hexahedron_volume

SUBTYPE OF (volume);

points : LIST[8:8] OF cartesian_point;

WHERE

WRH1: above_plane(points[1], points[2], points[3], points[4]) = 0.0;

WRH2: above_plane(points[5], points[8], points[7], points[6]) = 0.0;

WRH3: above_plane(points[1], points[4], points[8], points[5]) = 0.0;

WRH4: above_plane(points[4], points[3], points[7], points[8]) = 0.0;

WRH5: above_plane(points[3], points[2], points[6], points[7]) = 0.0;

WRH6: above_plane(points[1], points[5], points[6], points[2]) = 0.0;

WRH7: same_side([points[1], points[2], points[3]],

[points[5], points[6], points[7], points[8]]);

WRH8: same_side([points[1], points[4], points[8]],

[points[3], points[7], points[6], points[2]]);

WRH9: same_side([points[1], points[2], points[5]],

[points[3], points[7], points[8], points[4]]);

WRH10: same_side([points[5], points[6], points[7]],

[points[1], points[2], points[3], points[4]]);

WRH11: same_side([points[3], points[7], points[6]],

[points[1], points[4], points[8], points[5]]);

WRH12: same_side([points[3], points[7], points[8]],

[points[1], points[5], points[6], points[2]]);

WRH13: points[1].dim = 3;

END_ENTITY; (* STEP part 42 (new for 2nd edition) *)

ENTITY hyperbola

SUBTYPE OF (conic);

semi_axis : positive_length_measure;

semi_imag_axis : positive_length_measure;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

```
ENTITY id_attribute;
  attribute_value : identifier;
  identified_item : id_attribute_select;
END_ENTITY; (* STEP Part 41 2nd edition *)
```

```
ENTITY inertia_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
  WRI1 : 'STRUCTURAL_FRAME_SCHEMA.INERTIA_UNIT' IN
    TYPEOF (SELF\measure_with_unit.unit_component);
  WRI2 : 'STRUCTURAL_FRAME_SCHEMA.INERTIA_MEASURE' IN
    TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;
```

```
ENTITY inertia_unit
SUBTYPE OF (derived_unit);
WHERE
  WRI3 : SIZEOF(SELF\derived_unit.elements) = 1;
  WRI4 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
    TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
    AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 4.0);
END_ENTITY;
```

```
ENTITY intersection_curve
SUBTYPE OF (surface_curve);
WHERE
  WRI5 : SIZEOF(SELF\surface_curve.associated_geometry) = 2;
  WRI6 : associated_surface(SELF\surface_curve.associated_geometry[1]) <>
    associated_surface(SELF\surface_curve.associated_geometry[2]);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

(* New for LPM/6 - See Issue 96 *)

```
ENTITY item_cost_code
SUPERTYPE OF (item_cost_code_with_source);
  cost_code : label;
  description : OPTIONAL text;
END_ENTITY;
```

(* New for LPM/6 - See Issue 96 *)

```
ENTITY item_cost_code_assigned;
  code : item_cost_code;
  costed_item : structural_frame_item;
UNIQUE
  URI8 : code, costed_item;
END_ENTITY;
```

(* **New for LPM/6** - See Issue 96 *)

```
ENTITY item_cost_code_with_source
SUBTYPE OF (item_cost_code);
    source : item_ref_source;
UNIQUE
    URI9 : SELF\item_cost_code.cost_code, source;
END_ENTITY;
```

```
ENTITY item_defined_transformation;
    name : label;
    description : text;
    transform_item_1 : representation_item;
    transform_item_2 : representation_item;
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)
```

(* **Modified for LPM/6** - See Issue 81 *)

```
ENTITY item_property
SUPERTYPE OF (item_property_with_source);
    property_name : label;
    property_description : OPTIONAL text;
    property_value : measure_select;
END_ENTITY;
```

```
ENTITY item_property_assigned;
    property : item_property;
    item : structural_frame_item;
UNIQUE
    URI1 : property, item;
END_ENTITY;
```

(* **New for LPM/6** - See Issue 81 *)

```
ENTITY item_property_with_source
SUBTYPE OF (item_property);
    source : item_ref_source;
UNIQUE
    URI7 : SELF\item_property.property_name, source;
END_ENTITY;
```

```
ENTITY item_ref_source
ABSTRACT SUPERTYPE OF (ONEOF
    (item_ref_source_standard,
    item_ref_source_proprietary,
    item_ref_source_library));
END_ENTITY;
```

```

ENTITY item_ref_source_documented;
    documented_item_source : item_ref_source;
    document_reference : document_usage_constraint;
UNIQUE
    URI2 : documented_item_source, document_reference;
END_ENTITY;

```

(* [Modified for LPM/6](#) - See Issue 81 *)

```

ENTITY item_ref_source_library
SUBTYPE OF (item_ref_source);
    library_owner : person_and_organization;
    library_name : label;
    date_of_library : calendar_date;
    version_of_library : OPTIONAL label;
INVERSE
    library_items : SET [0:?] OF item_reference_library FOR source;
END_ENTITY;

```

(* [Modified for LPM/6](#) - See Issue 81 *)

```

ENTITY item_ref_source_proprietary
SUBTYPE OF (item_ref_source);
    manufacturers_name : organization;
    manufacturers_range : label;
    year_of_range : year_number;
    version_of_range : OPTIONAL label;
INVERSE
    proprietary_items : SET [0:?] OF item_reference_proprietary FOR source;
END_ENTITY;

```

(* [Modified for LPM/6](#) - See Issue 81 *)

```

ENTITY item_ref_source_standard
SUBTYPE OF (item_ref_source);
    standardization_organization : label;
    name_of_standard : label;
    year_of_standard : year_number;
    version_of_standard : OPTIONAL label;
INVERSE
    standard_items : SET [0:?] OF item_reference_standard FOR source;
END_ENTITY;

```

```

ENTITY item_reference
SUPERTYPE OF (ONEOF
    (item_reference_standard,
    item_reference_proprietary,
    item_reference_library));
    ref : identifier;
WHERE

```

WRI7 : LENGTH(ref) > 0;
 END_ENTITY;

ENTITY item_reference_assigned;
 assigned_reference : item_reference;
 assigned_to_item : structural_frame_item;
 UNIQUE
 URI3 : assigned_reference, assigned_to_item;
 END_ENTITY;

ENTITY item_reference_library
 SUBTYPE OF (item_reference);
 source : item_ref_source_library;
 UNIQUE
 URI4 : SELF\item_reference.ref, source;
 END_ENTITY;

ENTITY item_reference_proprietary
 SUBTYPE OF (item_reference);
 source : item_ref_source_proprietary;
 UNIQUE
 URI5 : SELF\item_reference.ref, source;
 END_ENTITY;

ENTITY item_reference_standard
 SUBTYPE OF (item_reference);
 source : item_ref_source_standard;
 UNIQUE
 URI6 : SELF\item_reference.ref, source;
 END_ENTITY;

(* [Modified for LPM/6](#) *)

ENTITY joint_system
 SUPERTYPE OF (ONEOF
 (joint_system_amorphous,
 joint_system_chemical,
 joint_system_complex,
 joint_system_mechanical,
 joint_system_welded))
 SUBTYPE OF (structural_frame_item);
 place_of_assembly : OPTIONAL shop_or_site;
 DERIVE
 joint_system_number : INTEGER := SELF\structural_frame_item.item_number;
 joint_system_name : LABEL := SELF\structural_frame_item.item_name;
 design_uses : SET [0:?] OF design_joint_system := bag_to_set
 (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
 DESIGN_JOINT_SYSTEM.DESIGN_JOINT_SYSTEM_SPEC'));


```

    physical_uses : SET [0:?] OF located_joint_system := bag_to_set
      (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
        LOCATED_JOINT_SYSTEM.DESCRPTIVE_JOINT_SYSTEM'));
  UNIQUE
    URJ1 : joint_system_number,joint_system_name;
  END_ENTITY;

  ENTITY joint_system_amorphous
  SUBTYPE OF (joint_system);
    specification : chemical_mechanism;
  END_ENTITY;

  ENTITY joint_system_chemical
  SUBTYPE OF (joint_system);
    joining_surface : surface;
    joined_surface : surface;
    specification : LIST [1:?] OF chemical_mechanism;
  DERIVE
    number_of_layers : INTEGER := SIZEOF(specification);
  WHERE
    WRJ1 : joining_surface :<>: joined_surface;
  END_ENTITY;

  ENTITY joint_system_complex
  SUBTYPE OF (joint_system);
    systems : LIST [1:?] OF joint_system;
  WHERE
    WRJ2 : SIZEOF(QUERY(joint <* systems | joint :=: (SELF)) ) = 0;
  END_ENTITY;

  ENTITY joint_system_mechanical
  SUBTYPE OF (joint_system);
    layout_points : LIST [1:?] OF point;
    mechanism : fastener_mechanism;
  END_ENTITY;

  (* Modified for LPM/6 - SUBTYPE added *)
  ENTITY joint_system_welded
  SUPERTYPE OF (ONEOF
    (joint_system_welded_point,
    joint_system_welded_linear,
    joint_system_welded_surface,
    joint_system_welded_with_shape))
  SUBTYPE OF (joint_system);
    weld_specification : weld_mechanism;
  END_ENTITY;

```

(* **Modified for LPM/6** - See Issues 87 and 88 *)

ENTITY joint_system_welded_linear

SUBTYPE OF (joint_system_welded);

weld_path : composite_curve;

WHERE

WRJ3 : NOT((SELF\joint_system_welded.weld_specification.weld_mechanism_type =
SPOT_WELD)

OR (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
PLUG_WELD));

END_ENTITY;

(* **Modified for LPM/6** - See Issues 87 and 88 *)

ENTITY joint_system_welded_point

SUBTYPE OF (joint_system_welded);

weld_position : point;

WHERE

WRJ4 : (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
SPOT_WELD) OR

(SELF\joint_system_welded.weld_specification.weld_mechanism_type =
PLUG_WELD) OR

(SELF\joint_system_welded.weld_specification.weld_mechanism_type =
STUD_WELD);

END_ENTITY;

(* **Modified for LPM/6** - See Issues 87 and 88 *)

ENTITY joint_system_welded_surface

SUBTYPE OF (joint_system_welded);

weld_surface : bounded_surface;

WHERE

WRJ5 : (SELF\joint_system_welded.weld_specification.weld_mechanism_type =
PLUG_WELD) OR

(SELF\joint_system_welded.weld_specification.weld_mechanism_type =
SURFACING_WELD) OR

(SELF\joint_system_welded.weld_specification.weld_mechanism_type =
SLOT_WELD);

END_ENTITY;

(* **New for LPM/6** - See Issues 87 and 88 *)

ENTITY joint_system_welded_with_shape

SUBTYPE OF (joint_system_welded);

weld_shape : shape_representation_with_units;

END_ENTITY;

ENTITY length_measure_with_unit

SUBTYPE OF (measure_with_unit);

WHERE

WRL38 : 'STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
TYPEOF (SELF\measure_with_unit.unit_component);

```

WRL39 : 'STRUCTURAL_FRAME_SCHEMA.LENGTH_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

```

```

ENTITY length_unit
SUBTYPE OF (named_unit);
WHERE
    WRL1 : (SELF\named_unit.dimensions.length_exponent = 1.0) AND
            (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
            (SELF\named_unit.dimensions.time_exponent = 0.0) AND
            (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
            (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
            (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
            (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY line
SUBTYPE OF (curve);
    pnt : cartesian_point;
    dir : vector;
WHERE
    WRL2 : dir.dim = pnt.dim;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY linear_acceleration_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
    WRL3 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_ACCELERATION_UNIT' IN
            TYPEOF (SELF\measure_with_unit.unit_component);
    WRL4 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_ACCELERATION_MEASURE' IN
            TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```

```

ENTITY linear_acceleration_unit
SUBTYPE OF (derived_unit);
WHERE
    WRL5 : SIZEOF (SELF\derived_unit.elements) = 2;
    WRL6 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
            TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
            AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRL7 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN
            TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
            AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -2.0);
END_ENTITY;

```

```

ENTITY linear_stiffness_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);

```

WHERE

WRL8 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_STIFFNESS_UNIT' IN

 TYPEOF (SELF\measure_with_unit.unit_component);

WRL9 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_STIFFNESS_MEASURE' IN

 TYPEOF (SELF\measure_with_unit.value_component);

END_ENTITY;

ENTITY linear_stiffness_unit

SUBTYPE OF (derived_unit);

WHERE

WRL10 : SIZEOF (SELF\derived_unit.elements) = 2;

WRL11 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN

 TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))

 AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);

WRL12 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN

 TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))

 AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);

END_ENTITY;

ENTITY linear_velocity_measure_with_unit

SUBTYPE OF (derived_measure_with_unit);

WHERE

WRL13 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_VELOCITY_UNIT' IN

 TYPEOF (SELF\measure_with_unit.unit_component);

WRL14 : 'STRUCTURAL_FRAME_SCHEMA.LINEAR_VELOCITY_MEASURE' IN

 TYPEOF (SELF\measure_with_unit.value_component);

END_ENTITY;

ENTITY linear_velocity_unit

SUBTYPE OF (derived_unit);

WHERE

WRL15 : SIZEOF (SELF\derived_unit.elements) = 2;

WRL16 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN

 TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))

 AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);

WRL17 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN

 TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))

 AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);

END_ENTITY;

(* [Modified for LPM/6](#) - see Issue 5 *)

ENTITY load

ABSTRACT SUPERTYPE OF (ONEOF (load_element, load_node, load_member,
 load_connection));

 parent_load_case : load_case;

 load_name : label;

load_description : OPTIONAL text;
END_ENTITY;

(* Modified for LPM/6 *)

ENTITY load_case
SUPERTYPE OF (load_case_documented);
load_case_name : label;
load_case_factor : OPTIONAL REAL;
governing_analyses : SET [0:?] OF analysis_method;
time_variation : physical_action;
DERIVE
load_components : SET [1:?] OF load := bag_to_set
(USEDIN(SELF,
'STRUCTURAL_FRAME_SCHEMA.LOAD.PARENT_LOAD_CASE'));
INVERSE
loads : SET [1:?] OF load FOR parent_load_case;
WHERE
WRL60 : NOT ((SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_ELEMENT') IN
TYPEOF(components))) > 0)
AND (SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_MEMBER') IN
TYPEOF(components))) > 0));
WRL61 : NOT ((SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_NODE') IN
TYPEOF(components))) > 0)
AND (SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_CONNECTION') IN
TYPEOF(components))) > 0));
WRL62 : NOT ((SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_NODE') IN
TYPEOF(components))) > 0)
AND (SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_MEMBER') IN
TYPEOF(components))) > 0));
WRL63 : NOT ((SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_ELEMENT') IN
TYPEOF(components))) > 0)
AND (SIZEOF(QUERY(components <* load_components |
('STRUCTURAL_FRAME_SCHEMA.LOAD_CONNECTION') IN
TYPEOF(components))) > 0));
END_ENTITY;

ENTITY load_case_documented
SUBTYPE OF (load_case);
code_ref : document_usage_constraint;
END_ENTITY;

```

ENTITY load_combination_occurrence;
    load_combination_factor : OPTIONAL REAL;
    parent_load_combination : loading_combination;
    component_load_case : load_case;
UNIQUE
    URL1 : parent_load_combination, component_load_case;
END_ENTITY;

```

```

ENTITY load_connection
SUBTYPE OF (load);
    supporting_connection : assembly_design_structural_connection;
    load_values : applied_load;
WHERE
    WRL59 : NOT ('STRUCTURAL_FRAME_SCHEMA.
        APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(load_values));
END_ENTITY;

```

```

ENTITY load_element
ABSTRACT SUPERTYPE OF (ONEOF
    (load_element_distributed,
    load_element_thermal,
    load_element_concentrated))
SUBTYPE OF (load);
    supporting_element : element;
    load_position_label : OPTIONAL label;
    reference_system : OPTIONAL text;
    destabilizing_load : OPTIONAL BOOLEAN;
    global_or_local : global_or_local_load;
END_ENTITY;

```

```

ENTITY load_element_concentrated
SUBTYPE OF (load_element);
    load_position : point;
    load_value : applied_load;
WHERE
    WRL40 : NOT ('STRUCTURAL_FRAME_SCHEMA.
        APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(load_value));
END_ENTITY;

```

(* [Modified for LPM/6](#) *)

```

ENTITY load_element_distributed
ABSTRACT SUPERTYPE OF (ONEOF
    (load_element_distributed_surface,
    load_element_distributed_curve))
SUBTYPE OF (load_element);
    projected_or_true : projected_or_true_length;

```

WHERE

WRL18 : NOT ('STRUCTURAL_FRAME_SCHEMA.ELEMENT_POINT' IN
 TYPEOF (SELFload_element.supporting_element));
 END_ENTITY;

ENTITY load_element_distributed_curve

SUPERTYPE OF (load_element_distributed_curve_line)

SUBTYPE OF (load_element_distributed);

start_load_value : applied_load;

end_load_value : applied_load;

curve_definition : curve;

WHERE

WRL41 : NOT ('STRUCTURAL_FRAME_SCHEMA.
 APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(start_load_value));

WRL42 : NOT ('STRUCTURAL_FRAME_SCHEMA.
 APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(end_load_value));

WRL44 : TYPEOF(start_load_value) = TYPEOF(end_load_value);

END_ENTITY;

ENTITY load_element_distributed_curve_line

SUBTYPE OF (load_element_distributed_curve);

WHERE

WRL19 : 'STRUCTURAL_FRAME_SCHEMA.LINE' IN TYPEOF
 (SELFload_element_distributed_curve.curve_definition);

END_ENTITY;

ENTITY load_element_distributed_surface

ABSTRACT SUPERTYPE OF (ONEOF

(load_element_distributed_surface_uniform,

load_element_distributed_surface_varying))

SUBTYPE OF (load_element_distributed);

surface_definition : bounded_surface;

WHERE

WRL20 : NOT ('STRUCTURAL_FRAME_SCHEMA.ELEMENT_CURVE' IN
 TYPEOF (SELFload_element.supporting_element));

END_ENTITY;

ENTITY load_element_distributed_surface_uniform

SUBTYPE OF (load_element_distributed_surface);

load_value : applied_load_static_pressure;

END_ENTITY;

ENTITY load_element_distributed_surface_varying

SUBTYPE OF (load_element_distributed_surface);

load_values : SET [4:4] OF applied_load_static_pressure;

WHERE

```

WRL45 : 'STRUCTURAL_FRAME_SCHEMA.
        RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF
        (SELFload_element_distributed_surface.surface_definition);
END_ENTITY;

```

```

ENTITY load_element_thermal
SUBTYPE OF (load_element);
    temperature_gradients : LIST [1:3] OF measure_with_unit;
END_ENTITY;

```

(* New for LPM/6 - see Issue 5 *)

```

ENTITY load_member
ABSTRACT SUPERTYPE OF (ONEOF(load_member_distributed,
    load_member_concentrated))
SUBTYPE OF (load);
    supporting_member : assembly_design_structural_member;
    load_position_label : OPTIONAL label;
    reference_system : OPTIONAL text;
    destabilizing_load : OPTIONAL BOOLEAN;
    global_or_local : global_or_local_load;
END_ENTITY;

```

(* New for LPM/6 - see Issue 5 *)

```

ENTITY load_member_concentrated
SUBTYPE OF (load_member);
    load_position : point;
    load_value : applied_load;
WHERE
    WRL53 : NOT ('STRUCTURAL_FRAME_SCHEMA.
        APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(load_value));
END_ENTITY;

```

(* New for LPM/6 - see Issue 5 *)

```

ENTITY load_member_distributed
ABSTRACT SUPERTYPE OF (ONEOF
    (load_member_distributed_surface,
    load_member_distributed_curve))
SUBTYPE OF (load_member);
    projected_or_true : projected_or_true_length;
END_ENTITY;

```

(* New for LPM/6 - see Issue 5 *)

```

ENTITY load_member_distributed_curve
SUPERTYPE OF (load_member_distributed_curve_line)
SUBTYPE OF (load_member_distributed);
    start_load_value : applied_load;
    end_load_value : applied_load;
    curve_definition : curve;

```


WHERE

WRL54 : NOT ('STRUCTURAL_FRAME_SCHEMA.
APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(start_load_value));

WRL55 : NOT ('STRUCTURAL_FRAME_SCHEMA.
APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(end_load_value));

WRL56 : TYPEOF(start_load_value) = TYPEOF(end_load_value);

END_ENTITY;

(* New for LPM/6 - see Issue 5 *)

ENTITY load_member_distributed_curve_line

SUBTYPE OF (load_member_distributed_curve);

WHERE

WRL57 : 'STRUCTURAL_FRAME_SCHEMA.LINE' IN TYPEOF
(SELF\load_member_distributed_curve.curve_definition);

END_ENTITY;

(* New for LPM/6 - see Issue 5 *)

ENTITY load_member_distributed_surface

ABSTRACT SUPERTYPE OF (ONEOF

(load_member_distributed_surface_uniform,
load_member_distributed_surface_varying))

SUBTYPE OF (load_member_distributed);

surface_definition : bounded_surface;

END_ENTITY;

(* New for LPM/6 - see Issue 5 *)

ENTITY load_member_distributed_surface_uniform

SUBTYPE OF (load_member_distributed_surface);

load_value : applied_load_static_pressure;

END_ENTITY;

(* New for LPM/6 - see Issue 5 *)

ENTITY load_member_distributed_surface_varying

SUBTYPE OF (load_member_distributed_surface);

load_values : SET [4:4] OF applied_load_static_pressure;

WHERE

WRL58 : 'STRUCTURAL_FRAME_SCHEMA.
RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF
(SELF\load_element_distributed_surface.surface_definition);

END_ENTITY;

ENTITY load_node

SUBTYPE OF (load);

supporting_node: node;

load_values : applied_load;

WHERE

WRL43 : NOT ('STRUCTURAL_FRAME_SCHEMA.
APPLIED_LOAD_STATIC_PRESSURE' IN TYPEOF(load_values));

END_ENTITY;

ENTITY loaded_product;
 product : structural_frame_product;
 load_definition : applied_load;
 time_variation : OPTIONAL physical_action;
 END_ENTITY;

(* [Modified for LPM/6](#) *)

ENTITY loading_combination;
 loading_combination_name : label;
 loading_combination_description : OPTIONAL text;
 loaded_model : OPTIONAL analysis_model;
 DERIVE
 cases : SET [1:?] OF load_combination_occurrence := bag_to_set (USEDIN(SELF,
 'Structural_Frame_Schema.
 LOAD_COMBINATION_OCCURRENCE.PARENT_LOAD_COMBINATION'));
 INVERSE
 load_cases : SET [1:?] OF load_combination_occurrence FOR parent_load_combination;
 END_ENTITY;

ENTITY local_time;
 hour_component : hour_in_day;
 minute_component : OPTIONAL minute_in_hour;
 second_component : OPTIONAL second_in_minute;
 zone : coordinated_universal_time_offset;
 WHERE
 WRL21 : valid_time (SELF);
 END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

(* [Modified for LPM/6](#) - see Issue 9 *)

ENTITY located_assembly
 SUPERTYPE OF (located_assembly_child ANDOR located_assembly_marked)
 SUBTYPE OF (located_item);
 location_on_grid : OPTIONAL SET [1:?] OF grid_offset;
 descriptive_assembly : assembly;
 parent_structure : structure_select;
 DERIVE
 component_parts : SET [0:?] OF located_part := bag_to_set (USEDIN(SELF,
 'Structural_Frame_Schema.Located_Part.Parent_Assembly'));
 sub_assemblies : SET [0:?] OF located_assembly := bag_to_set (USEDIN(SELF,
 'Structural_Frame_Schema.Located_Assembly.Parent_Structure'
)));
 UNIQUE
 URL2 : SELF\located_item.location, descriptive_assembly, parent_structure;
 WHERE
 WRL22 : SELF\located_item.location.coord_system_use =
 'Assembly Coordinate System';

```

WRL46 : parent_structure :<>: (SELF);
END_ENTITY;

```

```

(* Modified for LPM/6 - see Issue 9 *)

```

```

ENTITY located_assembly_child
SUBTYPE OF (located_assembly);
WHERE
  WRL47 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING_CHILD'
    IN TYPEOF(SELF\located_assembly.descriptive_assembly);
  WRL48 : ('STRUCTURAL_FRAME_SCHEMA.LOCATED_ASSEMBLY' IN
    TYPEOF(SELF\located_assembly.parent_structure)) AND
    ('STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN
    TYPEOF(SELF\located_assembly.parent_structure.descriptive_assembly));
  WRL49 : SELF\located_assembly.descriptive_assembly.parent_assembly :=:
    SELF\located_assembly.parent_structure.descriptive_assembly;
  WRL50 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
    TYPEOF (SELF\located_item.location);
  WRL51 : SELF\located_item.location.parent_coord_system :=:
    SELF\located_assembly.parent_structure\located_item.location;
  WRL52 : NOT ( (SIZEOF(SELF\located_assembly.component_parts) = 0)
    AND (SIZEOF(SELF\located_assembly.sub_assemblies) = 0) );
END_ENTITY;

```

```

(* New for LPM/6 - see Issue 96 *)

```

```

ENTITY located_assembly_marked
SUBTYPE OF (located_assembly);
  assembly_mark : label;
  client_mark : OPTIONAL label;
  prelim_mark : OPTIONAL label;
  shipping_mark : OPTIONAL label;
  bar_code : OPTIONAL label;
END_ENTITY;

```

```

ENTITY located_feature
ABSTRACT SUPERTYPE OF (ONEOF
  (located_feature_for_assembly,
  located_feature_for_design_part,
  located_feature_for_located_assembly,
  located_feature_for_located_part,
  located_feature_for_part))
SUBTYPE OF (located_item);
  descriptive_feature : feature;
WHERE
  WRL23 : SELF\located_item.location.coord_system_use = 'Feature Coordinate System';
END_ENTITY;

```

```

ENTITY located_feature_for_assembly
SUBTYPE OF (located_feature);
    modified_assembly : assembly;
UNIQUE
    URL3 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
        modified_assembly;
END_ENTITY;

```

```

ENTITY located_feature_for_design_part
SUBTYPE OF (located_feature);
    modified_part : design_part;
UNIQUE
    URL4 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
        modified_part;
END_ENTITY;

```

```

ENTITY located_feature_for_located_assembly
SUBTYPE OF (located_feature);
    modified_assembly : located_assembly;
UNIQUE
    URL5 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
        modified_assembly;
WHERE
    WRL24 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
        TYPEOF (SELF\located_item.location);
    WRL25 : SELF\located_item.location.parent_coord_system :=:
        modified_assembly\located_item.location;
END_ENTITY;

```

```

ENTITY located_feature_for_located_part
SUPERTYPE OF (located_feature_joint_dependent)
SUBTYPE OF (located_feature);
    modified_part : located_part;
UNIQUE
    URL6 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
        modified_part;
WHERE
    WRL26 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
        TYPEOF (SELF\located_item.location);
    WRL27 : SELF\located_item.location.parent_coord_system :=:
        modified_part\located_item.location;
END_ENTITY;

```

```

ENTITY located_feature_for_part
SUBTYPE OF (located_feature);
    modified_part : part;
UNIQUE

```

```

        URL7 : SELF\located_item.location, SELF\located_feature.descriptive_feature,
            modified_part;
END_ENTITY;

```

```

ENTITY located_feature_joint_dependent
SUBTYPE OF (located_feature_for_located_part);
    feature_for_joint : located_joint_system;
END_ENTITY;

```

```

ENTITY located_item
ABSTRACT SUPERTYPE OF (ONEOF
    (located_assembly,
    located_feature,
    located_joint_system,
    located_part,
    located_site,
    located_structure))
SUBTYPE OF (structural_frame_item);
    location : coord_system;
END_ENTITY;

```

```

ENTITY located_joint_system
SUBTYPE OF (located_item);
    descriptive_joint_system : joint_system;
    parent_assembly : located_assembly;
DERIVE
    features : SET [0:?] OF located_feature_joint_dependent := bag_to_set
        (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
            LOCATED_FEATURE_JOINT_DEPENDENT.FEATURE_FOR_JOINT'));
UNIQUE
    URL8 : SELF\located_item.location, descriptive_joint_system, parent_assembly;
WHERE
    WRL28 : SELF\located_item.location.coord_system_use =
        'Joint System Coordinate System';
    WRL29 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
        TYPEOF (SELF\located_item.location);
    WRL30 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING' IN
        TYPEOF (parent_assembly.descriptive_assembly);
    WRL31 : SELF\located_item.location.parent_coord_system
        := parent_assembly\located_assembly\located_item.location;
END_ENTITY;

```

```

ENTITY located_part
SUBTYPE OF (located_item);
    descriptive_part : part;
    parent_assembly : located_assembly;
DERIVE

```

```

features : SET [0:?] OF located_feature_for_located_part := bag_to_set
  (USEDIN(SELF,'STRUCTURAL_FRAME_SCHEMA.
    LOCATED_FEATURE_FOR_LOCATED_PART.MODIFIED_PART'));
UNIQUE
  URL9 : SELF\located_item.location, descriptive_part, parent_assembly;
WHERE
  WRL32 : SELF\located_item.location.coord_system_use =
    'Part Coordinate System';
  WRL33 : 'STRUCTURAL_FRAME_SCHEMA.COORD_SYSTEM_CHILD' IN
    TYPEOF (SELF\located_item.location);
  WRL34 : 'STRUCTURAL_FRAME_SCHEMA.ASSEMBLY_MANUFACTURING'
    IN TYPEOF (parent_assembly.descriptive_assembly);
  WRL35 : SELF\located_item.location.parent_coord_system
    := parent_assembly\located_assembly\located_item.location;
END_ENTITY;

(* New for LPM/6 - see Issue 91 *)
ENTITY located_part_marked
SUBTYPE OF (located_part);
  piece_mark : OPTIONAL label;
  prelim_mark : OPTIONAL label;
  bar_code : OPTIONAL label;
  quantity : OPTIONAL INTEGER;
  main_piece : LOGICAL;
END_ENTITY;

ENTITY located_part_joint;
  part_joint_label : label;
  part_joint_nature : OPTIONAL text;
  logically_joined_parts : SET [2:2] OF located_part;
  required_joints : SET [1:?] OF located_joint_system;
END_ENTITY;

ENTITY located_site
SUBTYPE OF (located_item);
  descriptive_site : site;
  parent_project : project_select;
UNIQUE
  URL10 : SELF\located_item.location, descriptive_site, parent_project;
WHERE
  WRL36 : SELF\located_item.location.coord_system_use = 'Site Coordinate System';
END_ENTITY;

ENTITY located_structure
SUBTYPE OF (located_item);
  descriptive_structure : structure;
  parent_site : site_select;

```

UNIQUE

URL11 : SELF\located_item.location, descriptive_structure, parent_site;

WHERE

WRL37 : SELF\located_item.location.coord_system_use = 'Structure Coordinate System';

END_ENTITY;

ENTITY loop

SUPERTYPE OF (ONEOF(vertex_loop, edge_loop, poly_loop))

SUBTYPE OF (topological_representation_item);

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY managed_application_installation;

application_id : INTEGER;

application_name : label;

application_version : label;

application_description : OPTIONAL text;

application_vendor : OPTIONAL organization;

installation_id : INTEGER;

installation_name : label;

installation_owner : OPTIONAL organization;

UNIQUE

URM1 : application_id, installation_id;

WHERE

WRM1 : installation_id > 0;

WRM2 : application_id > 0;

END_ENTITY;

ENTITY managed_data_creation

SUBTYPE OF (managed_data_transaction);

DERIVE

created_set : SET [1:?] OF managed_data_item := bag_to_set (USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));

WHERE

WRM3 : SELF\managed_data_transaction.new_ids_assigned = TRUE;

WRM4 : SIZEOF(QUERY(tmp <* created_set | tmp.originating_application :<>:
(SELF\managed_data_transaction.application))) = 0;

END_ENTITY;

ENTITY managed_data_deleted;

data_status : data_status_type;

DERIVE

deleted_data_items : SET [1:?] OF managed_data_item := bag_to_set (USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.DATA_ITEM'));

END_ENTITY;

```

ENTITY managed_data_export
SUBTYPE OF (managed_data_transaction);
    data_destination : select_data_source;
DERIVE
    exported_set : SET [1:?] OF managed_data_item := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));
    assignments : SET [0:?] OF managed_data_group := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_GROUP.ITEMS'));
END_ENTITY;

```

```

ENTITY managed_data_group
SUBTYPE OF (group_assignment);
    items : SET [1:?] OF managed_data_transaction;
END_ENTITY;

```

```

ENTITY managed_data_import
SUBTYPE OF (managed_data_transaction);
    data_source : select_data_source;
DERIVE
    imported_set : SET [1:?] OF managed_data_item := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));
    assignments : SET [0:?] OF managed_data_group := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_GROUP.ITEMS'));
END_ENTITY;

```

(* **Modified for LPM/6** - See Issues 80 and 103 *)

```

ENTITY managed_data_item;
    instance_id : globally_unique_id;
    originating_application : OPTIONAL managed_application_installation;
    data_item : select_data_item;
    history : LIST [0:?] OF UNIQUE managed_data_transaction;
    original_data : LOGICAL;
UNIQUE
    URM5 : instance_id, data_item;
WHERE
    WRM36 : unique_data_item(data_item);
END_ENTITY;

```

(* **New for LPM/6** - See Issues 80 and 102 *)

```

ENTITY managed_data_item_with_history
SUBTYPE OF (managed_data_item);
DERIVE
    number_of_uses : INTEGER := SIZEOF(SELF\managed_data_item.history);
    first_managing_application : managed_application_installation :=
        SELF\managed_data_item.history[1]
        \managed_data_transaction.application;
    first_managing_person : person :=
        SELF\managed_data_item.history[1]

```



```

        \managed_data_transaction.user.the_person;
date_first_managed : calendar_date :=
    SELF\managed_data_item.history[1]
        \managed_data_transaction.processing_date.date_component;
last_managing_application : managed_application_installation :=
    SELF\managed_data_item.history[number_of_uses]
        \managed_data_transaction.application;
last_managing_person : person :=
    SELF\managed_data_item.history[number_of_uses]
        \managed_data_transaction.user.the_person;
date_last_managed : calendar_date :=
    SELF\managed_data_item.history[number_of_uses]
        \managed_data_transaction.processing_date.date_component;
WHERE
    WRM60 : number_of_uses > 0;
    WRM61 : EXISTS(SELF\managed_data_item.originating_application);
    WRM6 : first_managing_application :=: originating_application;
    WRM7 : ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_CREATION'
        IN TYPEOF(SELF\managed_data_item.history[1]));
    WRM8 : NOT ( (original_data = TRUE) AND
        ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_IMPORT'
            IN TYPEOF(SELF\managed_data_item.history[1])) );
    WRM37 : SIZEOF(QUERY(creation <* SELF\managed_data_item.history |
        ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_CREATION') IN
            TYPEOF(creation))) = 1;
    WRM38 : NOT( ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_DELETED'
        IN TYPEOF(SELF\managed_data_item.data_item))
        AND (number_of_uses = 1) );
    WRM39 : NOT( ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_DELETED'
        IN TYPEOF(SELF\managed_data_item.data_item))
        AND (SIZEOF(QUERY(modification <* SELF\managed_data_item.history |
        ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_MODIFICATION')
            IN TYPEOF(modification))) < 1) );
END_ENTITY;

```

ENTITY managed_data_modification

SUBTYPE OF (managed_data_transaction);

DERIVE

```

    modified_set : SET [1:?] OF managed_data_item := bag_to_set
        (USEDIN(SELF,
            'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.HISTORY'));

```

WHERE

```

    WRM9 : SELF\managed_data_transaction.new_ids_assigned = FALSE;

```

END_ENTITY;

ENTITY managed_data_transaction

SUPERTYPE OF (ONEOF(managed_data_export, managed_data_import) ANDOR

```

        ONEOF (managed_data_creation, managed_data_modification));
    application : managed_application_installation;
    user : person_and_organization;
    processing_date : date_and_time;
    new_ids_assigned : BOOLEAN;
    life_cycle_stage : OPTIONAL label;
    transaction_description : OPTIONAL text;
INVERSE
    processed_items : SET [1:?] OF managed_data_item FOR history;
UNIQUE
    URM3 : application, processing_date;
END_ENTITY;

```

```

ENTITY manifold_solid_brep
SUPERTYPE OF (ONEOF(brep_with_voids, faceted_brep))
SUBTYPE OF (solid_model);
    outer : closed_shell;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY map_location
SUBTYPE OF (geographical_location);
    map_name : label;
    map_code : identifier;
    eastings : length_measure_with_unit;
    northings : length_measure_with_unit;
END_ENTITY;

```

```

ENTITY mapped_item
SUBTYPE OF (representation_item);
    mapping_source : representation_map;
    mapping_target : representation_item;
WHERE
    WRM10 : acyclic_mapped_representation(using_representations(SELF), [SELF]);
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

```

```

ENTITY mass_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRM11 : 'STRUCTURAL_FRAME_SCHEMA.MASS_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
    WRM12 : 'STRUCTURAL_FRAME_SCHEMA.MASS_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

```

```

ENTITY mass_per_length_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE

```

```

WRM13 : 'STRUCTURAL_FRAME_SCHEMA.MASS_PER_LENGTH_UNIT' IN
    TYPEOF (SELF\measure_with_unit.unit_component);
WRM14 : 'STRUCTURAL_FRAME_SCHEMA.MASS_PER_LENGTH_MEASURE' IN
    TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```

```

ENTITY mass_per_length_unit
SUBTYPE OF (derived_unit);
WHERE
    WRM15 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRM16 : ('STRUCTURAL_FRAME_SCHEMA.MASS_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRM17 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);
END_ENTITY;

```

```

ENTITY mass_unit
SUBTYPE OF (named_unit);
WHERE
    WRM18 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND
        (SELF\named_unit.dimensions.mass_exponent = 1.0) AND
        (SELF\named_unit.dimensions.time_exponent = 0.0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY material
SUPERTYPE OF (ONEOF
    (material_anisotropic,
    material_isotropic,
    material_orthotropic) ANDOR
    material_constituent)
SUBTYPE OF (structural_frame_item);
DERIVE
    material_number : INTEGER := SELF\structural_frame_item.item_number;
    material_name : label := SELF\structural_frame_item.item_name;
    material_grade : BAG OF identifier := SELF\structural_frame_item.item_ref;
UNIQUE
    URM6 : material_number, material_name;
END_ENTITY;

```

```

ENTITY material_anisotropic
SUBTYPE OF (material);
    properties : LIST [2:?] OF UNIQUE material_representation;
    material_location : LIST [2:?] OF UNIQUE placement;
WHERE
    WRM41 : SIZEOF(properties) = SIZEOF(material_location);
END_ENTITY;

```

```

ENTITY material_constituent
SUBTYPE OF (material);
    parent_material : material;
    constituent_amount : ratio_measure_with_unit;
    composition_basis : OPTIONAL label;
    class : label;
    determination_method : OPTIONAL text;
WHERE
    WRM19 : parent_material :<>: (SELF);
END_ENTITY;

```

```

ENTITY material_elasticity
SUBTYPE OF (material_representation_item);
    poisson_ratio : OPTIONAL REAL;
    young_modulus : OPTIONAL pressure_measure;
    shear_modulus : OPTIONAL pressure_measure;
    secant_modulus : OPTIONAL pressure_measure;
WHERE
    WRM20 : EXISTS (poisson_ratio) OR EXISTS (young_modulus)
            OR EXISTS (shear_modulus) OR EXISTS (secant_modulus);
END_ENTITY;

```

```

ENTITY material_hardness
SUBTYPE OF (material_representation_item);
    hardness_values : context_dependent_measure;
END_ENTITY;

```

```

ENTITY material_isotropic
SUBTYPE OF (material);
    definition : material_representation;
END_ENTITY;

```

```

ENTITY material_mass_density
SUBTYPE OF (material_representation_item);
    mass_density : context_dependent_measure;
END_ENTITY;

```

```

ENTITY material_orthotropic
SUBTYPE OF (material);

```

```

    in_plane_properties : material_representation;
    out_of_plane_properties : material_representation;
WHERE
    WRM40 : in_plane_properties :<>: out_of_plane_properties;
END_ENTITY;

```

```

ENTITY material_property_context
ABSTRACT SUPERTYPE OF (
    material_property_context_dimensional ANDOR
    material_property_context_loading ANDOR
    material_property_context_strain ANDOR
    material_property_context_stress ANDOR
    material_property_context_temperature)
SUBTYPE OF (representation_context);
END_ENTITY;

```

```

ENTITY material_property_context_dimensional
SUBTYPE OF (material_property_context);
    lower_value_for_dimension : length_measure;
    upper_value_for_dimension : length_measure;
END_ENTITY;

```

```

ENTITY material_property_context_loading
SUBTYPE OF (material_property_context);
    loading : loading_status;
END_ENTITY;

```

```

ENTITY material_property_context_strain
SUBTYPE OF (material_property_context);
    lower_value_for_strain : ratio_measure;
    upper_value_for_strain : ratio_measure;
END_ENTITY;

```

```

ENTITY material_property_context_stress
SUBTYPE OF (material_property_context);
    lower_value_for_stress : pressure_measure;
    upper_value_for_stress : pressure_measure;
END_ENTITY;

```

```

ENTITY material_property_context_temperature
SUBTYPE OF (material_property_context);
    temperature_lower_bound : thermodynamic_temperature_measure;
    temperature_upper_bound : thermodynamic_temperature_measure;
END_ENTITY;

```

```

ENTITY material_representation
SUBTYPE OF (representation);

```

WHERE

WRM21 : ('STRUCTURAL_FRAME_SCHEMA.MATERIAL_PROPERTY_CONTEXT' IN
 TYPEOF (SELFrepresentation.context_of_items)) AND
 ('STRUCTURAL_FRAME_SCHEMA.GLOBAL_UNIT_ASSIGNED_CONTEXT' IN
 TYPEOF (SELFrepresentation.context_of_items));

WRM22 : SIZEOF(QUERY(temp <* SELFrepresentation.items |
 ('STRUCTURAL_FRAME_SCHEMA.MATERIAL_REPRESENTATION_ITEM') IN
 TYPEOF(temp))) > 0;

END_ENTITY;

ENTITY material_representation_item

SUPERTYPE OF (ONEOF

(material_elasticity,
 material_mass_density,
 material_thermal_expansion,
 material_strength,
 material_hardness,
 material_toughness))

SUBTYPE OF (representation_item);

WHERE

WRM23 : SIZEOF (QUERY (tmp_rep <* using_representations (SELF) |
 NOT ('STRUCTURAL_FRAME_SCHEMA.MATERIAL_PROPERTY_CONTEXT' IN
 TYPEOF (tmp_rep.context_of_items)))) = 0;

END_ENTITY;

ENTITY material_strength

SUBTYPE OF (material_representation_item);

material_strength_value : pressure_measure;

END_ENTITY;

ENTITY material_thermal_expansion

SUBTYPE OF (material_representation_item);

thermal_expansion_coeff : context_dependent_measure;

END_ENTITY;

ENTITY material_toughness

SUBTYPE OF (material_representation_item);

toughness_values : context_dependent_measure;

END_ENTITY;

ENTITY measure_qualification;

name : label;

description : text;

qualified_measure : measure_with_unit;

qualifiers : SET [1:?] OF value_qualifier;

WHERE

WRM24 : SIZEOF(QUERY(temp <* qualifiers

```

    | 'STRUCTURAL_FRAME_SCHEMA.PRECISION_QUALIFIER' IN
      TYPEOF(temp))) < 2;
END_ENTITY; (* STEP Part 45 (unchanged in TC1) *)

```

```

ENTITY measure_with_unit
SUPERTYPE OF (ONEOF
  (length_measure_with_unit,
   mass_measure_with_unit,
   time_measure_with_unit,
   thermodynamic_temperature_measure_with_unit,
   plane_angle_measure_with_unit,
   solid_angle_measure_with_unit,
   area_measure_with_unit,
   volume_measure_with_unit,
   ratio_measure_with_unit,
   force_measure_with_unit,
   frequency_measure_with_unit,
   pressure_measure_with_unit,
   positive_length_measure_with_unit,
   derived_measure_with_unit,
   uncertainty_measure_with_unit));
  value_component : measure_value;
  unit_component : unit;
WHERE
  WRM25 : valid_units (SELF);
END_ENTITY; (* STEP Part 41 expanded (2nd edition unchanged) *)

```

```

ENTITY media_content
SUBTYPE OF (group_assignment);
WHERE
  WRM26 : 'STRUCTURAL_FRAME_SCHEMA.MEDIA_FILE' IN TYPEOF
    (SELF\group_assignment.assigned_group);
END_ENTITY;

```

(* **New for LPM/6** - see Issue 97 *)

```

ENTITY media_content_drawing
SUBTYPE OF (media_content);
WHERE
  WRM42 : 'STRUCTURAL_FRAME_SCHEMA.MEDIA_FILE_DRAWING' IN TYPEOF
    (SELF\group_assignment.assigned_group);
END_ENTITY;

```

(* **Modified for LPM/6** *)

```

ENTITY media_file
SUPERTYPE OF (ONEOF(step_file, media_file_drawing, media_file_cnc))
SUBTYPE OF (group);
  file_source : label;

```

```

file_format : label;
file_date : date_and_time;
media_type : label;
author : LIST [1:?] OF person_and_organization;
owner : LIST [0:?] OF person_and_organization;
END_ENTITY;

```

(* New for LPM/6 - see Issue 97 *)

```

ENTITY media_file_cnc
SUBTYPE OF (media_file);
  cnc_data_format : label;
DERIVE
  cnc_file_title : label := SELF\group.group_name;
  created_by : person := SELF\media_file.author[1].the_person;
  detail_company : organization := SELF\media_file.author[1].the_organization;
  creation_date : date_and_time := SELF\media_file.file_date;
  cnc_filename : label := SELF\media_file.file_source;
END_ENTITY;

```

(* New for LPM/6 - see Issue 97 *)

```

ENTITY media_file_drawing
SUBTYPE OF (media_file);
  drawing_number : OPTIONAL label;
  drawing_type : drawing_class;
  drawing_size : OPTIONAL label;
  current_revision_mark : OPTIONAL label;
  current_revision_by : OPTIONAL person_and_organization;
  current_revision_date : OPTIONAL date_and_time;
  current_revision_note : OPTIONAL text;
DERIVE
  drawing_title : label := SELF\group.group_name;
  drawn_by : person := SELF\media_file.author[1].the_person;
  detail_company : organization := SELF\media_file.author[1].the_organization;
  creation_date : date_and_time := SELF\media_file.file_date;
  drawing_filename : label := SELF\media_file.file_source;
END_ENTITY;

```

```

ENTITY modulus_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
  WRM27 : 'STRUCTURAL_FRAME_SCHEMA.MODULUS_UNIT' IN
    TYPEOF (SELF\measure_with_unit.unit_component);
  WRM28 : 'STRUCTURAL_FRAME_SCHEMA.MODULUS_MEASURE' IN
    TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```



```

ENTITY modulus_unit
SUBTYPE OF (derived_unit);
WHERE
    WRM29 : SIZEOF(SELF\derived_unit.elements) = 1;
    WRM30 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 3.0);
END_ENTITY;

```

```

ENTITY moment_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
    WRM31 : 'STRUCTURAL_FRAME_SCHEMA.MOMENT_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
    WRM32 : 'STRUCTURAL_FRAME_SCHEMA.MOMENT_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```

```

ENTITY moment_unit
SUBTYPE OF (derived_unit);
WHERE
    WRM33 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRM34 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRM35 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = 1.0);
END_ENTITY;

```

```

ENTITY move
SUBTYPE OF (structural_frame_process);
    initial_location : placement;
    final_location : placement;
    path : OPTIONAL curve;
END_ENTITY;

```

```

(* New for LPM/6 *)
ENTITY name_attribute;
    attribute_value : label;
    named_item : name_attribute_select;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

```

ENTITY named_unit
    SUPERTYPE OF ((ONEOF
        (length_unit,

```

```

        mass_unit,
        time_unit,
        thermodynamic_temperature_unit,
        plane_angle_unit,
        solid_angle_unit,
        area_unit,
        volume_unit,
        ratio_unit,
        force_unit,
        frequency_unit,
        pressure_unit)) ANDOR (ONEOF
        (si_unit,
        conversion_based_unit,
        context_dependent_unit)));
    dimensions : dimensional_exponents;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY node;
    node_name : label;
    node_coords : point;
    restraints : OPTIONAL boundary_condition;
    parent_model : analysis_model;
UNIQUE
    URN1 : node_name, parent_model;
    URN2 : node_coords, parent_model;
WHERE
    WRN1 : node_coords.dim = parent_model.coordinate_space_dimension;
END_ENTITY;

```

```

ENTITY node_dependency;
    master_node : node;
    slave_node : node;
    dependency_description : OPTIONAL text;
WHERE
    WRN2 : master_node :<>: slave_node;
END_ENTITY;

```

```

(* New for LPM/6 *)
ENTITY object_role;
    name : label;
    description : OPTIONAL text;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

```

ENTITY offset_curve_2d
SUBTYPE OF (curve);
    basis_curve: curve;

```

```

    distance : length_measure;
    self_intersect : LOGICAL;
WHERE
    WRO1 : basis_curve.dim = 2;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY offset_curve_3d
SUBTYPE OF (curve);
    basis_curve: curve;
    distance : length_measure;
    self_intersect : LOGICAL;
    ref_direction : direction;
WHERE
    WRO2 : (basis_curve.dim = 3) AND (ref_direction.dim = 3);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY offset_surface
SUBTYPE OF (surface);
    basis_surface : surface;
    distance : length_measure;
    self_intersect : LOGICAL;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY open_path
SUBTYPE OF (path);
DERIVE
    ne : INTEGER := SIZEOF(SELF\path.edge_list);
WHERE
    WRO3 : (SELF\path.edge_list[1].edge_element.edge_start) :<>:
        (SELF\path.edge_list[ne].edge_element.edge_end);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY open_shell
SUBTYPE OF (connected_face_set);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY organization;
    id : OPTIONAL identifier;
    name : label;
    description : text;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY organizational_address
SUBTYPE OF (address);
    organizations : SET [1:?] OF organization;
    description : text;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

(* Modified for LPM/6 *)

```
ENTITY organization_relationship
SUPERTYPE OF (organization_relationship_contractual);
    name : label;
    description : OPTIONAL text;
    relating_organization : organization;
    related_organization : organization;
END_ENTITY; (* STEP Part 41 2nd edition expanded *)
```

```
ENTITY organization_relationship_contractual
SUBTYPE OF (organization_relationship);
    assigned_contract : contract;
    effective_start_date : date_and_time;
    effective_end_date : OPTIONAL date_and_time;
UNIQUE
    URO1 : SELF.organization_relationship.name,
        SELF.organization_relationship.relatating_organization,
        SELF.organization_relationship.related_organization;
WHERE
    WRO4 : acyclic_organization_relationship(SELF,
        [SELF.organization_relationship.related_organization],
        'STRUCTURAL_FRAME_SCHEMA.
        ORGANIZATION_RELATIONSHIP.RELATED_ORGANIZATION');
END_ENTITY;
```

```
ENTITY oriented_closed_shell
SUBTYPE OF (closed_shell);
    closed_shell_element : closed_shell;
    orientation : BOOLEAN;
DERIVE
    SELF.connected_face_set.cfs_faces : SET [1:?] OF face :=
        conditional_reverse(SELF.orientation, SELF.closed_shell_element.cfs_faces);
WHERE
    WRO5 : NOT ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_CLOSED_SHELL'
        IN TYPEOF (SELF.closed_shell_element));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```
ENTITY oriented_edge
SUBTYPE OF (edge);
    edge_element : edge;
    orientation : BOOLEAN;
DERIVE
    SELF.edge.edge_start : vertex := boolean_choose
        (SELF.orientation, SELF.edge_element.edge_start, SELF.edge_element.edge_end);
    SELF.edge.edge_end : vertex := boolean_choose
        (SELF.orientation, SELF.edge_element.edge_end, SELF.edge_element.edge_start);
WHERE
```

```

WRO6 : NOT ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_EDGE'
  IN TYPEOF (SELF.edge_element));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY oriented_face
SUBTYPE OF (face);
  face_element : face;
  orientation : BOOLEAN;
DERIVE
  SELF\face.bound : SET [1:?] OF face_bound := conditional_reverse
    (SELF.orientation,SELF.face_element.bound);
WHERE
  WRO7 : NOT ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_FACE'
    IN TYPEOF (SELF.face_element));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY oriented_open_shell
SUBTYPE OF (open_shell);
  open_shell_element : open_shell;
  orientation : BOOLEAN;
DERIVE
  SELF\connected_face_set.cfs_faces : SET [1:?] OF face := conditional_reverse
    (SELF.orientation, SELF.open_shell_element.cfs_faces);
WHERE
  WRO8 : NOT ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_OPEN_SHELL'
    IN TYPEOF (SELF.open_shell_element));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* **Modified for LPM/6** *)

```

ENTITY oriented_path
SUBTYPE OF (path);
  path_element : path;
  orientation : BOOLEAN;
DERIVE
  SELF\path.edge_list: LIST [1:?] OF UNIQUE oriented_edge := conditional_reverse
    (SELF.orientation, SELF.path_element.edge_list);
WHERE
  WRO9 : NOT ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_PATH'
    IN TYPEOF (SELF.path_element));
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

(* **New for LPM/6** *)

```

ENTITY oriented_surface
SUBTYPE OF (surface);
  orientation : BOOLEAN;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```
ENTITY outer_boundary_curve
SUBTYPE OF (boundary_curve);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```
ENTITY parabola
SUBTYPE OF (conic);
    focal_dist : length_measure;
WHERE
    WRP1 : focal_dist <> 0.0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```
ENTITY parametric_representation_context
SUBTYPE OF (representation_context);
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)
```

(* [Modified for LPM/6](#) *)

```
ENTITY part
SUPERTYPE OF (ONEOF
    (part_prismatic,
    part_sheet,
    part_complex) ANDOR
    part_derived)
SUBTYPE OF (structural_frame_product);
    fabrication_method : fabrication_type;
    manufacturers_ref : OPTIONAL text;
DERIVE
    part_number : INTEGER := SELF\structural_frame_item.item_number;
    part_name : label := SELF\structural_frame_item.item_name;
    design_uses : SET [0:?] OF design_part := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.DESIGN_PART.DESIGN_PART_SPEC'));
    physical_uses : SET [0:?] OF located_part := bag_to_set (USEDIN(SELF,
        'STRUCTURAL_FRAME_SCHEMA.LOCATED_PART.DESCRPTIVE_PART'));
UNIQUE
    URP6 : part_number,part_name;
END_ENTITY;
```

```
ENTITY part_complex
SUBTYPE OF (part);
    part_shape : shape_representation_with_units;
END_ENTITY;
```

```
ENTITY part_derived
SUBTYPE OF (part);
    made_from : part;
    involved_processes : SET [0:?] OF structural_frame_process;
WHERE
```

```

    WRP2 : made_from :<>: (SELF);
END_ENTITY;

```

```

ENTITY part_map;
    represented_part : part;
    representing_elements : SET [1:?] OF element;
END_ENTITY;

```

```

ENTITY part_prismatic
ABSTRACT SUPERTYPE OF (ONEOF(part_prismatic_complex, part_prismatic_simple))
SUBTYPE OF (part);
END_ENTITY;

```

```

ENTITY part_prismatic_complex
SUPERTYPE OF (part_prismatic_complex_tapered)
SUBTYPE OF (part_prismatic);
    cross_sections : LIST [2:?] OF section_profile;
    points_defining_part_axis : LIST [2:?] OF UNIQUE point_on_curve;
    section_orientations : LIST [2:?] OF orientation_select;
DERIVE
    number_of_sections : INTEGER := SIZEOF(cross_sections);
    curve_defining_part : curve := points_defining_part_axis[1]\point_on_curve.basis_curve;
WHERE
    WRP3 : ( (SIZEOF (points_defining_part_axis) = number_of_sections) AND
        (SIZEOF (section_orientations) = number_of_sections) );
    WRP4 : SIZEOF(QUERY(temp <* points_defining_part_axis |
        (temp\point_on_curve.basis_curve) :<>: curve_defining_part)) = 0;
END_ENTITY;

```

```

ENTITY part_prismatic_complex_tapered
SUBTYPE OF (part_prismatic_complex);
    taper_description : OPTIONAL text;
    absolute_taper_1 : OPTIONAL length_measure_with_unit;
    absolute_taper_2 : OPTIONAL length_measure_with_unit;
    relative_taper_1 : OPTIONAL ratio_measure_with_unit;
    relative_taper_2 : OPTIONAL ratio_measure_with_unit;
WHERE
    WRP5 : EXISTS (absolute_taper_1) OR EXISTS (absolute_taper_2) OR
        EXISTS (relative_taper_1) OR EXISTS (relative_taper_2);
END_ENTITY;

```

```

ENTITY part_prismatic_simple
SUPERTYPE OF (part_prismatic_simple_cambered ANDOR
    part_prismatic_simple_castellated ANDOR
    part_prismatic_simple_curved)
SUBTYPE OF (part_prismatic);
    profile : section_profile;

```

```

    cut_length : positive_length_measure_with_unit;
    stock_length : OPTIONAL positive_length_measure_with_unit;
    x_offset : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY part_prismatic_simple_cambered
SUPERTYPE OF (part_prismatic_simple_cambered_absolute ANDOR
    part_prismatic_simple_cambered_relative)
SUBTYPE OF (part_prismatic_simple);
    camber_description : OPTIONAL text;
END_ENTITY;

```

```

ENTITY part_prismatic_simple_cambered_absolute
SUBTYPE OF (part_prismatic_simple_cambered);
    absolute_offset_position : positive_length_measure_with_unit;
    absolute_offset_y : length_measure_with_unit;
    absolute_offset_z : length_measure_with_unit;
END_ENTITY;

```

(* [Modified for LPM/6](#) *)

```

ENTITY part_prismatic_simple_cambered_relative
SUBTYPE OF (part_prismatic_simple_cambered);
    relative_offset_position : ratio_measure_with_unit;
    relative_offset_y : ratio_measure_with_unit;
    relative_offset_z : ratio_measure_with_unit;
END_ENTITY;

```

```

ENTITY part_prismatic_simple_castellated
SUBTYPE OF (part_prismatic_simple);
    part_castellation_type : castellation_type;
    end_post_width_1 : positive_length_measure_with_unit;
    end_post_width_2 : OPTIONAL positive_length_measure_with_unit;
    castellation_spacing : OPTIONAL positive_length_measure_with_unit;
    castellation_height : OPTIONAL positive_length_measure_with_unit;
    castellation_width : OPTIONAL positive_length_measure_with_unit;
    castellation_depth : OPTIONAL positive_length_measure_with_unit;
END_ENTITY;

```

```

ENTITY part_prismatic_simple_curved
SUBTYPE OF (part_prismatic_simple);
    axis_definition : curve;
END_ENTITY;

```

```

ENTITY part_sheet
SUPERTYPE OF (part_sheet_bounded ANDOR part_sheet_profiled)
SUBTYPE OF (part);
    sheet_thickness : positive_length_measure_with_unit;

```


END_ENTITY;

ENTITY part_sheet_bounded
 ABSTRACT SUPERTYPE OF (ONEOF
 (part_sheet_bounded_complex,
 part_sheet_bounded_simple))
 SUBTYPE OF (part_sheet);
 END_ENTITY;

ENTITY part_sheet_bounded_complex
 SUBTYPE OF (part_sheet_bounded);
 sheet_boundary : bounded_surface;
 END_ENTITY;

ENTITY part_sheet_bounded_simple
 SUBTYPE OF (part_sheet_bounded);
 cut_y_dimension : positive_length_measure_with_unit;
 cut_z_dimension : positive_length_measure_with_unit;
 stock_y_dimension : OPTIONAL positive_length_measure_with_unit;
 stock_z_dimension : OPTIONAL positive_length_measure_with_unit;
 y_offset : OPTIONAL positive_length_measure_with_unit;
 z_offset : OPTIONAL positive_length_measure_with_unit;
 END_ENTITY;

ENTITY part_sheet_profiled
 SUBTYPE OF (part_sheet);
 sheet_profile : curve;
 profile_properties : OPTIONAL section_properties;
 END_ENTITY;

(* [Modified for LPM/6](#) *)

ENTITY path
 SUPERTYPE OF (ONEOF(oriented_path, open_path, edge_loop))
 SUBTYPE OF (topological_representation_item);
 edge_list : LIST [1:?] OF UNIQUE oriented_edge;
 WHERE
 WRP6 : path_head_to_tail(SELF);
 END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

ENTITY pcurve
 SUBTYPE OF (curve);
 basis_surface : surface;
 reference_to_curve : definitional_representation;
 WHERE
 WRP7 : SIZEOF(reference_to_curve\representation.items) = 1;
 WRP8 : 'STRUCTURAL_FRAME_SCHEMA.CURVE' IN TYPEOF
 (reference_to_curve\representation.items[1]);

```

WRP9 : reference_to_curve\representation.
      items[1]\geometric_representation_item.dim = 2;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY person;
  id : identifier;
  last_name : OPTIONAL label;
  first_name : OPTIONAL label;
  middle_names : OPTIONAL LIST [1:?] OF label;
  prefix_titles : OPTIONAL LIST [1:?] OF label;
  suffix_titles : OPTIONAL LIST [1:?] OF label;
UNIQUE
  URP1 : id;
WHERE
  WRP10 : EXISTS (last_name) OR EXISTS (first_name);
END_ENTITY; (* based on STEP Part 41 2nd edition (UR added) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY person_and_organization;
  the_person : person;
  the_organization : organization;
DERIVE
  name : label := get_name_value (SELF);
  description : text := get_description_value (SELF);
WHERE
  WRP27 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
    'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
  WRP28 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
    'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY person_and_organization_role;
  name : label;
DERIVE
  description : text := get_description_value (SELF);
WHERE
  WRP29 : SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
    'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; (* STEP Part 41 2nd edition *)

```

```

ENTITY personal_address
SUBTYPE OF (address);
  people : SET [1:?] OF person;
  description : text;
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY physical_action
SUPERTYPE OF (ONEOF
    (physical_action_permanent,
    physical_action_variable,
    physical_action_accidental,
    physical_action_seismic));
action_nature : static_or_dynamic;
action_spatial_variation : spatial_variation;
action_type : direct_or_indirect_action;
basic_magnitude : OPTIONAL measure_with_unit;
derived_magnitude : OPTIONAL measure_with_unit;
derivation_factors : LIST [0:?] OF REAL;
derivation_factor_labels : LIST [0:?] OF label;
WHERE
    WRP11 : SIZEOF (derivation_factors) = SIZEOF (derivation_factor_labels);
END_ENTITY;

```

```

ENTITY physical_action_accidental
SUBTYPE OF (physical_action);
    action_source : action_source_accidental;
END_ENTITY;

```

```

ENTITY physical_action_permanent
SUBTYPE OF (physical_action);
    action_source : action_source_permanent;
END_ENTITY;

```

```

ENTITY physical_action_seismic
SUBTYPE OF (physical_action);
END_ENTITY;

```

```

ENTITY physical_action_variable
ABSTRACT SUPERTYPE OF (ONEOF
    (physical_action_variable_long_term,
    physical_action_variable_transient,
    physical_action_variable_short_term))
SUBTYPE OF (physical_action);
END_ENTITY;

```

```

ENTITY physical_action_variable_long_term
SUBTYPE OF (physical_action_variable);
    action_source : action_source_variable_long_term;
END_ENTITY;

```

```

ENTITY physical_action_variable_short_term
SUBTYPE OF (physical_action_variable);
    action_source : action_source_variable_short_term;

```

END_ENTITY;

ENTITY physical_action_variable_transient
 SUBTYPE OF (physical_action_variable);
 action_source : action_source_variable_transient;
 END_ENTITY;

ENTITY placement
 SUPERTYPE OF (ONEOF
 (axis2_placement_2d,
 axis2_placement_3d,
 axis1_placement))
 SUBTYPE OF (geometric_representation_item);
 location : cartesian_point;
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY plane
 SUBTYPE OF (elementary_surface);
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY plane_angle_measure_with_unit
 SUBTYPE OF (measure_with_unit);
 WHERE
 WRP12 : 'STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN
 TYPEOF (SELF\measure_with_unit.unit_component);
 WRP13 : 'STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_MEASURE' IN
 TYPEOF (SELF\measure_with_unit.value_component);
 END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

ENTITY plane_angle_unit
 SUBTYPE OF (named_unit);
 WHERE
 WRP14 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND
 (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
 (SELF\named_unit.dimensions.time_exponent = 0.0) AND
 (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
 (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
 (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
 (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
 END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

(* [Modified for LPM/6](#) *)

ENTITY point
 SUPERTYPE OF (ONEOF
 (cartesian_point,
 point_in_volume,
 point_on_curve,

```

    point_on_surface,
    point_replica,
    degenerate_pcurve))
SUBTYPE OF (geometric_representation_item);
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY point_in_volume
SUBTYPE OF (point);
    basis_volume : volume;
    point_parameter_u : parameter_value;
    point_parameter_v : parameter_value;
    point_parameter_w : parameter_value;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY point_on_curve
SUBTYPE OF (point);
    basis_curve : curve;
    point_parameter : parameter_value;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY point_on_surface
SUBTYPE OF (point);
    basis_surface : surface;
    point_parameter_u : parameter_value;
    point_parameter_v : parameter_value;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY point_replica
SUBTYPE OF (point);
    parent_pt : point;
    transformation : cartesian_transformation_operator;
WHERE
    WRP15 : transformation.dim = parent_pt.dim;
    WRP16 : acyclic_point_replica (SELF,parent_pt);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY polar_point
SUBTYPE OF (cartesian_point);
    r : length_measure;
    theta : plane_angle_measure;
DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure :=
        [r*cos(theta), r*sin(theta)];
WHERE
    WRP30 : r >= 0.0;

```

END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

(* **Modified for LPM/6** *)

ENTITY poly_loop

SUBTYPE OF (loop, geometric_representation_item);

 polygon : LIST [3:?] OF UNIQUE cartesian_point;

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

(* **New for LPM/6** *)

ENTITY polygonal_area

SUBTYPE OF (primitive_2d);

 bounds : LIST [3:?] OF UNIQUE cartesian_point;

END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

ENTITY polyline

SUBTYPE OF (bounded_curve);

 points : LIST [2:?] OF cartesian_point;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY positive_length_measure_with_unit

SUBTYPE OF (measure_with_unit);

WHERE

 WRP17 : 'STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN

 TYPEOF (SELF\measure_with_unit.unit_component);

 WRP18 : 'STRUCTURAL_FRAME_SCHEMA.POSITIVE_LENGTH_MEASURE' IN

 TYPEOF (SELF\measure_with_unit.value_component);

END_ENTITY;

ENTITY precision_qualifier;

 precision_value : INTEGER;

END_ENTITY; (* STEP Part 45 (unchanged in TC1) *)

ENTITY pressure_measure_with_unit

SUBTYPE OF (measure_with_unit);

WHERE

 WRP19 : 'STRUCTURAL_FRAME_SCHEMA.PRESSURE_UNIT' IN

 TYPEOF (SELF\measure_with_unit.unit_component);

 WRP20 : 'STRUCTURAL_FRAME_SCHEMA.PRESSURE_MEASURE' IN

 TYPEOF (SELF\measure_with_unit.value_component);

END_ENTITY;

ENTITY pressure_unit

SUBTYPE OF (named_unit);

WHERE

 WRP21 : (SELF\named_unit.dimensions.length_exponent = -1.0) AND

 (SELF\named_unit.dimensions.mass_exponent = 1.0) AND

 (SELF\named_unit.dimensions.time_exponent = -2.0) AND

```

        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;

```

(* **New for LPM/6** *)

```

ENTITY primitive_2d
SUPERTYPE OF (ONEOF (
    circular_area,
    elliptic_area,
    rectangular_area,
    polygonal_area))
SUBTYPE OF (geometric_representation_item);
WHERE
    WR1 : SELF\geometric_representation_item.dim = 2;
END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

```

(* **Modified for LPM/6** *)

```

ENTITY procure
SUBTYPE OF (structural_frame_process);
    vendors : person_and_organization;
    purchaser : person_and_organization;
    purchased_products : LIST [1:?] OF product_item_select;
    sales_contract : contract;
    delivery_dates : LIST [1:?] OF calendar_date;
INVERSE
    prices : SET [1:?] OF structural_frame_item_priced FOR priced_item;
WHERE
    WRP23 : SIZEOF (delivery_dates) = SIZEOF (purchased_products);
END_ENTITY;

```

```

ENTITY project
SUBTYPE OF (structural_frame_item);
UNIQUE
    URP2 : SELF\structural_frame_item.item_number,
          SELF\structural_frame_item.item_name;
END_ENTITY;

```

(* **New for LPM/6** - See Issues 90 *)

```

ENTITY project_data_group
SUBTYPE OF (group_assignment);
    parent_project : project;
END_ENTITY;

```

```

ENTITY project_organization;
    project_participant : person_and_organization;

```

```

    related_project : project;
    role : person_and_organization_role;
END_ENTITY;

```

```

ENTITY project_plan
SUBTYPE OF (structural_frame_item);
    project_plan_author : project_organization;
    project_plan_date : date_and_time;
    related_project : project;
INVERSE
    items : SET [1:?] OF project_plan_item FOR item_for_plan;
UNIQUE
    URP3 : SELF\structural_frame_item.item_number, related_project;
WHERE
    WRP25 : project_plan_author.related_project :=: related_project;
END_ENTITY;

```

```

ENTITY project_plan_item
SUPERTYPE OF (ONEOF(project_process_item))
SUBTYPE OF (structural_frame_item);
    item_for_plan : project_plan;
    start_date : date_and_time;
    end_date : date_and_time;
    item_duration : time_measure_with_unit;
    actors : OPTIONAL SET [1:?] OF project_organization;
    sequence_number : OPTIONAL INTEGER;
    item_status : OPTIONAL label;
UNIQUE
    URP4 : SELF\structural_frame_item.item_number, item_for_plan;
WHERE
    WRP26 : NOT (EXISTS(actors) AND
        (SIZEOF(QUERY(the_project <= actors | the_project.related_project :<>:
            (item_for_plan.related_project)) ) <> 0));
END_ENTITY;

```

```

ENTITY project_plan_item_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    related_plan_item : project_plan_item;
    relating_plan_item : project_plan_item;
WHERE
    WRP24 : related_plan_item :<>: relating_plan_item;
END_ENTITY;

```

(* [Modified for LPM/6](#) - See Issue 15 *)

```

ENTITY project_process_item
SUBTYPE OF (project_plan_item);

```



```

    scheduled_process : structural_frame_process;
    resulting_product : SET [0:?] OF product_item_select;
    processed_products : SET [1:?] OF product_item_select;
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY pyramid_volume
SUBTYPE OF (volume);
    position : axis2_placement_3d;
    xlength : positive_length_measure;
    ylength : positive_length_measure;
    height : positive_length_measure;
END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

```

```

ENTITY qualitative_uncertainty
SUBTYPE OF (uncertainty_qualifier);
    uncertainty_value : text;
END_ENTITY; (* STEP Part 45 (unchanged in TC1) *)

```

```

ENTITY quasi_uniform_curve
SUBTYPE OF (b_spline_curve);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY quasi_uniform_surface
SUBTYPE OF (b_spline_surface);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY quasi_uniform_volume
    SUBTYPE OF (b_spline_volume);
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY ratio_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRR1 : 'STRUCTURAL_FRAME_SCHEMA.RATIO_UNIT' IN
TYPEOF (SELF\measure_with_unit.unit_component);
    WRR2 : 'STRUCTURAL_FRAME_SCHEMA.RATIO_MEASURE' IN
    TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

```

```

ENTITY ratio_unit
SUBTYPE OF (named_unit);
WHERE
    WRR3 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND

```

```

        (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
        (SELF\named_unit.dimensions.time_exponent = 0.0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY rational_b_spline_curve
SUBTYPE OF (b_spline_curve);
    weights_data : LIST [2:?] OF REAL;
DERIVE
    weights : ARRAY [0:upper_index_on_control_points] OF REAL := list_to_array
        (weights_data, 0, upper_index_on_control_points);
WHERE
    WRR4 : SIZEOF(weights_data) = SIZEOF(SELF\b_spline_curve.control_points_list);
    WRR5 : curve_weights_positive(SELF);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY rational_b_spline_surface
SUBTYPE OF (b_spline_surface);
    weights_data : LIST [2:?] OF LIST [2:?] OF REAL;
DERIVE
    weights : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF REAL := make_array_of_array
        (weights_data, 0, u_upper, 0, v_upper);
WHERE
    WRR6 : (SIZEOF(weights_data) = SIZEOF(SELF\b_spline_surface.control_points_list))
        AND (SIZEOF(weights_data[1]) =
            SIZEOF(SELF\b_spline_surface.control_points_list[1]));
    WRR7 : surface_weights_positive(SELF);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY rational_b_spline_volume
SUBTYPE OF (b_spline_volume);
    weights_data : LIST [2:?] OF LIST [2:?] OF LIST [2:?] OF REAL;
DERIVE
    weights : ARRAY [0:u_upper] OF
        ARRAY [0:v_upper] OF
            ARRAY [0:w_upper] OF REAL
        := make_array_of_array_of_array
            (weights_data,0,u_upper,0,v_upper,0,w_upper);
WHERE
    WRR53 : (SIZEOF(weights_data) = SIZEOF(SELF\b_spline_volume.control_points_list))
        AND (SIZEOF(weights_data[1]) =
            SIZEOF(SELF\b_spline_volume.control_points_list[1]))
        AND (SIZEOF(weights_data[1][1]) =

```

```

        SIZEOF(SELF\b_spline_volume.control_points_list[1][1]));
    WRR54 : volume_weights_positive(SELF);
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

ENTITY reaction

ABSTRACT SUPERTYPE OF (ONEOF

```

    (reaction_force,
     reaction_displacement,
     reaction_velocity,
     reaction_acceleration,
     reaction_dynamic,
     reaction_equilibrium));

```

END_ENTITY;

ENTITY reaction_acceleration

SUBTYPE OF (reaction);

```

    reaction_acceleration_ax : OPTIONAL linear_acceleration_measure_with_unit;
    reaction_acceleration_ay : OPTIONAL linear_acceleration_measure_with_unit;
    reaction_acceleration_az : OPTIONAL linear_acceleration_measure_with_unit;
    reaction_acceleration_arx : OPTIONAL rotational_acceleration_measure_with_unit;
    reaction_acceleration_ary : OPTIONAL rotational_acceleration_measure_with_unit;
    reaction_acceleration_arz : OPTIONAL rotational_acceleration_measure_with_unit;

```

WHERE

```

    WRR8 : EXISTS (reaction_acceleration_ax) OR
           EXISTS (reaction_acceleration_ay) OR
           EXISTS (reaction_acceleration_az) OR
           EXISTS (reaction_acceleration_arx) OR
           EXISTS (reaction_acceleration_ary) OR
           EXISTS (reaction_acceleration_arz);

```

END_ENTITY;

ENTITY reaction_displacement

SUBTYPE OF (reaction);

```

    reaction_displacement_dx : OPTIONAL length_measure_with_unit;
    reaction_displacement_dy : OPTIONAL length_measure_with_unit;
    reaction_displacement_dz : OPTIONAL length_measure_with_unit;
    reaction_displacement_rx : OPTIONAL plane_angle_measure_with_unit;
    reaction_displacement_ry : OPTIONAL plane_angle_measure_with_unit;
    reaction_displacement_rz : OPTIONAL plane_angle_measure_with_unit;

```

WHERE

```

    WRR9 : EXISTS (reaction_displacement_dx) OR
           EXISTS (reaction_displacement_dy) OR
           EXISTS (reaction_displacement_dz) OR
           EXISTS (reaction_displacement_rx) OR
           EXISTS (reaction_displacement_ry) OR
           EXISTS (reaction_displacement_rz);

```

END_ENTITY;

ENTITY reaction_dynamic

SUBTYPE OF (reaction);

phase_angle : OPTIONAL plane_angle_measure_with_unit;

response_amplitude : OPTIONAL length_measure_with_unit;

natural_frequency : OPTIONAL frequency_measure_with_unit;

WHERE

WRR10 : EXISTS (phase_angle) OR

EXISTS (response_amplitude) OR

EXISTS (natural_frequency);

END_ENTITY;

ENTITY reaction_equilibrium

SUBTYPE OF (reaction);

equilibrium_dx : OPTIONAL ratio_measure_with_unit;

equilibrium_dy : OPTIONAL ratio_measure_with_unit;

equilibrium_dz : OPTIONAL ratio_measure_with_unit;

equilibrium_mx : OPTIONAL ratio_measure_with_unit;

equilibrium_my : OPTIONAL ratio_measure_with_unit;

equilibrium_mz : OPTIONAL ratio_measure_with_unit;

WHERE

WRR11 : EXISTS (equilibrium_dx) OR EXISTS (equilibrium_dy) OR EXISTS
(equilibrium_dz);

WRR12 : EXISTS (equilibrium_mx) OR EXISTS (equilibrium_my) OR EXISTS
(equilibrium_mz);

END_ENTITY;

ENTITY reaction_force

SUBTYPE OF (reaction);

reaction_force_fx : OPTIONAL force_measure_with_unit;

reaction_force_fy : OPTIONAL force_measure_with_unit;

reaction_force_fz : OPTIONAL force_measure_with_unit;

reaction_force_mx : OPTIONAL moment_measure_with_unit;

reaction_force_my : OPTIONAL moment_measure_with_unit;

reaction_force_mz : OPTIONAL moment_measure_with_unit;

WHERE

WRR13 : EXISTS (reaction_force_fx) OR

EXISTS (reaction_force_fy) OR

EXISTS (reaction_force_fz) OR

EXISTS (reaction_force_mx) OR

EXISTS (reaction_force_my) OR

EXISTS (reaction_force_mz);

END_ENTITY;

ENTITY reaction_velocity

SUBTYPE OF (reaction);

reaction_velocity_vx : OPTIONAL linear_velocity_measure_with_unit;

```

reaction_velocity_vy : OPTIONAL linear_velocity_measure_with_unit;
reaction_velocity_vz : OPTIONAL linear_velocity_measure_with_unit;
reaction_velocity_vrx : OPTIONAL rotational_velocity_measure_with_unit;
reaction_velocity_vry : OPTIONAL rotational_velocity_measure_with_unit;
reaction_velocity_vrz : OPTIONAL rotational_velocity_measure_with_unit;

```

WHERE

```

WRR14 : EXISTS (reaction_velocity_vx) OR
  EXISTS (reaction_velocity_vy) OR
  EXISTS (reaction_velocity_vz) OR
  EXISTS (reaction_velocity_vrx) OR
  EXISTS (reaction_velocity_vry) OR
  EXISTS (reaction_velocity_vrz);

```

END_ENTITY;

(* New for LPM/6 *)

ENTITY rectangular_area

SUBTYPE OF (primitive_2d);

position: axis2_placement_2d;

x: positive_length_measure;

y: positive_length_measure;

END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

(* Modified for LPM/6 *)

ENTITY rectangular_composite_surface

SUBTYPE OF (bounded_surface);

segments : LIST [1:?] OF LIST [1:?] OF surface_patch;

DERIVE

n_u : INTEGER := SIZEOF(segments);

n_v : INTEGER := SIZEOF(segments[1]);

WHERE

WRR15 : SIZEOF(QUERY (s <* segments | n_v <> SIZEOF (s))) = 0;

WRR16 : constraints_rectangular_composite_surface(SELf);

END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

(* New for LPM/6 *)

ENTITY rectangular_pyramid

SUBTYPE OF (geometric_representation_item);

position : axis2_placement_3d;

xlength : positive_length_measure;

ylength : positive_length_measure;

height : positive_length_measure;

END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

ENTITY rectangular_trimmed_surface

SUBTYPE OF (bounded_surface);

basis_surface : surface;

u1 : parameter_value;

```

u2 : parameter_value;
v1 : parameter_value;
v2 : parameter_value;
usense : BOOLEAN;
vsense : BOOLEAN;
WHERE
WRR17 : u1 <> u2;
WRR18 : v1 <> v2;
WRR19 : (('STRUCTURAL_FRAME_SCHEMA.ELEMENTARY_SURFACE' IN
  TYPEOF(basis_surface)) AND (NOT ('STRUCTURAL_FRAME_SCHEMA.PLANE' IN
  TYPEOF(basis_surface)))) OR
  ('STRUCTURAL_FRAME_SCHEMA.SURFACE_OF_REVOLUTION' IN
  TYPEOF(basis_surface)) OR (usense = (u2 > u1));
WRR20 : (('STRUCTURAL_FRAME_SCHEMA.SPHERICAL_SURFACE' IN
  TYPEOF(basis_surface)) OR
  ('STRUCTURAL_FRAME_SCHEMA.TOROIDAL_SURFACE' IN
  TYPEOF(basis_surface))) OR (vsense = (v2 > v1));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

ENTITY release

ABSTRACT SUPERTYPE OF (ONEOF

(release_logical, release_spring_linear, release_spring_non_linear));

release_name : label;

release_description : OPTIONAL text;

INVERSE

release_for_element_nodes : SET [1:?] OF element_node_connectivity FOR fixity;

END_ENTITY;

ENTITY release_logical

SUBTYPE OF (release);

release_axial_force : LOGICAL;

release_y_force : LOGICAL;

release_z_force : LOGICAL;

release_torsional_moment : LOGICAL;

release_y_bending_moment : LOGICAL;

release_z_bending_moment : LOGICAL;

END_ENTITY;

ENTITY release_spring_linear

SUPERTYPE OF (release_warping)

SUBTYPE OF (release);

release_axial_force : OPTIONAL linear_stiffness_measure_with_unit;

release_y_force : OPTIONAL linear_stiffness_measure_with_unit;

release_z_force : OPTIONAL linear_stiffness_measure_with_unit;

release_torsional_moment : OPTIONAL rotational_stiffness_measure_with_unit;

release_y_bending_moment : OPTIONAL rotational_stiffness_measure_with_unit;

release_z_bending_moment : OPTIONAL rotational_stiffness_measure_with_unit;

WHERE

```

WRR21 : EXISTS (release_axial_force) OR
        EXISTS (release_y_force) OR
        EXISTS (release_z_force) OR
        EXISTS (release_torsional_moment) OR
        EXISTS (release_y_bending_moment) OR
        EXISTS (release_z_bending_moment);
END_ENTITY;

```

```

ENTITY release_spring_non_linear
SUBTYPE OF (release);
    change_values : LIST [2:?] OF measure_with_unit;
    values : LIST [2:?] OF release_spring_linear;
DERIVE
    number_of_values : INTEGER := SIZEOF(change_values);
WHERE
    WRR23 : SIZEOF(values) = SIZEOF(change_values);
END_ENTITY;

```

```

ENTITY release_warping
SUBTYPE OF (release_spring_linear);
    release_warping_moment : rotational_stiffness_measure_with_unit;
END_ENTITY;

```

```

ENTITY reparametrised_composite_curve_segment
SUBTYPE OF (composite_curve_segment);
    param_length : parameter_value;
WHERE
    WRR24 : param_length > 0.0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY representation
SUPERTYPE OF (ONEOF
    (shape_representation, material_representation) ANDOR
    definitional_representation);
    name : label;
    items : SET [1:?] OF representation_item;
    context_of_items : representation_context;
DERIVE
    id : identifier := get_id_value (SELF);
    description : text := get_description_value (SELF);
WHERE
    WRR51: SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
        'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
    WRR52: SIZEOF (USEDIN (SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
        'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; (* STEP Part 43 expanded (modified in 2nd edition) *)

```

```

ENTITY representation_context
SUPERTYPE OF (ONEOF
    (geometric_representation_context,
    parametric_representation_context,
    material_property_context) ANDOR
    global_unit_assigned_context ANDOR
    global_uncertainty_assigned_context);
context_identifier : identifier;
context_type : text;
INVERSE
    representations_in_context : SET [1:?] OF representation FOR context_of_items;
END_ENTITY; (* STEP Part 43 expanded (unchanged in 2nd edition) *)

```

```

ENTITY representation_item
SUPERTYPE OF (geometric_representation_item ANDOR
    topological_representation_item ANDOR
    mapped_item ANDOR
    material_representation_item);
name : label;
WHERE
    WRR25 : SIZEOF(using_representations(SELF)) > 0;
END_ENTITY; (* STEP Part 43 expanded (unchanged in 2nd edition) *)

```

```

ENTITY representation_map;
mapping_origin : representation_item;
mapped_representation : representation;
INVERSE
    map_usage : SET [1:?] OF mapped_item FOR mapping_source;
WHERE
    WRR26 : item_in_context(SELF.mapping_origin,
        SELF.mapped_representation.context_of_items);
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

```

```

ENTITY representation_relationship
SUPERTYPE OF (representation_relationship_with_transformation);
name : label;
description : text;
rep_1 : representation;
rep_2 : representation;
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

```

```

ENTITY representation_relationship_with_transformation
SUBTYPE OF (representation_relationship);
transformation_operator : transformation;
WHERE
    WRR27 : SELF\representation_relationship.rep_1.context_of_items

```



```

        :<>: SELF\representation_relationship.rep_2.context_of_items;
END_ENTITY; (* STEP Part 43 (unchanged in 2nd edition) *)

```

ENTITY resistance

```

ABSTRACT SUPERTYPE OF (ONEOF(resistance_bending, resistance_shear,
    resistance_axial));
    resistance_type : label;
    resistance_description : OPTIONAL text;
    resistance_factor : REAL;
    elastic_or_plastic : elastic_or_plastic_resistance;
    local_or_global : global_or_local_resistance;
INVERSE
    results : SET [1:?] OF design_result FOR design_resistance;
END_ENTITY;

```

ENTITY resistance_axial

```

SUBTYPE OF (resistance);
    tensile_resistance_ptx : OPTIONAL force_measure_with_unit;
    compressive_resistance_pcx : OPTIONAL force_measure_with_unit;
WHERE
    WRR30 : EXISTS (tensile_resistance_ptx) OR EXISTS (compressive_resistance_pcx);
END_ENTITY;

```

ENTITY resistance_bending

```

SUBTYPE OF (resistance);
    torsional_resistance_mx : OPTIONAL moment_measure_with_unit;
    bending_resistance_my : OPTIONAL moment_measure_with_unit;
    bending_resistance_mz : OPTIONAL moment_measure_with_unit;
    buckling_resistance_mb : OPTIONAL moment_measure_with_unit;
WHERE
    WRR28 : EXISTS (torsional_resistance_mx) OR
        EXISTS (bending_resistance_my) OR
        EXISTS (bending_resistance_mz) OR
        EXISTS (buckling_resistance_mb);
END_ENTITY;

```

ENTITY resistance_shear

```

SUBTYPE OF (resistance);
    normal_shear_resistance_pv : OPTIONAL force_measure_with_unit;
    buckling_shear_resistance_pbv : OPTIONAL force_measure_with_unit;
WHERE
    WRR29 : EXISTS(normal_shear_resistance_pv) OR
        EXISTS(buckling_shear_resistance_pbv);
END_ENTITY;

```

ENTITY restraint

```

ABSTRACT SUPERTYPE OF (ONEOF(restraint_logical, restraint_spring));

```

```

    restraint_name : label;
    restraint_description : OPTIONAL text;
    restraint_location : point;
    restrained_mbr : assembly_design_structural_member;
END_ENTITY;

```

```

ENTITY restraint_logical
SUBTYPE OF (restraint);
    restraint_x_displacement : LOGICAL;
    restraint_y_displacement : LOGICAL;
    restraint_z_displacement : LOGICAL;
    restraint_x_rotation : LOGICAL;
    restraint_y_rotation : LOGICAL;
    restraint_z_rotation : LOGICAL;
END_ENTITY;

```

```

ENTITY restraint_spring
SUPERTYPE OF (restraint_warping)
SUBTYPE OF (restraint);
    restraint_x_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    restraint_y_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    restraint_z_displacement : OPTIONAL linear_stiffness_measure_with_unit;
    restraint_x_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
    restraint_y_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
    restraint_z_rotation : OPTIONAL rotational_stiffness_measure_with_unit;
WHERE
    WRR31 : EXISTS (restraint_x_displacement) OR
        EXISTS (restraint_y_displacement) OR
        EXISTS (restraint_z_displacement) OR
        EXISTS (restraint_x_rotation) OR
        EXISTS (restraint_y_rotation) OR
        EXISTS (restraint_z_rotation);
END_ENTITY;

```

```

ENTITY restraint_warping
SUBTYPE OF (restraint_spring);
    restraint_w_rotation : rotational_stiffness_measure_with_unit;
END_ENTITY;

```

(* Modified for LPM/6 *)

```

ENTITY revolved_area_solid
SUBTYPE OF (swept_area_solid);
    axis : axis1_placement;
    angle : plane_angle_measure;
DERIVE
    axis_line : line := representation_item("")||
        geometric_representation_item()||

```

```

        curve()||
        line(axis.location, representation_item(""))||
        geometric_representation_item()||
        vector(axis.z, 1.0));
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

ENTITY revolved_face_solid
SUBTYPE OF (swept_face_solid);
    axis : axis1_placement;
    angle : plane_angle_measure;
DERIVE
    axis_line : line := representation_item("")||
        geometric_representation_item()||
        curve()||
        line(axis.location, representation_item(""))||
        geometric_representation_item()||
        vector(axis.z, 1.0));
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

ENTITY right_angular_wedge
SUBTYPE OF (geometric_representation_item);
    position : axis2_placement_3d;
    x : positive_length_measure;
    y : positive_length_measure;
    z : positive_length_measure;
    ltx : length_measure;
WHERE
    WRR33 : ((0.0 <= ltx) AND (ltx < x));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY right_circular_cone
SUBTYPE OF (geometric_representation_item);
    position : axis1_placement;
    height : positive_length_measure;
    radius : length_measure;
    semi_angle : plane_angle_measure;
WHERE
    WRR34 : radius >= 0.0;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY right_circular_cylinder
SUBTYPE OF (geometric_representation_item);
    position : axis1_placement;
    height : positive_length_measure;
    radius : positive_length_measure;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```
ENTITY role_association;
  role : object_role;
  item_with_role : role_select;
END_ENTITY; (* STEP Part 41 (new for 2nd edition) *)
```

```
ENTITY rotational_acceleration_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
  WRR35 : 'STRUCTURAL_FRAME_SCHEMA.ROTATIONAL_ACCELERATION_UNIT' IN
    TYPEOF (SELF\measure_with_unit.unit_component);
  WRR36 : 'STRUCTURAL_FRAME_SCHEMA.
    ROTATIONAL_ACCELERATION_MEASURE' IN TYPEOF
    (SELF\measure_with_unit.value_component);
END_ENTITY;
```

```
ENTITY rotational_acceleration_unit
SUBTYPE OF (derived_unit);
WHERE
  WRR37 : SIZEOF(SELF\derived_unit.elements) = 2;
  WRR38 : ('STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN
    TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
    AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
  WRR39 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN
    TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
    AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -2.0);
END_ENTITY;
```

```
ENTITY rotational_stiffness_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
  WRR40 : 'STRUCTURAL_FRAME_SCHEMA.ROTATIONAL_STIFFNESS_UNIT' IN
    TYPEOF (SELF\measure_with_unit.unit_component);
  WRR41 : 'STRUCTURAL_FRAME_SCHEMA.ROTATIONAL_STIFFNESS_MEASURE'
    IN TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;
```

```
ENTITY rotational_stiffness_unit
SUBTYPE OF (derived_unit);
WHERE
  WRR42 : SIZEOF(SELF\derived_unit.elements) = 3;
  WRR43 : ('STRUCTURAL_FRAME_SCHEMA.FORCE_UNIT' IN
    TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
    AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
  WRR44 : ('STRUCTURAL_FRAME_SCHEMA.LENGTH_UNIT' IN
    TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
    AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = 1.0);
```

```

WRR45 : ('STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN
        TYPEOF (SELF\derived_unit.elements[3]\derived_unit_element.unit))
        AND (SELF\derived_unit.elements[3]\derived_unit_element.exponent = -1.0);
END_ENTITY;

```

```

ENTITY rotational_velocity_measure_with_unit
SUBTYPE OF (derived_measure_with_unit);
WHERE
    WRR46 : 'STRUCTURAL_FRAME_SCHEMA.ROTATIONAL_VELOCITY_UNIT' IN
            TYPEOF (SELF\measure_with_unit.unit_component);
    WRR47 : 'STRUCTURAL_FRAME_SCHEMA.ROTATIONAL_VELOCITY_MEASURE' IN
            TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY;

```

```

ENTITY rotational_velocity_unit
SUBTYPE OF (derived_unit);
WHERE
    WRR48 : SIZEOF(SELF\derived_unit.elements) = 2;
    WRR49 : ('STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_UNIT' IN
            TYPEOF (SELF\derived_unit.elements[1]\derived_unit_element.unit))
            AND (SELF\derived_unit.elements[1]\derived_unit_element.exponent = 1.0);
    WRR50 : ('STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN
            TYPEOF (SELF\derived_unit.elements[2]\derived_unit_element.unit))
            AND (SELF\derived_unit.elements[2]\derived_unit_element.exponent = -1.0);
END_ENTITY;

```

```

ENTITY seam_curve
SUBTYPE OF (surface_curve);
WHERE
    WRS1 : SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    WRS2 : associated_surface(SELF\surface_curve.associated_geometry[1]) =
            associated_surface(SELF\surface_curve.associated_geometry[2]);
    WRS3 : 'STRUCTURAL_FRAME_SCHEMA.PCURVE' IN
            TYPEOF (SELF\surface_curve.associated_geometry[1]);
    WRS4 : 'STRUCTURAL_FRAME_SCHEMA.PCURVE' IN
            TYPEOF (SELF\surface_curve.associated_geometry[2]);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

ENTITY section_profile
SUPERTYPE OF (ONEOF(section_profile_simple, section_profile_complex))
SUBTYPE OF (structural_frame_item);
    section_classification : OPTIONAL label;
    cardinal_point : cardinal_point_ref;
    mirrored : LOGICAL;
DERIVE
    section_ref : BAG OF identifier := SELF\structural_frame_item.item_ref;

```

END_ENTITY;

ENTITY section_profile_angle

SUBTYPE OF (section_profile_simple);

depth : positive_length_measure_with_unit;

width : OPTIONAL positive_length_measure_with_unit;

thickness : positive_length_measure_with_unit;

internal_fillet_radius : OPTIONAL positive_length_measure_with_unit;

edge_fillet_radius : OPTIONAL positive_length_measure_with_unit;

leg_slope : OPTIONAL ratio_measure_with_unit;

DERIVE

width_value : REAL := NVL(width.value_component, depth.value_component);

WHERE

WRS5 : depth.value_component > thickness.value_component;

END_ENTITY;

ENTITY section_profile_centreline

SUBTYPE OF (section_profile_complex);

centreline : curve;

thickness : positive_length_measure_with_unit;

END_ENTITY;

ENTITY section_profile_channel

SUBTYPE OF (section_profile_simple);

overall_depth : positive_length_measure_with_unit;

flange_width : positive_length_measure_with_unit;

flange_thickness : positive_length_measure_with_unit;

web_thickness : positive_length_measure_with_unit;

root_radius : OPTIONAL positive_length_measure_with_unit;

fillet_radius : OPTIONAL positive_length_measure_with_unit;

flange_slope : OPTIONAL ratio_measure_with_unit;

DERIVE

overall_depth_value : REAL := overall_depth.value_component;

flange_width_value : REAL := flange_width.value_component;

flange_thickness_value : REAL := flange_thickness.value_component;

web_thickness_value : REAL := web_thickness.value_component;

WHERE

WRS6 : flange_thickness_value < (overall_depth_value/2);

WRS7 : web_thickness_value < flange_width_value;

END_ENTITY;

ENTITY section_profile_circle

SUPERTYPE OF (section_profile_circle_hollow)

SUBTYPE OF (section_profile_simple);

external_radius : positive_length_measure_with_unit;

END_ENTITY;

```

ENTITY section_profile_circle_hollow
SUBTYPE OF (section_profile_circle);
    wall_thickness : positive_length_measure_with_unit;
DERIVE
    external_radius_value : REAL :=
        (SELF\section_profile_circle.external_radius.value_component);
    wall_thickness_value : REAL := wall_thickness.value_component;
    inside_diameter : REAL := ((external_radius_value - wall_thickness_value)*2);
WHERE
    WRS8 : wall_thickness_value < external_radius_value;
END_ENTITY;

```

```

ENTITY section_profile_complex
ABSTRACT SUPERTYPE OF (ONEOF
    (section_profile_compound,
    section_profile_edge_defined,
    section_profile_centreline,
    section_profile_derived))
SUBTYPE OF (section_profile);
END_ENTITY;

```

```

ENTITY section_profile_compound
SUBTYPE OF (section_profile_complex);
    component_sections : LIST [2:?] OF section_profile;
    positions : LIST [2:?] OF point;
    orientations : LIST [2:?] OF orientation_select;
DERIVE
    number_of_sections : INTEGER := SIZEOF (component_sections);
WHERE
    WRS9 : SIZEOF (positions) = number_of_sections;
    WRS10 : SIZEOF (orientations) = number_of_sections;
    WRS11 : SIZEOF (QUERY(sections < * component_sections | sections := (SELF))) = 0;
END_ENTITY;

```

```

ENTITY section_profile_derived
SUBTYPE OF (section_profile_complex);
    original_section : section_profile;
WHERE
    WRS12 : original_section :<> SELF;
END_ENTITY;

```

```

ENTITY section_profile_edge_defined
SUBTYPE OF (section_profile_complex);
    external_edge : bounded_curve;
    internal_edges : LIST [0:?] OF bounded_curve;
END_ENTITY;

```

ENTITY section_profile_i_type

SUPERTYPE OF (section_profile_i_type_asymmetric)

SUBTYPE OF (section_profile_simple);

overall_depth : positive_length_measure_with_unit;
 overall_width : positive_length_measure_with_unit;
 web_thickness : positive_length_measure_with_unit;
 flange_thickness : positive_length_measure_with_unit;
 internal_depth : OPTIONAL positive_length_measure_with_unit;
 flange_slope : OPTIONAL ratio_measure_with_unit;
 root_radius : OPTIONAL positive_length_measure_with_unit;
 edge_radius : OPTIONAL positive_length_measure_with_unit;

DERIVE

overall_depth_value : REAL := overall_depth.value_component;
 overall_width_value : REAL := overall_width.value_component;
 web_thickness_value : REAL := web_thickness.value_component;
 flange_thickness_value : REAL := flange_thickness.value_component;

WHERE

WRS13 : flange_thickness_value < (overall_depth_value/2);
 WRS14 : web_thickness_value < overall_width_value;

END_ENTITY;

ENTITY section_profile_i_type_asymmetric

SUPERTYPE OF (section_profile_i_type_rail)

SUBTYPE OF (section_profile_i_type);

top_flange_width : positive_length_measure_with_unit;
 bottom_flange_width : positive_length_measure_with_unit;
 bottom_flange_thickness : positive_length_measure_with_unit;
 bottom_root_radius : OPTIONAL positive_length_measure_with_unit;
 bottom_flange_slope : OPTIONAL ratio_measure_with_unit;
 bottom_flange_edge_radius : OPTIONAL positive_length_measure_with_unit;

WHERE

WRS43 : (SELF\section_profile_i_type.overall_width :=: top_flange_width) OR
 (SELF\section_profile_i_type.overall_width :=: bottom_flange_width);

END_ENTITY;

ENTITY section_profile_i_type_rail

SUBTYPE OF (section_profile_i_type_asymmetric);

top_edge_radius : positive_length_measure_with_unit;
 bottom_edge_radius : positive_length_measure_with_unit;
 top_flange_inner_slope : ratio_measure_with_unit;
 bottom_flange_inner_slope : ratio_measure_with_unit;
 transition_radius_top : OPTIONAL positive_length_measure_with_unit;
 transition_radius_bottom : OPTIONAL positive_length_measure_with_unit;

END_ENTITY;

ENTITY section_profile_rectangle

SUPERTYPE OF (section_profile_rectangle_hollow)


```

SUBTYPE OF (section_profile_simple);
  overall_depth : positive_length_measure_with_unit;
  overall_width : OPTIONAL positive_length_measure_with_unit;
  external_fillet_radius : OPTIONAL positive_length_measure_with_unit;
DERIVE
  overall_depth_value : REAL := overall_depth.value_component;
  overall_width_value : REAL := NVL(overall_width.value_component,
    overall_depth.value_component);
  external_fillet_value : REAL := NVL(external_fillet_radius.value_component, 0.0);
WHERE
  WRS15 : external_fillet_value < (overall_depth_value/2);
END_ENTITY;

```

```

ENTITY section_profile_rectangle_hollow
SUBTYPE OF (section_profile_rectangle);
  wall_thickness : positive_length_measure_with_unit;
  internal_fillet_radius : OPTIONAL positive_length_measure_with_unit;
DERIVE
  wall_thickness_value : REAL := wall_thickness.value_component;
  internal_radius_value : REAL := NVL(internal_fillet_radius.value_component, 0.0);
WHERE
  WRS16 : wall_thickness_value <
    ((SELF\section_profile_rectangle.overall_depth_value)/2);
  WRS17 : internal_radius_value <
    ((SELF\section_profile_rectangle.overall_depth_value)/2);
END_ENTITY;

```

```

ENTITY section_profile_simple
ABSTRACT SUPERTYPE OF (ONEOF
  (section_profile_i_type,
  section_profile_t_type,
  section_profile_channel,
  section_profile_angle,
  section_profile_circle,
  section_profile_rectangle))
SUBTYPE OF (section_profile);
END_ENTITY;

```

```

ENTITY section_profile_t_type
SUBTYPE OF (section_profile_simple);
  overall_depth : positive_length_measure_with_unit;
  flange_width : positive_length_measure_with_unit;
  flange_thickness : positive_length_measure_with_unit;
  web_thickness : positive_length_measure_with_unit;
  root_radius : OPTIONAL positive_length_measure_with_unit;
  flange_slope : OPTIONAL ratio_measure_with_unit;
  web_slope : OPTIONAL ratio_measure_with_unit;

```

```

    edge_radius : OPTIONAL positive_length_measure_with_unit;
DERIVE
    overall_depth_value : REAL := overall_depth.value_component;
    flange_width_value : REAL := flange_width.value_component;
    flange_thickness_value : REAL := flange_thickness.value_component;
    web_thickness_value : REAL := web_thickness.value_component;
WHERE
    WRS18 : flange_thickness_value < overall_depth_value;
    WRS19 : web_thickness_value < flange_width_value;
END_ENTITY;

```

ENTITY section_properties

```

SUPERTYPE OF (section_properties_asymmetric);
    profile : section_profile;
    origin_offset : ARRAY [1:2] OF OPTIONAL length_measure_with_unit;
    torsional_constant_lx : OPTIONAL inertia_measure_with_unit;
    inertia_moment_ly : OPTIONAL inertia_measure_with_unit;
    inertia_moment_lz : OPTIONAL inertia_measure_with_unit;
    section_area_Ax : OPTIONAL area_measure_with_unit;
    shear_area_Asy : OPTIONAL area_measure_with_unit;
    shear_area_Asz : OPTIONAL area_measure_with_unit;
    shear_deformation_area_Ay : OPTIONAL area_measure_with_unit;
    shear_deformation_area_Az : OPTIONAL area_measure_with_unit;
    surface_per_length : OPTIONAL positive_length_measure_with_unit;
    radius_of_gyration_ry : OPTIONAL positive_length_measure_with_unit;
    radius_of_gyration_rz : OPTIONAL positive_length_measure_with_unit;
    plastic_modulus_Sy : OPTIONAL modulus_measure_with_unit;
    plastic_modulus_Sz : OPTIONAL modulus_measure_with_unit;
    warping_constant : OPTIONAL derived_measure_with_unit;
    torsional_index : OPTIONAL REAL;
    buckling_parameter : OPTIONAL REAL;
    nominal_mass : OPTIONAL mass_per_length_measure_with_unit;
    actual_mass : OPTIONAL mass_per_length_measure_with_unit;
DERIVE
    y_offset : REAL := NVL(origin_offset[1].value_component, 0.0);
    z_offset : REAL := NVL(origin_offset[2].value_component, 0.0);
WHERE
    WRS39 : NOT( (profile.cardinal_point = 10) AND (y_offset <> 0.0) );
    WRS40 : NOT( (profile.cardinal_point = 10) AND (z_offset <> 0.0) );
END_ENTITY;

```

ENTITY section_properties_asymmetric

```

SUBTYPE OF (section_properties);
    neutral_axis_shear_centre : ARRAY [1:2] OF OPTIONAL length_measure_with_unit;
    theta_angle_z_axis_v_axis : OPTIONAL plane_angle_measure_with_unit;
    inertia_moment_lu : OPTIONAL inertia_measure_with_unit;
    inertia_moment_lv : OPTIONAL inertia_measure_with_unit;

```

```

    radius_of_gyration_ru : OPTIONAL positive_length_measure_with_unit;
    radius_of_gyration_rv : OPTIONAL positive_length_measure_with_unit;
    section_modulii : ARRAY [1:8] OF OPTIONAL modulus_measure_with_unit;
END_ENTITY;

```

```

ENTITY setting_out_point;
    set_out_site : site;
    location : geographical_location;
END_ENTITY;

```

```

ENTITY shape_representation
    SUPERTYPE OF (shape_representation_with_units)
    SUBTYPE OF (representation);
END_ENTITY; (* STEP Part 41 expanded (2nd edition unchanged) *)

```

```

ENTITY shape_representation_with_units
    SUBTYPE OF (shape_representation);
    WHERE
        WRS20 : ('STRUCTURAL_FRAME_SCHEMA.
            GEOMETRIC_REPRESENTATION_CONTEXT' IN TYPEOF
            (SELF\representation.context_of_items)) AND
            ('STRUCTURAL_FRAME_SCHEMA.GLOBAL_UNIT_ASSIGNED_CONTEXT' IN
            TYPEOF (SELF\representation.context_of_items));
        WRS21 : SIZEOF(QUERY(temp <* SELF\representation.items |
            ('STRUCTURAL_FRAME_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM') IN
            TYPEOF(temp))) > 0;
END_ENTITY;

```

```

ENTITY shell_based_surface_model
    SUBTYPE OF (geometric_representation_item);
    sbasm_boundary : SET [1:?] OF shell;
    WHERE
        WRS22 : constraints_geometry_shell_based_surface_model(SELF);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY shell_based_wireframe_model
    SUBTYPE OF (geometric_representation_item);
    sbwm_boundary : SET [1:?] OF shell;
    WHERE
        WRS23 : constraints_geometry_shell_based_wireframe_model(SELF);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY si_unit
    SUBTYPE OF (named_unit);
    prefix : OPTIONAL si_prefix;
    name : si_unit_name;
    DERIVE
        SELF\named_unit.dimensions : dimensional_exponents :=

```

```

        dimensions_for_si_unit (SELF.name);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

```

ENTITY site
SUPERTYPE OF (site_with_shape)
SUBTYPE OF (structural_frame_item);
    site_address : OPTIONAL address;
UNIQUE
    URS5 : SELF\structural_frame_item.item_number,
    SELF\structural_frame_item.item_name;
END_ENTITY;

```

```

ENTITY site_with_shape
SUBTYPE OF (site);
    shape : shape_representation_with_units;
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY solder
SUBTYPE OF (structural_frame_process);
    solder_type : soldering_type;
END_ENTITY;

```

```

ENTITY solid_angle_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRS24 : 'STRUCTURAL_FRAME_SCHEMA.SOLID_ANGLE_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);
    WRS25 : 'STRUCTURAL_FRAME_SCHEMA.SOLID_ANGLE_MEASURE' IN
        TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

```

```

ENTITY solid_angle_unit
SUBTYPE OF (named_unit);
WHERE
    WRS26 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND
        (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
        (SELF\named_unit.dimensions.time_exponent = 0.0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

(* Modified for LPM/6 *)

```

ENTITY solid_model
SUPERTYPE OF (ONEOF

```

```

        (manifold_solid_brep,
        csg_solid,
        swept_face_solid,
        swept_area_solid,
        solid_replica,
        brep_2d,
        trimmed_volume))
SUBTYPE OF (geometric_representation_item);
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

(* **Modified for LPM/6** *)

```

ENTITY solid_replica
SUBTYPE OF (solid_model);
    parent_solid : solid_model;
    transformation : cartesian_transformation_operator_3d;
WHERE
    WRS27 : acyclic_solid_replica(SELF, parent_solid);
    WRS44 : parent_solid\geometric_representation_item.dim = 3;
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

ENTITY sphere
SUBTYPE OF (geometric_representation_item);
    radius : positive_length_measure;
    centre : point;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* **Modified for LPM/6** - CIS ENTITY replaced with STEP Part 42 entity *)

```

ENTITY spherical_point
SUBTYPE OF (cartesian_point);
    r : length_measure;
    theta : plane_angle_measure;
    phi : plane_angle_measure;
DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure :=
        [r*sin(theta)*cos(phi), r*sin(theta)*sin(phi), r*cos(theta)];
WHERE
    WRS45 : r >= 0.0;
END_ENTITY; (* STEP Part 42 (New in 2nd edition) *)

```

```

ENTITY spherical_surface
SUBTYPE OF (elementary_surface);
    radius : positive_length_measure;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* **New for LPM/6** *)

```

ENTITY spherical_volume
SUBTYPE OF (volume);

```

```

    position : axis2_placement_3d;
    radius : positive_length_measure;
END_ENTITY; (* STEP Part 42 (New in 2nd edition) *)

```

```

ENTITY standard_uncertainty
SUBTYPE OF (uncertainty_qualifier);
    uncertainty_value : REAL;
END_ENTITY; (* STEP Part 45 (unchanged in TC1) *)

```

(* Modified for LPM/6 *)

```

ENTITY step_file
SUBTYPE OF (media_file);
INVERSE
    selected_content : SET [1:?] OF group_assignment FOR assigned_group;
WHERE
    WRS28 : (SELF\media_file.file_format = 'STP') OR (SELF\media_file.file_format = 'stp');
END_ENTITY;

```

(* Modified for LPM/6 *)

```

ENTITY structural_frame_item
SUPERTYPE OF (ONEOF
    (building,
    building_complex,
    feature,
    joint_system,
    located_item,
    material,
    project,
    project_plan,
    project_plan_item,
    section_profile,
    site,
    structural_frame_process,
    structural_frame_product,
    structure));
    item_number : INTEGER;
    item_name : label;
    item_description : OPTIONAL text;
DERIVE
    item_ref : BAG OF identifier := get_item_ref(SELF);
    cost_code : BAG OF label := get_item_cost_code(SELF);
    object_id : globally_unique_id := get_instance_id(SELF);
WHERE
    WRS49 : SIZEOF (USEDIN (SELF,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.DATA_ITEM')) <= 1;
END_ENTITY;

```

```
ENTITY structural_frame_item_approved;
    approved_item : structural_frame_item;
    assigned_approval : approval;
UNIQUE
    URS1 : approved_item, assigned_approval;
END_ENTITY;
```

```
ENTITY structural_frame_item_certified;
    certified_item : structural_frame_item;
    assigned_certification : certification;
UNIQUE
    URS2 : certified_item, assigned_certification;
END_ENTITY;
```

```
ENTITY structural_frame_item_documented;
    documented_item : structural_frame_item;
    document_reference : document_usage_constraint;
UNIQUE
    URS3 : documented_item, document_reference;
END_ENTITY;
```

```
ENTITY structural_frame_item_priced;
    priced_item : structural_frame_item;
    assigned_price: currency_measure_with_unit;
    price_description : text;
UNIQUE
    URS4 : priced_item, assigned_price;
END_ENTITY;
```

```
ENTITY structural_frame_item_relationship;
    relationship_name : label;
    relationship_description : OPTIONAL text;
    related_item : structural_frame_item;
    relating_item : structural_frame_item;
WHERE
    WRS29 : related_item :<>: relating_item;
END_ENTITY;
```

(* [Modified for LPM/6](#) - subtypes added *)

```
ENTITY structural_frame_process
SUPERTYPE OF (ONEOF
    (assemble,
    bend,
    braze,
    cut,
    dispatch,
    move,
```

```

        procure,
        solder,
        surface_treatment,
        weld))
SUBTYPE OF (structural_frame_item);
    place_of_process : OPTIONAL organizational_address;
END_ENTITY;

```

```

ENTITY structural_frame_product
SUPERTYPE OF (ONEOF
    (assembly,
    coating,
    part,
    fastener,
    fastener_mechanism,
    weld_mechanism,
    chemical_mechanism) ANDOR
    structural_frame_product_with_material)
SUBTYPE OF (structural_frame_item);
    life_cycle_stage : OPTIONAL label;
END_ENTITY;

```

```

ENTITY structural_frame_product_with_material
SUBTYPE OF (structural_frame_product);
    material_definition : material;
    nominal_mass : OPTIONAL mass_measure_with_unit;
    actual_mass : OPTIONAL mass_measure_with_unit;
END_ENTITY;

```

```

ENTITY structure
SUBTYPE OF (structural_frame_item);
UNIQUE
    URS6 : SELF\structural_frame_item.item_number,
    SELF\structural_frame_item.item_name;
END_ENTITY;

```

```

(* New for LPM/6 *)
ENTITY subedge
SUBTYPE OF (edge);
    parent_edge : edge;
END_ENTITY; (* STEP Part 42 2nd edition *)

```

```

ENTITY subface
SUBTYPE OF (face);
    parent_face : face;
WHERE
    WRS30 : NOT (mixed_loop_type_set(list_to_set(list_face_loops(SELF)) +

```



```

        list_to_set(list_face_loops(parent_face))));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY surface
SUPERTYPE OF (ONEOF
    (elementary_surface,
     swept_surface,
     bounded_surface,
     offset_surface,
     surface_replica))
SUBTYPE OF (geometric_representation_item);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY surface_curve
SUPERTYPE OF (ONEOF(intersection_curve, seam_curve) ANDOR
    bounded_surface_curve)
SUBTYPE OF (curve);
    curve_3d : curve;
    associated_geometry : LIST [1:2] OF pcurve_or_surface;
    master_representation : preferred_surface_curve_representation;
DERIVE
    basis_surface : SET [1:2] OF surface := get_basis_surface(SELf);
WHERE
    WRS31 : curve_3d.dim = 3;
    WRS32 : ('STRUCTURAL_FRAME_SCHEMA.PCURVE' IN
        TYPEOF(associated_geometry[1])) OR (master_representation <> pcurve_s1);
    WRS33 : ('STRUCTURAL_FRAME_SCHEMA.PCURVE' IN
        TYPEOF(associated_geometry[2])) OR (master_representation <> pcurve_s2);
    WRS34 : NOT ('STRUCTURAL_FRAME_SCHEMA.PCURVE' IN TYPEOF(curve_3d));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY surface_curve_swept_area_solid
SUBTYPE OF (swept_area_solid);
    directrix : curve;
    start_param : REAL;
    end_param : REAL;
    reference_surface : surface;
WHERE
    WRS46 : (NOT ('STRUCTURAL_FRAME_SCHEMA.SURFACE_CURVE' IN
        TYPEOF(directrix))) OR (reference_surface IN
        (directrix\surface_curve.basis_surface));
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY surface_curve_swept_face_solid
SUBTYPE OF (swept_face_solid);
    directrix : curve;

```

```

    start_param : REAL;
    end_param : REAL;
    reference_surface : surface;
WHERE
    WRS47 : (NOT ('STRUCTURAL_FRAME_SCHEMA.SURFACE_CURVE' IN
        TYPEOF(directrix))) OR (reference_surface IN
        (directrix\surface_curve.basis_surface));
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY surface_curve_swept_surface
SUBTYPE OF (swept_surface);
    directrix : curve;
    reference_surface : surface;
WHERE
    WRS48 : (NOT ('STRUCTURAL_FRAME_SCHEMA.SURFACE_CURVE' IN
        TYPEOF(directrix))) OR (reference_surface IN
        (directrix\surface_curve.basis_surface));
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY surface_of_linear_extrusion
SUBTYPE OF (swept_surface);
    extrusion_axis : vector;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

ENTITY surface_of_revolution
SUBTYPE OF (swept_surface);
    axis_position : axis1_placement;
DERIVE
    axis_line : line := dummy_gri ||
        curve() ||
        line (axis_position.location,dummy_gri ||
            vector(axis_position.z, 1.0));
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

ENTITY surface_patch;
    parent_surface : bounded_surface;
    u_transition : transition_code;
    v_transition : transition_code;
    u_sense : BOOLEAN;
    v_sense : BOOLEAN;
INVERSE
    using_surfaces : BAG [1:?] OF rectangular_composite_surface FOR segments;
WHERE
    WRS35 : (NOT ('STRUCTURAL_FRAME_SCHEMA.CURVE_BOUNDED_SURFACE'
        IN TYPEOF(parent_surface)));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```
ENTITY surface_replica
SUBTYPE OF (surface);
  parent_surface : surface;
  transformation : cartesian_transformation_operator_3d;
WHERE
  WRS36 : acyclic_surface_replica(SELF, parent_surface);
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)
```

```
ENTITY surface_treatment
ABSTRACT SUPERTYPE OF (ONEOF
  (surface_treatment_clean,
   surface_treatment_coat,
   surface_treatment_grind,
   surface_treatment_hard_stamp,
   surface_treatment_thermal))
SUBTYPE OF (structural_frame_process);
  surface_finish_specification : text;
END_ENTITY;
```

```
ENTITY surface_treatment_clean
SUBTYPE OF (surface_treatment);
  method : cleaning_method;
END_ENTITY;
```

```
ENTITY surface_treatment_coat
SUBTYPE OF (surface_treatment);
  methods : LIST [1:?] OF coating_method;
  layer_thicknesses : LIST [1:?] OF positive_length_measure_with_unit;
  coating_specifications : LIST [1:?] OF coating;
DERIVE
  number_of_layers : INTEGER := SIZEOF(methods);
WHERE
  WRS41 : SIZEOF(layer_thicknesses) = number_of_layers;
  WRS42 : SIZEOF(coating_specifications) = number_of_layers;
END_ENTITY;
```

```
ENTITY surface_treatment_grind
SUBTYPE OF (surface_treatment);
  finished_surface_irregularity : OPTIONAL length_measure_with_unit;
END_ENTITY;
```

```
ENTITY surface_treatment_hard_stamp
SUBTYPE OF (surface_treatment);
  stamp_method : cutting_type;
END_ENTITY;
```

```

ENTITY surface_treatment_thermal
SUPERTYPE OF (surface_treatment_thermal_timed)
SUBTYPE OF (surface_treatment);
    initial_temperature : thermodynamic_temperature_measure_with_unit;
    final_temperature : thermodynamic_temperature_measure_with_unit;
    maximum_temperature : thermodynamic_temperature_measure_with_unit;
END_ENTITY;

```

```

ENTITY surface_treatment_thermal_timed
SUBTYPE OF (surface_treatment_thermal);
    time_to_maximum : time_measure_with_unit;
    time_at_maximum : time_measure_with_unit;
    time_to_final : time_measure_with_unit;
END_ENTITY;

```

```

(* Modified for LPM/6 *)
ENTITY swept_area_solid
SUPERTYPE OF (ONEOF(
    extruded_area_solid,
    revolved_area_solid,
    surface_curve_swept_area_solid))
SUBTYPE OF (solid_model);
    swept_area : curve_bounded_surface;
WHERE
    WRS37 : 'STRUCTURAL_FRAME_SCHEMA.PLANE' IN
        TYPEOF(swept_area.basis_surface);
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

(* Modified for LPM/6 *)
ENTITY swept_face_solid
SUPERTYPE OF (ONEOF(
    extruded_face_solid,
    revolved_face_solid,
    surface_curve_swept_face_solid))
SUBTYPE OF (solid_model);
    swept_face : face_surface;
WHERE
    WRS38 : 'STRUCTURAL_FRAME_SCHEMA.PLANE' IN
        TYPEOF(swept_face.face_geometry);
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

```

(* Modified for LPM/6 *)
ENTITY swept_surface
SUPERTYPE OF (ONEOF(
    surface_of_linear_extrusion,
    surface_of_revolution,

```

```

    surface_curve_swept_surface,
    fixed_reference_swept_surface))
SUBTYPE OF (surface);
    swept_curve : curve;
END_ENTITY; (* STEP Part 42 (modified in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY tetrahedron
    SUBTYPE OF (faceted_primitive);
WHERE
    WRT15 : SIZEOF(points) = 4 ;
    WRT16: above_plane(points[1], points[2], points[3], points[4]) <> 0.0;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY tetrahedron_volume
    SUBTYPE OF (volume);
    point_1 : cartesian_point;
    point_2 : cartesian_point;
    point_3 : cartesian_point;
    point_4 : cartesian_point;
WHERE
    WRT10 : point_1.dim = 3;
    WRT11 : above_plane(point_1, point_2, point_3, point_4) <> 0.0 ;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY thermodynamic_temperature_measure_with_unit
    SUBTYPE OF (measure_with_unit);
WHERE
    WRT1 : 'STRUCTURAL_FRAME_SCHEMA.
    THERMODYNAMIC_TEMPERATURE_UNIT' IN TYPEOF
    (SELF\measure_with_unit.unit_component);
    WRT2 : 'STRUCTURAL_FRAME_SCHEMA.
    THERMODYNAMIC_TEMPERATURE_MEASURE' IN TYPEOF
    (SELF\measure_with_unit.value_component);
END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

```

```

ENTITY thermodynamic_temperature_unit
    SUBTYPE OF (named_unit);
WHERE
    WRT3 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND
    (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
    (SELF\named_unit.dimensions.time_exponent = 0.0) AND
    (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
    (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 1.0) AND
    (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
    (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);

```

END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

ENTITY time_measure_with_unit

SUBTYPE OF (measure_with_unit);

WHERE

WRT4 : 'STRUCTURAL_FRAME_SCHEMA.TIME_UNIT' IN
TYPEOF (SELF\measure_with_unit.unit_component);

WRT5 : 'STRUCTURAL_FRAME_SCHEMA.TIME_MEASURE' IN
TYPEOF (SELF\measure_with_unit.value_component);

END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

ENTITY time_unit

SUBTYPE OF (named_unit);

WHERE

WRT6 : (SELF\named_unit.dimensions.length_exponent = 0.0) AND
(SELF\named_unit.dimensions.mass_exponent = 0.0) AND
(SELF\named_unit.dimensions.time_exponent = 1.0) AND
(SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
(SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
(SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
(SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);

END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

ENTITY topological_representation_item

SUPERTYPE OF (ONEOF

(connected_edge_set,
connected_face_set,
edge,
face,
face_bound,
vertex,
vertex_shell,
wire_shell) ANDOR
loop ANDOR
path)

SUBTYPE OF (representation_item);

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY toroidal_surface

SUBTYPE OF (elementary_surface);

major_radius : positive_length_measure;

minor_radius : positive_length_measure;

END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

ENTITY toroidal_volume

SUBTYPE OF (volume);

```

    position : axis2_placement_3d;
    major_radius : positive_length_measure;
    minor_radius : positive_length_measure;
WHERE
    WRT13 : minor_radius < major_radius;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY torus
SUBTYPE OF (geometric_representation_item);
    position : axis1_placement;
    major_radius : positive_length_measure;
    minor_radius : positive_length_measure;
WHERE
    WRT7 : major_radius > minor_radius;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY trimmed_curve
SUBTYPE OF (bounded_curve);
    basis_curve : curve;
    trim_1 : SET [1:2] OF trimming_select;
    trim_2 : SET [1:2] OF trimming_select;
    sense_agreement : BOOLEAN;
    master_representation : trimming_preference;
WHERE
    WRT8 : (HIINDEX(trim_1) = 1) XOR (TYPEOF(trim_1[1]) <> TYPEOF(trim_1[2]));
    WRT9 : (HIINDEX(trim_2) = 1) XOR (TYPEOF(trim_2[1]) <> TYPEOF(trim_2[2]));
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY trimmed_volume
SUBTYPE OF (solid_model);
    basis_volume : volume;
    u1 : parameter_value;
    u2 : parameter_value;
    v1 : parameter_value;
    v2 : parameter_value;
    w1 : parameter_value;
    w2 : parameter_value;
WHERE
    WRT12 : u1 <> u2;
    WRT13 : v1 <> v2;
    WRT14 : w1 <> w2;
END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

```

```

ENTITY truncated_pyramid
SUBTYPE OF (boolean_result);
END_ENTITY; (* STEP Part 225 *)

```

ENTITY type_qualifier;
 name : label;
 END_ENTITY; (* STEP Part 45 (unchanged in TC1) *)

(* Modified for LPM/6 *)

ENTITY uncertainty_measure_with_unit
 SUBTYPE OF (measure_with_unit);
 name : label;
 description : text;
 WHERE
 WRU1 : valid_measure_value (SELF\measure_with_unit.value_component);
 END_ENTITY; (* STEP Part 43 (WR Modified in 2nd edition) *)

ENTITY uncertainty_qualifier
 SUPERTYPE OF (ONEOF(qualitative_uncertainty, standard_uncertainty));
 measure_name : label;
 description : text;
 END_ENTITY; (* STEP Part 45 (unchanged in TC1) *)

ENTITY uniform_curve
 SUBTYPE OF (b_spline_curve);
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

ENTITY uniform_surface
 SUBTYPE OF (b_spline_surface);
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* New for LPM/6 *)

ENTITY uniform_volume
 SUBTYPE OF (b_spline_volume);
 END_ENTITY; (* STEP Part 42 (new in 2nd edition) *)

ENTITY vector
 SUBTYPE OF (geometric_representation_item);
 orientation : direction;
 magnitude : length_measure;
 WHERE
 WRV1 : magnitude >= 0.0;
 END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

(* Modified for LPM/6 *)

ENTITY versioned_action_request;
 id : identifier;
 version : label;


```

    purpose : text;
    description : OPTIONAL text;
END_ENTITY; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

ENTITY vertex
SUBTYPE OF (topological_representation_item);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY vertex_loop
SUBTYPE OF (loop);
    loop_vertex : vertex;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY vertex_point
SUBTYPE OF (vertex, geometric_representation_item);
    vertex_geometry : point;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

ENTITY vertex_shell
SUBTYPE OF (topological_representation_item);
    vertex_shell_extent : vertex_loop;
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

ENTITY volume
SUPERTYPE OF (ONEOF(
    block_volume,
    wedge_volume,
    spherical_volume,
    cylindrical_volume,
    eccentric_conical_volume,
    toroidal_volume,
    pyramid_volume,
    b_spline_volume,
    ellipsoid_volume,
    tetrahedron_volume,
    hexahedron_volume))
SUBTYPE OF (geometric_representation_item);
WHERE
    WRV5 : SELF\geometric_representation_item.dim = 3;
END_ENTITY; (* STEP Part 42 (new for 2nd edition) *)

```

```

ENTITY volume_measure_with_unit
SUBTYPE OF (measure_with_unit);
WHERE
    WRV2 : 'STRUCTURAL_FRAME_SCHEMA.VOLUME_UNIT' IN
        TYPEOF (SELF\measure_with_unit.unit_component);

```

```

WRV3 : 'STRUCTURAL_FRAME_SCHEMA.VOLUME_MEASURE' IN
      TYPEOF (SELF\measure_with_unit.value_component);
END_ENTITY; (* based on STEP Part 41 2nd edition (WR added) *)

```

```

ENTITY volume_unit
SUBTYPE OF (named_unit);
WHERE
  WRV4 : (SELF\named_unit.dimensions.length_exponent = 3.0) AND
        (SELF\named_unit.dimensions.mass_exponent = 0.0) AND
        (SELF\named_unit.dimensions.time_exponent = 0.0) AND
        (SELF\named_unit.dimensions.electric_current_exponent = 0.0) AND
        (SELF\named_unit.dimensions.thermodynamic_temperature_exponent = 0.0) AND
        (SELF\named_unit.dimensions.amount_of_substance_exponent = 0.0) AND
        (SELF\named_unit.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY; (* STEP Part 41 (2nd edition unchanged) *)

```

(* New for LPM/6 *)

```

ENTITY wedge_volume
SUBTYPE OF (volume);
  position : axis2_placement_3d;
  x : positive_length_measure;
  y : positive_length_measure;
  z : positive_length_measure;
  ltx : length_measure;
WHERE
  WRE5 : ((0.0 <= ltx) AND (ltx < x));
END_ENTITY; (* STEP Part 42 (New for 2nd edition) *)

```

(* Modified for LPM/6 Subtypes added - see Issue 99 *)

```

ENTITY weld
SUPERTYPE OF (ONEOF(weld_arc,
  weld_beam,
  weld_gas,
  weld_other,
  weld_pressure,
  weld_resistance,
  weld_stud))
SUBTYPE OF (structural_frame_process);
  electrode_type : OPTIONAL label;
  weld_type : welding_type;
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY weld_arc
SUBTYPE OF (weld);
  weld_arc_type : welding_type_arc;
WHERE

```

```

WRW13 : SELF\weld.weld_type = FUSION_WELD;
WRW14 : EXISTS(SELF\weld.electrode_type);
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY weld_beam
SUBTYPE OF (weld);
    weld_beam_type : welding_type_beam;
WHERE
    WRW19 : (SELF\weld.weld_type = FUSION_WELD) OR (SELF\weld.weld_TYPE =
        LASER_WELD);
    WRW20 : NOT(EXISTS(SELF\weld.electrode_type));
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY weld_gas
SUBTYPE OF (weld);
    weld_gas_type : welding_type_gas;
WHERE
    WRW15 : SELF\weld.weld_type = FUSION_WELD;
    WRW16 : NOT(EXISTS(SELF\weld.electrode_type));
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY weld_other
SUBTYPE OF (weld);
    weld_other_type : welding_type_other;
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY weld_pressure
SUBTYPE OF (weld);
    weld_pressure_type : welding_type_pressure;
WHERE
    WRW17 : NOT((SELF\weld.weld_type = LASER_WELD) OR (SELF\weld.weld_TYPE =
        FLASH_WELD));
    WRW18 : NOT(EXISTS(SELF\weld.electrode_type));
END_ENTITY;

```

(* New for LPM/6 *)

```

ENTITY weld_resistance
SUBTYPE OF (weld);
    weld_resistance_type : welding_type_resistance;
END_ENTITY;

```

(* New for LPM/6 *)

```
ENTITY weld_stud
SUBTYPE OF (weld);
    weld_stud_type : welding_type_stud;
END_ENTITY;
```

(* Modified for LPM/6 - subtypes added *)

```
ENTITY weld_mechanism
SUPERTYPE OF (ONEOF(
    weld_mechanism_complex,
    weld_mechanism_prismatic,
    weld_mechanism_fillet,
    weld_mechanism_groove,
    weld_mechanism_spot_seam))
SUBTYPE OF (structural_frame_product);
    weld_mechanism_type : weld_type;
    penetration : weld_penetration;
    weld_dimension : OPTIONAL positive_length_measure_with_unit;
    weld_dimension_name : OPTIONAL label;
    weld_design_strength : OPTIONAL pressure_measure_with_unit;
END_ENTITY;
```

(* New for LPM/6 - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_complex
SUBTYPE OF (weld_mechanism);
    weld_shape : shape_representation_with_units;
END_ENTITY;
```

(* New for LPM/6 - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_fillet
SUPERTYPE OF (ONEOF(weld_mechanism_fillet_continuous,
    weld_mechanism_fillet_intermittent))
SUBTYPE OF (weld_mechanism);
    sidedness : weld_sidedness;
    surface_shape : weld_surface_shape;
    joint_configuration : weld_configuration;
    leg_length_y : OPTIONAL positive_length_measure_with_unit;
    leg_length_z : OPTIONAL positive_length_measure_with_unit;
WHERE
    WRW7 : SELF\weld_mechanism.weld_mechanism_type = FILLET_WELD;
    WRW8 : NOT ((SELF\weld_mechanism.penetration = FULL_PENETRATION) OR
        (SELF\weld_mechanism.penetration = PARTIAL_PENETRATION));
END_ENTITY;
```

(* **New for LPM/6** - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_fillet_continuous
SUBTYPE OF (weld_mechanism_fillet);
END_ENTITY;
```

(* **New for LPM/6** - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_fillet_intermittent
SUBTYPE OF (weld_mechanism_fillet);
    end_rules : weld_intermittent_rule ;
    cutout_rules : weld_intermittent_rule ;
    penetration_rules : weld_intermittent_rule ;
    fillet_weld_length : positive_length_measure_with_unit;
    fillet_weld_spacing : positive_length_measure_with_unit;
    fillet_alignment : weld_alignment;
END_ENTITY;
```

(* **New for LPM/6** - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_groove
SUPERTYPE OF(ONEOF(weld_mechanism_groove_beveled,
    weld_mechanism_groove_butt))
SUBTYPE OF (weld_mechanism);
    sidedness : weld_sidedness;
    backing_type : OPTIONAL weld_backing_type;
    weld_joint_spacer : BOOLEAN;
    surface_shape : weld_surface_shape;
    joint_configuration : weld_configuration;
    root_gap : OPTIONAL positive_length_measure_with_unit;
    root_face : OPTIONAL positive_length_measure_with_unit;
WHERE
    WRW9 : SELF\weld_mechanism.weld_mechanism_type = BUTT_WELD;
    WRW10 : NOT (SELF\weld_mechanism.penetration = DEEP_PENETRATION);
END_ENTITY;
```

(* **New for LPM/6** - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_groove_beveled
SUBTYPE OF (weld_mechanism_groove);
    endcut_shape : weld_shape_bevel;
    groove_angle : OPTIONAL plane_angle_measure_with_unit;
    groove_depth : OPTIONAL positive_length_measure_with_unit;
    groove_radius : OPTIONAL positive_length_measure_with_unit;
    taper : weld_taper_type;
    taper_angle : OPTIONAL plane_angle_measure_with_unit;
WHERE
    WRW12 : NOT ((taper = NON_TAPER) AND EXISTS(taper_angle));
END_ENTITY;
```

(* New for LPM/6 - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_groove_butt
SUBTYPE OF (weld_mechanism_groove);
    face_shape : weld_shape_butt;
END_ENTITY;
```

(* New for LPM/6 - see Issues 87 and 88 *)

```
ENTITY weld_mechanism_prismatic
SUBTYPE OF (weld_mechanism);
    cross_sections : LIST [2:?] OF section_profile;
    points_defining_weld_path : LIST [2:?] OF UNIQUE point_on_curve;
    section_orientations : LIST [2:?] OF orientation_select;
    joint_configuration : weld_configuration;
DERIVE
    number_of_sections : INTEGER := SIZEOF(cross_sections);
    curve_defining_weld : curve :=
        points_defining_weld_path[1]\point_on_curve.basis_curve;
    joints : SET [0:?] OF joint_system_welded := bag_to_set
        (USEDIN(SELf,'STRUCTURAL_FRAME_SCHEMA.
            JOINT_SYSTEM_WELDED.WELD_SPECIFICATION'));
WHERE
    WRW3 : ( (SIZEOF (points_defining_weld_path) = number_of_sections)
        AND (SIZEOF (section_orientations) = number_of_sections) );
    WRW4 : SIZEOF(QUERY(temp <* points_defining_weld_path |
        (temp\point_on_curve.basis_curve) :<>: curve_defining_weld)) = 0;
    WRW5 : SIZEOF(QUERY(joint <* joints | (NOT('STRUCTURAL_FRAME_SCHEMA.
        JOINT_SYSTEM_WELDED_LINEAR' IN TYPEOF(joint))))=0;
    WRW6 : SIZEOF(QUERY(joint <* joints | (NOT(joint.weld_path :=
        curve_defining_weld)))) = 0;
END_ENTITY;
```

(* New for LPM/6 - see Issue 87 *)

```
ENTITY weld_mechanism_spot_seam
SUBTYPE OF (weld_mechanism);
    joint_configuration : weld_configuration;
WHERE
    WRW11 : (SELF\weld_mechanism.weld_mechanism_type = SPOT_WELD) OR
        (SELF\weld_mechanism.weld_mechanism_type = SEAM_WELD);
END_ENTITY;
```

```
ENTITY wire_shell
SUBTYPE OF (topological_representation_item);
    wire_shell_extent : SET [1:?] OF loop;
WHERE
    WRW1 : NOT mixed_loop_type_set(wire_shell_extent);
END_ENTITY; (* STEP Part 42 (unchanged in 2nd edition) *)
```

```

ENTITY zone
ABSTRACT SUPERTYPE OF (ONEOF
    (zone_of_building,
    zone_of_project,
    zone_of_site,
    zone_of_structure) ANDOR
    zone_bounded);
zone_name : label;
zone_description : OPTIONAL text;
END_ENTITY;

```

```

ENTITY zone_bounded
SUBTYPE OF (zone);
    bounding_gridlines : SET [2:4] OF gridline;
    bounding_levels : OPTIONAL SET [1:2] OF grid_level;
DERIVE
    bounding_grid : grid := bounding_gridlines[1].parent_grid;
WHERE
    WRZ1 : SIZEOF(QUERY(line <* bounding_gridlines | line.parent_grid :<>:
        (bounding_grid)) ) = 0;
    WRZ2 : NOT (EXISTS(bounding_levels) AND
        (SIZEOF(QUERY(level <* bounding_levels | level.parent_grid :<>:
            (bounding_grid)) ) <> 0));
END_ENTITY;

```

```

ENTITY zone_of_building
SUPERTYPE OF (zone_of_building_storey)
SUBTYPE OF (zone);
    zone_for_building : building;
END_ENTITY;

```

```

ENTITY zone_of_building_storey
SUBTYPE OF (zone_of_building);
    storey_height : positive_length_measure_with_unit;
    storey_level : OPTIONAL length_measure_with_unit;
    datum_name : OPTIONAL text;
END_ENTITY;

```

```

ENTITY zone_of_project
SUBTYPE OF (zone);
    zone_for_project : project;
END_ENTITY;

```

```

ENTITY zone_of_site
SUBTYPE OF (zone);
    zone_for_site : site;
END_ENTITY;

```

(* Extended for LPM/6 - see Issue 98 *)

```
ENTITY zone_of_structure
  SUPERTYPE OF (ONEOF(zone_of_structure_sequence))
  SUBTYPE OF (zone);
    zone_for_structure : structure;
END_ENTITY;
```

(* **New for LPM/6** - see Issue 98 *)

```
ENTITY zone_of_structure_sequence
  SUPERTYPE OF (ONEOF(zone_of_structure_sequence_lot))
  SUBTYPE OF (zone_of_structure);
    parent_zone : zone_of_structure;
  DERIVE
    lots : SET[0:?] OF zone_of_structure_sequence_lot := bag_to_set
      (USEDIN(SELF, 'STRUCTURAL_FRAME_SCHEMA.' +
        'ZONE_OF_STRUCTURE_SEQUENCE_LOT.' +
        'PARENT_SEQUENCE'));
    assemblies : SET[1:?] OF located_assembly := bag_to_set (USEDIN(SELF,
      'STRUCTURAL_FRAME_SCHEMA.LOCATED_ASSEMBLY.PARENT_STRUCTURE'
    ));
  WHERE
    WRZ3 : parent_zone :<>: (SELF);
    WRZ4 : NOT ('STRUCTURAL_FRAME_SCHEMA.
      ZONE_OF_STRUCTURE_SEQUENCE_LOT' IN TYPEOF (parent_zone));
END_ENTITY;
```

(* **New for LPM/6** - see Issue 98 *)

```
ENTITY zone_of_structure_sequence_lot
  SUBTYPE OF (zone_of_structure_sequence);
  DERIVE
    parent_sequence : zone_of_structure :=
      SELF\zone_of_structure_sequence.parent_zone;
  WHERE
    WRZ5 : 'STRUCTURAL_FRAME_SCHEMA.ZONE_OF_STRUCTURE_SEQUENCE' IN
      TYPEOF(parent_sequence);
    WRZ6 : SIZEOF (SELF\zone_of_structure_sequence.lots) = 0;
    WRZ7 : (SELF\zone_of_structure_sequence.parent_zone) :<>: (SELF);
END_ENTITY;
```

(* **RULE Declaration** *)

```
RULE compatible_dimension
  FOR(cartesian_point,
    direction,
    representation_context,
```



```
geometric_representation_context);
```

```
WHERE
```

```
  WRR1 : (* ensure that the count of coordinates of each cartesian_point
           matches the coordinate_space_dimension of each geometric_context in
           which it is geometrically_founded *)
```

```
  SIZEOF(QUERY(x <* cartesian_point | SIZEOF(QUERY
    (y <* geometric_representation_context | item_in_context(x,y) AND
    (HIINDEX(x.coordinates) <> y.coordinate_space_dimension)))) > 0 )) = 0;
```

```
  WRR2 : (* ensure that the count of direction_ratios of each direction
           matches the coordinate_space_dimension of each geometric_context in
           which it is geometrically_founded *)
```

```
  SIZEOF(QUERY(x <* direction | SIZEOF( QUERY
    (y <* geometric_representation_context | item_in_context(x,y) AND
    (HIINDEX(x.direction_ratios) <> y.coordinate_space_dimension))))
    > 0 )) = 0;
```

```
END_RULE;
```

```
(* FUNCTION Declarations *)
```

```
(* New for LPM/6 *)
```

```
FUNCTION above_plane
```

```
  (p1, p2, p3, p4 : cartesian_point) : REAL;
```

```
  LOCAL
```

```
    dir2, dir3, dir4 : direction :=
      dummy_gri || direction([1.0, 0.0, 0.0]);
```

```
    val, mag : REAL;
```

```
  END_LOCAL;
```

```
  IF (p1.dim <> 3) THEN
```

```
    RETURN(?);
```

```
  END_IF;
```

```
  REPEAT i := 1 TO 3;
```

```
    dir2.direction_ratios[i] := p2.coordinates[i] - p1.coordinates[i];
```

```
    dir3.direction_ratios[i] := p3.coordinates[i] - p1.coordinates[i];
```

```
    dir4.direction_ratios[i] := p4.coordinates[i] - p1.coordinates[i];
```

```
    mag := dir4.direction_ratios[i]*dir4.direction_ratios[i];
```

```
  END_REPEAT;
```

```
  mag := sqrt(mag);
```

```
  val := mag*dot_product(dir4, cross_product(dir2, dir3).orientation);
```

```
  RETURN(val);
```

```
END_FUNCTION; (* STEP Part 42 (New for 2nd edition) *)
```

```

FUNCTION acyclic_curve_replica
    (rep : curve_replica; parent : curve) : BOOLEAN;
IF NOT (('STRUCTURAL_FRAME_SCHEMA.CURVE_REPLICA') IN TYPEOF(parent))
    THEN
    RETURN (TRUE);
END_IF;

IF (parent :=: rep) THEN
    RETURN (FALSE);

    ELSE RETURN(acyclic_curve_replica(rep, parent\curve_replica.parent_curve));
    END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION acyclic_document_relationship
    (relation      : document_relationship;
     relatives     : SET [1:?] OF document;
     specific_relation : STRING) : BOOLEAN;
LOCAL
    x      : SET OF document_relationship;
END_LOCAL;
IF relation.relying_document IN relatives THEN
    RETURN (FALSE);
END_IF;
x := QUERY (doc <* bag_to_set
            (USEDIN (relation.relying_document,
                    'STRUCTURAL_FRAME_SCHEMA.' +
                    'DOCUMENT_RELATIONSHIP.' +
                    'RELATED_DOCUMENT')) |
            specific_relation IN TYPEOF (doc));
REPEAT i := 1 TO HIINDEX(x);
    IF NOT acyclic_document_relationship
        (x[i],
         relatives + relation.relying_document,
         specific_relation) THEN
        RETURN(FALSE);
    END_IF;
END_REPEAT;
RETURN(TRUE);
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION acyclic_group_relationship
    (relation      : group_relationship;

```

```

    relatives      : SET [1:?] OF group;
    specific_relation : STRING) : BOOLEAN;
LOCAL
    x      : SET OF group_relationship;
END_LOCAL;
IF relation.relating_group IN relatives THEN
    RETURN (FALSE);
END_IF;
x := QUERY (grp <* bag_to_set
            (USEDIN (relation.relating_group,
                    'STRUCTURAL_FRAME_SCHEMA.' +
                    'GROUP_RELATIONSHIP.' +
                    'RELATED_GROUP')) |
            specific_relation IN TYPEOF (grp));
REPEAT i := 1 TO HIINDEX(x);
    IF NOT acyclic_group_relationship
        (x[i],
         relatives + relation.relating_group,
         specific_relation) THEN
        RETURN(FALSE);
    END_IF;
END_REPEAT;
RETURN(TRUE);
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

```

FUNCTION acyclic_mapped_representation
    (parent_set : SET OF representation;
     children_set : SET OF representation_item) : BOOLEAN;
LOCAL
    x,y : SET OF representation_item;
    i : INTEGER;
END_LOCAL;

    (* Determine the subset of children_set that are mapped_items. *)

    x := QUERY(z <* children_set | 'STRUCTURAL_FRAME_SCHEMA.MAPPED_ITEM'
              IN TYPEOF(z));

    (* Determine that the subset has elements. *)

    IF SIZEOF(x) > 0 THEN

        (* Check each element of the set. *)

        REPEAT i := 1 TO HIINDEX(x);

```

```

    (* If the selected element maps a representation in the parent_set, then return false. *)

    IF x[i]\mapped_item.mapping_source.mapped_representation
      IN parent_set THEN
      RETURN (FALSE);
    END_IF;

    (* Recursive check of the items of mapped_rep. *)
    IF NOT acyclic_mapped_representation
      (parent_set + x[i]\mapped_item.mapping_source.mapped_representation,
       x[i]\mapped_item.mapping_source.mapped_representation.items) THEN
      RETURN (FALSE);
    END_IF;
    END_REPEAT;
  END_IF;

  (* Determine the subset of children_set that are not mapped_items. *)

  x := children_set - x;

  (* Determine that the subset has elements. *)

  IF SIZEOF(x) > 0 THEN

    (* For each element of the set: *)

    REPEAT i := 1 TO HIINDEX(x);

      (* Determine the set of representation_items referenced. *)

      y := QUERY(z < * bag_to_set( USEDIN(x[i], "")) |
        'STRUCTURAL_FRAME_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

      (* Recursively check for an offending mapped_item.
         Return false for any errors encountered. *)

      IF NOT acyclic_mapped_representation(parent_set, y) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;

  (* Return true when all elements are checked and no error conditions found. *)
  RETURN (TRUE);
END_FUNCTION; (* STEP Part 43 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION acyclic_organization_relationship
(relation      : organization_relationship;
 relatives     : SET [1:?] OF organization;
 specific_relation : STRING) : BOOLEAN;
LOCAL
  x      : SET OF organization_relationship;
END_LOCAL;
IF relation.relying_organization IN relatives THEN
  RETURN (FALSE);
END_IF;
x := QUERY (org <* bag_to_set
            (USEDIN (relation.relying_organization,
                    'STRUCTURAL_FRAME_SCHEMA.' +
                    'ORGANIZATION_RELATIONSHIP.' +
                    'RELATED_ORGANIZATION')) |
            specific_relation IN TYPEOF (org));
REPEAT i := 1 TO HIINDEX(x);
  IF NOT acyclic_organization_relationship
    (x[i],
     relatives + relation.relying_organization,
     specific_relation) THEN
    RETURN(FALSE);
  END_IF;
END_REPEAT;
RETURN(TRUE);
END_FUNCTION; (* STEP Part 41 (modified in 2nd edition) *)

```

```

FUNCTION acyclic_point_replica
(rep : point_replica; parent : point) : BOOLEAN;
IF NOT (('STRUCTURAL_FRAME_SCHEMA.POINT_REPLICA') IN TYPEOF(parent))
THEN
  RETURN (TRUE);
END_IF;

IF (parent :=: rep) THEN
  RETURN (FALSE);

  ELSE RETURN(acyclic_point_replica(rep, parent\point_replica.parent_pt));
END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION acyclic_set_replica
(rep : geometric_set_replica; parent : geometric_set) : BOOLEAN;

```

```

IF NOT (('STRUCTURAL_FRAME_SCHEMA.GEOMETRIC_SET_REPLICA') IN
  TYPEOF(parent)) THEN
  RETURN (TRUE);
END_IF;

```

```

IF (parent :=: rep) THEN
  RETURN (FALSE);

```

```

  ELSE RETURN(acyclic_set_replica(rep,
    parent\geometric_set_replica.parent_set));
  END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION acyclic_solid_replica
  (rep : solid_replica; parent : solid_model) : BOOLEAN;
IF NOT (('STRUCTURAL_FRAME_SCHEMA.SOLID_REPLICA') IN TYPEOF(parent))
  THEN
  RETURN (TRUE);
END_IF;

```

```

IF (parent :=: rep) THEN
  RETURN (FALSE);

```

```

  ELSE RETURN(acyclic_solid_replica(rep, parent\solid_replica.parent_solid));
  END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION acyclic_surface_replica
  (rep : surface_replica; parent : surface) : BOOLEAN;
IF NOT (('STRUCTURAL_FRAME_SCHEMA.SURFACE_REPLICA') IN TYPEOF(parent))
  THEN
  RETURN (TRUE);
END_IF;

```

```

IF (parent :=: rep) THEN
  RETURN (FALSE);

```

```

  ELSE RETURN(acyclic_surface_replica(rep,
    parent\surface_replica.parent_surface));
  END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION associated_surface
  (arg : pcurve_or_surface) : surface;
LOCAL
  surf : surface;

```

```

END_LOCAL;

IF 'STRUCTURAL_FRAME_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
    surf := arg.basis_surface;
ELSE
    surf := arg;
END_IF;
RETURN(surf);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION bag_to_set
(the_bag : BAG OF GENERIC : intype) : SET OF GENERIC : intype;
LOCAL
    the_set: SET OF GENERIC : intype := [];
END_LOCAL;
IF SIZEOF (the_bag) > 0 THEN
    REPEAT i := 1 to HIINDEX (the_bag);
        the_set := the_set + the_bag [i];
    END_REPEAT;
END_IF;
RETURN (the_set);
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

(* Modified for LPM/6 *)

```

FUNCTION base_axis
(dim : INTEGER; axis1, axis2, axis3 : direction) : LIST [2:3] OF direction;
LOCAL
    u : LIST [2:3] OF direction;
    factor : REAL;
    d1, d2 : direction;
END_LOCAL;

IF (dim = 3) THEN
    d1 := NVL(normalise(axis3), dummy_gri || direction([0.0,0.0,1.0]));
    d2 := first_proj_axis(d1,axis1);
    u := [d2, second_proj_axis(d1,d2,axis2), d1];
ELSE
    IF EXISTS(axis1) THEN
        d1 := normalise(axis1);
        u := [d1, orthogonal_complement(d1)];
        IF EXISTS(axis2) THEN
            factor := dot_product(axis2,u[2]);
            IF (factor < 0.0) THEN
                u[2].direction_ratios[1] := -u[2].direction_ratios[1];
            END_IF;
        END_IF;
    END_IF;
END_IF;

```

```

        u[2].direction_ratios[2] := -u[2].direction_ratios[2];
    END_IF;
END_IF;
ELSE
    IF EXISTS(axis2) THEN
        d1 := normalise(axis2);
        u := [orthogonal_complement(d1), d1];
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := -u[1].direction_ratios[2];
    ELSE
        u := [dummy_gri || direction([1.0, 0.0]), dummy_gri ||
            direction([0.0, 1.0])];
    END_IF;
END_IF;
END_IF;
RETURN(u);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION boolean_choose
    (b : boolean; choice1, choice2 : generic : item) : generic : item;

    IF b THEN
        RETURN (choice1);
    ELSE
        RETURN (choice2);
    END_IF;
END_FUNCTION; (* STEP Part 42 (modified for 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION build_2axes
    (ref_direction : direction) : LIST [2:2] OF direction;
    LOCAL
        d : direction := NVL(normalise(ref_direction),
            dummy_gri || direction([1.0,0.0]));
    END_LOCAL;

    RETURN([d, orthogonal_complement(d)]);
END_FUNCTION; (* STEP part 42 (modified in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION build_axes
    (axis, ref_direction : direction) : LIST [3:3] OF direction;
    LOCAL
        d1, d2 : direction;

```



```

END_LOCAL;
d1 := NVL(normalise(axis), dummy_gri || direction([0.0,0.0,1.0]));
d2 := first_proj_axis(d1, ref_direction);
RETURN([d2, normalise(cross_product(d1,d2)).orientation, d1]);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION build_transformed_set
  (tr: cartesian_transformation_operator;
   gset : geometric_set) : SET [0:?] OF geometric_set_select;
LOCAL
  s      : SET [1:?] OF geometric_set_select := gset.elements;
  trset  : SET [0:?] OF geometric_set_select := [];
END_LOCAL;
REPEAT j := 1 TO SIZEOF(s);
  IF ('STRUCTURAL_FRAME_SCHEMA.CURVE' IN TYPEOF(s[j])) THEN
    trset := trset + dummy_gri || curve() || curve_replica(s[j],tr); ELSE
    IF ('STRUCTURAL_FRAME_SCHEMA.POINT' IN TYPEOF(s[j])) THEN
      trset := trset + dummy_gri || point() || point_replica(s[j],tr);
    ELSE
      IF ('STRUCTURAL_FRAME_SCHEMA.SURFACE' IN TYPEOF(s[j])) THEN
        trset := trset + dummy_gri || surface() || surface_replica(s[j],
          tr || cartesian_transformation_operator_3d (?));
      END_IF;
    END_IF;
  END_IF;
END_REPEAT;
RETURN(trset);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* New for LPM/6 *)

```

FUNCTION closed_shell_reversed
  (a_shell : closed_shell) : oriented_closed_shell;
LOCAL
  the_reverse : oriented_closed_shell;
END_LOCAL;
IF ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF
  (a_shell) ) THEN
  the_reverse := dummy_tri ||
    connected_face_set (
      a_shell\connected_face_set.cfs_faces) ||
    closed_shell () || oriented_closed_shell(
      a_shell\oriented_closed_shell.closed_shell_element,
      NOT(a_shell\oriented_closed_shell.orientation));
ELSE

```

```

    the_reverse := dummy_tri ||
        connected_face_set (
            a_shell\connected_face_set.cfs_faces) ||
        closed_shell () || oriented_closed_shell (a_shell, FALSE);
END_IF;
RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (new in 2nd edition) *)

```

```

FUNCTION conditional_reverse
    (p : BOOLEAN; an_item : reversible_topology) : reversible_topology;
IF p THEN
    RETURN (an_item);
ELSE
    RETURN (topology_reversed (an_item));
END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION constraints_composite_curve_on_surface
    (c: composite_curve_on_surface) : BOOLEAN;
LOCAL
    n_segments : INTEGER := SIZEOF(c.segments);
END_LOCAL;

REPEAT k := 1 TO n_segments;
    IF (NOT('STRUCTURAL_FRAME_SCHEMA.PCURVE' IN
        TYPEOF(c\composite_curve.segments[k].parent_curve))) AND
        (NOT('STRUCTURAL_FRAME_SCHEMA.SURFACE_CURVE' IN
            TYPEOF(c\composite_curve.segments[k].parent_curve))) AND
        (NOT('STRUCTURAL_FRAME_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN
            TYPEOF(c\composite_curve.segments[k].parent_curve))) THEN
        RETURN (FALSE);
    END_IF;
END_REPEAT;
RETURN(TRUE);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION constraints_geometry_shell_based_surface_model
    (m: shell_based_surface_model): BOOLEAN;
LOCAL
    result : BOOLEAN := TRUE;
END_LOCAL;

REPEAT j := 1 TO SIZEOF(m.sbsm_boundary);

```

```

    IF (NOT ('STRUCTURAL_FRAME_SCHEMA.OPEN_SHELL' IN
    TYPEOF(m.sbsm_boundary[j])) AND
    (NOT ('STRUCTURAL_FRAME_SCHEMA.CLOSED_SHELL' IN
    TYPEOF(m.sbsm_boundary[j]))))
    THEN
        result := FALSE;
        RETURN(result);
        (* A surface model is composed of OPEN_ and CLOSED_SHELLs. *)
    END_IF;
END_REPEAT;

RETURN(result);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

FUNCTION constraints_geometry_shell_based_wireframe_model
    (m : shell_based_wireframe_model) : BOOLEAN;
LOCAL
    result : BOOLEAN := TRUE;
END_LOCAL;

REPEAT j := 1 TO SIZEOF(m.sbwm_boundary);
    IF (NOT ('STRUCTURAL_FRAME_SCHEMA.WIRE_SHELL' IN
    TYPEOF(m.sbwm_boundary[j])) AND
    (NOT ('STRUCTURAL_FRAME_SCHEMA.VERTEX_SHELL' IN
    TYPEOF(m.sbwm_boundary[j]))))
    THEN
        result := FALSE;
        RETURN(result);
        (* A wireframe model is composed of WIRE_ and VERTEX_SHELLs *)
    END_IF;
END_REPEAT;

RETURN(result);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

FUNCTION constraints_param_b_spline
    (degree, up_knots, up_cp : INTEGER; knot_mult :
    LIST OF INTEGER; knots : LIST OF parameter_value) : BOOLEAN;
LOCAL
    result : BOOLEAN := TRUE;
    k,l,sum : INTEGER;
END_LOCAL;

sum := knot_mult[1];

REPEAT i := 2 TO up_knots;
    sum := sum + knot_mult[i];

```

```

END_REPEAT;

IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR
  (sum <> (degree + up_cp + 2)) THEN
  result := FALSE;
  RETURN(result);
END_IF;

k := knot_mult[1];

IF (k < 1) OR (k > degree + 1) THEN
  result := FALSE;
  RETURN(result);
END_IF;

REPEAT i := 2 TO up_knots;
  IF (knot_mult[i] < 1) OR (knots[i] <= knots[i-1]) THEN
    result := FALSE;
    RETURN(result);
  END_IF;

  k := knot_mult[i];

  IF (i < up_knots) AND (k > degree) THEN
    result := FALSE;
    RETURN(result);
  END_IF;

  IF (i = up_knots) AND (k > degree + 1) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
END_REPEAT;
RETURN(result);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION constraints_rectangular_composite_surface
  (s : rectangular_composite_surface) : BOOLEAN;

  REPEAT i := 1 TO s.n_u;
    REPEAT j := 1 TO s.n_v;
      IF NOT (('STRUCTURAL_FRAME_SCHEMA.B_SPLINE_SURFACE' IN TYPEOF
        (s.segments[i][j].parent_surface)) OR
        ('STRUCTURAL_FRAME_SCHEMA.RECTANGULAR_TRIMMED_SURFACE' IN
        TYPEOF

```

```

        (s.segments[i][j].parent_surface))) THEN
    RETURN(FALSE);
END_IF;
END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO s.n_u-1;
    REPEAT j := 1 TO s.n_v;
        IF s.segments[i][j].u_transition = discontinuous THEN
            RETURN(FALSE);
        END_IF;
    END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO s.n_u;
    REPEAT j := 1 TO s.n_v-1;
        IF s.segments[i][j].v_transition = discontinuous THEN
            RETURN(FALSE);
        END_IF;
    END_REPEAT;
END_REPEAT;
RETURN(TRUE);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION cross_product
    (arg1, arg2 : direction) : vector;
LOCAL
    mag : REAL;
    res : direction;
    v1,v2 : LIST[3:3] OF REAL;
    result : vector;
END_LOCAL;

IF ( NOT EXISTS (arg1) OR (arg1.dim = 2)) OR
   ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
    RETURN(?);
ELSE
    BEGIN
        v1 := normalise(arg1).direction_ratios;
        v2 := normalise(arg2).direction_ratios;
        res := dummy_gri || direction([(v1[2]*v2[3] - v1[3]*v2[2]),
            (v1[3]*v2[1] - v1[1]*v2[3]), (v1[1]*v2[2] - v1[2]*v2[1])]);
        mag := 0.0;
        REPEAT i := 1 TO 3;
            mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
        END_REPEAT;
    END
END

```

```

END_REPEAT;
IF (mag > 0.0) THEN
  result := dummy_gri || vector(res, SQRT(mag));
ELSE
  result := dummy_gri || vector(arg1, 0.0);
END_IF;
RETURN(result);
END;
END_IF;
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```

FUNCTION curve_weights_positive
  (b: rational_b_spline_curve) : BOOLEAN;
LOCAL
  result : BOOLEAN := TRUE;
END_LOCAL;

REPEAT i := 0 TO b.upper_index_on_control_points;
  IF b.weights[i] <= 0.0 THEN
    result := FALSE;
    RETURN(result);
  END_IF;
END_REPEAT;
RETURN(result);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION derive_dimensional_exponents
  (x : unit) : dimensional_exponents;

LOCAL
  i : INTEGER;
  result : dimensional_exponents :=
    dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
END_LOCAL;

IF 'STRUCTURAL_FRAME_SCHEMA.DERIVED_UNIT' IN
  TYPEOF(x) THEN      (* x is a derived unit *)
  REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements);

    result.length_exponent :=
      result.length_exponent +
      (x.elements[i].exponent * x.elements[i].unit.dimensions.length_exponent);

    result.mass_exponent :=
      result.mass_exponent +

```

```

        (x.elements[i].exponent * x.elements[i].unit.dimensions.mass_exponent);

result.time_exponent :=
    result.time_exponent +
    (x.elements[i].exponent * x.elements[i].unit.dimensions.time_exponent);

result.electric_current_exponent :=
    result.electric_current_exponent +
    (x.elements[i].exponent * x.elements[i].unit.dimensions.electric_current_exponent);

result.thermodynamic_temperature_exponent :=
    result.thermodynamic_temperature_exponent +
    (x.elements[i].exponent *
    x.elements[i].unit.dimensions.thermodynamic_temperature_exponent);

result.amount_of_substance_exponent :=
    result.amount_of_substance_exponent +
    (x.elements[i].exponent *
    x.elements[i].unit.dimensions.amount_of_substance_exponent);

result.luminous_intensity_exponent :=
    result.luminous_intensity_exponent +
    (x.elements[i].exponent *
    x.elements[i].unit.dimensions.luminous_intensity_exponent);

END_REPEAT;
ELSE (* x is a unitless or a named unit *)
    result := x.dimensions;
END_IF;
RETURN (result);
END_FUNCTION; (* based on STEP Part 41 2nd edition *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION dimension_of
    (item : geometric_representation_item) :
    dimension_count;
LOCAL
    x : SET OF representation;
    y : representation_context;
    dim : dimension_count;
END_LOCAL;
(* For cartesian_point, direction, or vector dimension is determined by
counting components. *)
IF 'STRUCTURAL_FRAME_SCHEMA.CARTESIAN_POINT' IN TYPEOF(item) THEN
    dim := SIZEOF(item\cartesian_point.coordinates);
RETURN(dim);

```

```

END_IF;
IF 'STRUCTURAL_FRAME_SCHEMA.DIRECTION' IN TYPEOF(item) THEN
    dim := SIZEOF(item\direction.direction_ratios);
    RETURN(dim);
END_IF;
IF 'STRUCTURAL_FRAME_SCHEMA.VECTOR' IN TYPEOF(item) THEN
    dim := SIZEOF(item\vector.orientation\direction.direction_ratios);
    RETURN(dim);
END_IF;
(* For all other types of geometric_representation_item dim is obtained
via context.
Find the set of representation in which the item is used. *)

x := using_representations(item);

(* Determines the dimension_count of the
geometric_representation_context. Note that the
RULE compatible_dimension ensures that the context_of_items
is of TYPE geometric_representation_context and has
the same dimension_count for all values of x.
The SET x is non-empty since this is required by WR1 of
representation_item. *)
y := x[1].context_of_items;
dim := y\geometric_representation_context.coordinate_space_dimension;
RETURN (dim);

END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```

FUNCTION dimensions_for_si_unit
    (n : si_unit_name) : dimensional_exponents;
CASE n OF
    metre : RETURN (dimensional_exponents (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    gram : RETURN (dimensional_exponents (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    second : RETURN (dimensional_exponents (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
    ampere : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
    kelvin : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    mole : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
    candela : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    radian : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    steradian : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    hertz : RETURN (dimensional_exponents (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    newton : RETURN (dimensional_exponents (1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    pascal : RETURN (dimensional_exponents (-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    joule : RETURN (dimensional_exponents (2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    watt : RETURN (dimensional_exponents (2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
    coulomb : RETURN (dimensional_exponents (0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));

```



```

    volt : RETURN (dimensional_exponents (2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
    farad : RETURN (dimensional_exponents (-2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0));
    ohm : RETURN (dimensional_exponents (2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
    siemens : RETURN (dimensional_exponents (-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
    weber : RETURN (dimensional_exponents (2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    tesla : RETURN (dimensional_exponents (0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    henry : RETURN (dimensional_exponents (2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
    degree_Celsius : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    lumen : RETURN (dimensional_exponents (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    lux : RETURN (dimensional_exponents (-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    becquerel : RETURN (dimensional_exponents (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    gray : RETURN (dimensional_exponents (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    sievert : RETURN (dimensional_exponents (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    OTHERWISE : RETURN(?);
    END_CASE;
END_FUNCTION; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

FUNCTION dot_product
    (arg1, arg2 : direction) : REAL;
    LOCAL
        scalar : REAL;
        vec1, vec2: direction;
        ndim : INTEGER;
    END_LOCAL;

    IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
        scalar := ?;

    ELSE
        IF (arg1.dim <> arg2.dim) THEN
            scalar := ?;

        ELSE
            BEGIN
                vec1 := normalise(arg1);
                vec2 := normalise(arg2);
                ndim := arg1.dim;
                scalar := 0.0;
                REPEAT i := 1 TO ndim;
                    scalar := scalar + vec1.direction_ratios[i]*vec2.direction_ratios[i];
                END_REPEAT;
            END;
        END_IF;
    END_IF;
    RETURN (scalar);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION edge_reversed
  (an_edge : edge) : oriented_edge;
LOCAL
  the_reverse : oriented_edge;
END_LOCAL;

IF ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_EDGE' IN TYPEOF (an_edge) )
  THEN
    the_reverse := dummy_tri ||
      edge(an_edge.edge_end, an_edge.edge_start) ||
      oriented_edge(an_edge\oriented_edge.edge_element,
        NOT (an_edge\oriented_edge.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
      edge(an_edge.edge_end, an_edge.edge_start) ||
      oriented_edge(an_edge, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION face_bound_reversed
  (a_face_bound : face_bound) : face_bound;
LOCAL
  the_reverse : face_bound ;
END_LOCAL;
IF ('STRUCTURAL_FRAME_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF
  (a_face_bound) ) THEN
  the_reverse := dummy_tri ||
    face_bound(a_face_bound\face_bound.bound,
      NOT (a_face_bound\face_bound.orientation))
    || face_outer_bound() ;
  ELSE
    the_reverse := dummy_tri ||
      face_bound(a_face_bound.bound, NOT(a_face_bound.orientation));
  END_IF;
  RETURN (the_reverse);
END_FUNCTION; (* STEP PART 42 (modified in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION face_reversed
  (a_face : face) : oriented_face;

```

```

LOCAL
  the_reverse : oriented_face ;
END_LOCAL;
IF ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_FACE' IN TYPEOF (a_face) ) THEN
  the_reverse := dummy_tri ||
    face(set_of_topology_reversed(a_face.bounds)) ||
    oriented_face(a_face\oriented_face.face_element,
      NOT (a_face\oriented_face.orientation)) ;
ELSE
  the_reverse := dummy_tri ||
    face(set_of_topology_reversed(a_face.bounds)) ||
    oriented_face(a_face, FALSE) ;
END_IF;
RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION first_proj_axis
  (z_axis, arg : direction) : direction;
LOCAL
  x_axis : direction;
  v      : direction;
  z      : direction;
  x_vec  : vector;
END_LOCAL;

IF (NOT EXISTS(z_axis)) THEN
  RETURN (?);
ELSE
  z := normalise(z_axis);
  IF NOT EXISTS(arg) THEN
    IF ((z.direction_ratios <> [1.0,0.0,0.0]) AND
      (z.direction_ratios <> [-1.0,0.0,0.0])) THEN
      v := dummy_gri || direction([1.0,0.0,0.0]);
    ELSE
      v := dummy_gri || direction([0.0,1.0,0.0]);
    END_IF;
  ELSE
    IF (arg.dim <> 3) THEN
      RETURN (?);
    END_IF;
    IF ((cross_product(arg,z).magnitude) = 0.0) THEN
      RETURN (?);
    ELSE
      v := normalise(arg);
    END_IF;
  END_IF;

```

```

END_IF;
x_vec := scalar_times_vector(dot_product(v, z), z);
x_axis := vector_difference(v, x_vec).orientation;
x_axis := normalise(x_axis);
END_IF;
RETURN(x_axis);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```

FUNCTION get_basis_surface
  (c : curve_on_surface) : SET[0:2] OF surface;
LOCAL
  surfs : SET[0:2] OF surface;
  n : INTEGER;
END_LOCAL;
surfs := [];
IF 'STRUCTURAL_FRAME_SCHEMA.PCURVE' IN TYPEOF (c) THEN
  surfs := [c\pcurve.basis_surface];
ELSE
  IF 'STRUCTURAL_FRAME_SCHEMA.SURFACE_CURVE' IN TYPEOF (c) THEN
    n := SIZEOF(c\surface_curve.associated_geometry);
    REPEAT i := 1 TO n;
      surfs := surfs +
        associated_surface(c\surface_curve.associated_geometry[i]);
    END_REPEAT;
  END_IF;
END_IF;
IF 'STRUCTURAL_FRAME_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF
(c) THEN
  n := SIZEOF(c\composite_curve.segments);
  surfs := get_basis_surface(c\composite_curve.segments[1].parent_curve);
  IF n > 1 THEN
    REPEAT i := 2 TO n;
      surfs := surfs * get_basis_surface(c\composite_curve.segments[i].parent_curve);
    END_REPEAT;
  END_IF;
END_IF;
RETURN(surfs);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* New for LPM/6 *)

```

FUNCTION get_description_value
  (obj : description_attribute_select) : text;
LOCAL
  description_bag : BAG OF description_attribute :=
    (USEDIN (obj,

```

```

        'STRUCTURAL_FRAME_SCHEMA.' +
        'DESCRIPTION_ATTRIBUTE.' +
        'DESCRIBED_ITEM'));
    END_LOCAL;
    IF SIZEOF (description_bag) = 1
        THEN RETURN (description_bag[1].attribute_value);
        ELSE RETURN (?);
    END_IF;
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

(* New for LPM/6 *)

```

FUNCTION get_id_value
(obj : id_attribute_select) : identifier;
LOCAL
    id_bag : BAG OF id_attribute :=
        (USEDIN (obj,
            'STRUCTURAL_FRAME_SCHEMA.' +
            'ID_ATTRIBUTE.' +
            'IDENTIFIED_ITEM'));
    END_LOCAL;
    IF SIZEOF (id_bag) = 1
        THEN RETURN (id_bag[1].attribute_value);
        ELSE RETURN (?);
    END_IF;
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

(* New for LPM/6 *)

(* created for LPM/6 based on STEP Part 41 FUNCTION get_name_value *)

```

FUNCTION get_instance_id
(obj : select_data_item) : globally_unique_id;
LOCAL
    id_bag : BAG OF managed_data_item :=
        (USEDIN (obj,
            'STRUCTURAL_FRAME_SCHEMA.' +
            'MANAGED_DATA_ITEM.' +
            'DATA_ITEM'));
    n : INTEGER;
    END_LOCAL;

    n := SIZEOF(id_bag);

    CASE n OF
        0 : RETURN ('UNMANAGED');
        1 : RETURN (id_bag[1].instance_id);
        OTHERWISE : RETURN (?);
    END_CASE;

```

END_CASE;

END_FUNCTION;

(* New for LPM/6 *)

FUNCTION get_item_cost_code

(item : structural_frame_item) : BAG OF label;

LOCAL

i : INTEGER;

cost_codes : BAG OF label := [];

item_assignment : BAG OF item_cost_code_assigned :=

(USEDIN (item,
'STRUCTURAL_FRAME_SCHEMA.' +
'ITEM_COST_CODE_ASSIGNED.' +
'COSTED_ITEM'));

END_LOCAL;

IF SIZEOF (item_assignment) > 0 THEN

REPEAT i := 1 to HIINDEX (item_assignment);

cost_codes := cost_codes + item_assignment[i].code\item_cost_code.cost_code;

END_REPEAT;

END_IF;

RETURN (cost_codes);

END_FUNCTION;

(* New for LPM/6 *)

FUNCTION get_item_ref

(item : structural_frame_item) : BAG OF identifier;

LOCAL

i : INTEGER;

refs : BAG OF identifier := [];

item_assignment : BAG OF item_reference_assigned :=

(USEDIN (item,
'STRUCTURAL_FRAME_SCHEMA.' +
'ITEM_REFERENCE_ASSIGNED.' +
'ASSIGNED_TO_ITEM'));

END_LOCAL;

IF SIZEOF (item_assignment) > 0 THEN

REPEAT i := 1 to HIINDEX (item_assignment);

refs := refs + item_assignment[i].assigned_reference\item_reference.ref;

END_REPEAT;

END_IF;

```

RETURN (refs);
END_FUNCTION;

```

```
(* New for LPM/6 *)
```

```

FUNCTION get_name_value
(obj : name_attribute_select) : label;
LOCAL
  name_bag : BAG OF name_attribute :=
    (USEDIN (obj,
      'STRUCTURAL_FRAME_SCHEMA.' +
      'NAME_ATTRIBUTE.' +
      'NAMED_ITEM'));
END_LOCAL;
IF SIZEOF (name_bag) = 1
  THEN RETURN (name_bag[1].attribute_value);
  ELSE RETURN (?);
END_IF;
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

```
(* New for LPM/6 *)
```

```

FUNCTION get_role
(obj : role_select) : object_role;
LOCAL
  role_bag : BAG OF role_association:=
    (USEDIN (obj,
      'STRUCTURAL_FRAME_SCHEMA.' +
      'ROLE_ASSOCIATION.' +
      'ITEM_WITH_ROLE'));
END_LOCAL;
IF SIZEOF (role_bag) = 1
  THEN RETURN (role_bag[1].role);
  ELSE RETURN (?);
END_IF;
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

```

FUNCTION item_in_context
  (item : representation_item; cntxt : representation_context) : BOOLEAN;
LOCAL
  i : INTEGER;
  y : BAG OF representation_item;
END_LOCAL;

```

```
(* If there is one or more representation using both the item and cntxt return true. *)
```

```

IF SIZEOF(USEDIN(item,'STRUCTURAL_FRAME_SCHEMA.REPRESENTATION.ITEMS')
  * cntxt.representations_in_context) > 0 THEN
  RETURN (TRUE);

  (* Determine the bag of representation_items that reference item. *)

ELSE
  y := QUERY(z <* USEDIN (item , ") |
  'STRUCTURAL_FRAME_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

  (* Ensure that the bag is not empty. *)

  IF SIZEOF(y) > 0 THEN

    (* For each element in the bag *)

    REPEAT i := 1 TO HIINDEX(y);

    (* Check to see it is an item in the input cntxt. *)

    IF item_in_context(y[i], cntxt) THEN
      RETURN (TRUE);
    END_IF;
    END_REPEAT;
    END_IF;
  END_IF;

  (* Return false when all possible branches have been checked with no success. *)
  RETURN (FALSE);
END_FUNCTION; (* STEP Part 43 (unchanged in 2nd edition) *)

```

```

FUNCTION leap_year
  (year : year_number) : BOOLEAN;

  IF (((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR
    ((year MOD 400) = 0) THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END_IF;
END_FUNCTION; (* STEP Part 41 unchanged in 2nd edition *)

```

```

FUNCTION list_face_loops
  (f: face) : LIST[0:?] OF loop;

```



```

LOCAL
  loops : LIST[0:?] OF loop := [];
END_LOCAL;

REPEAT i := 1 TO SIZEOF(f.bounds);
  loops := loops +(f.bounds[i].bound);
END_REPEAT;

RETURN(loops);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION list_loop_edges
  (l: loop): LIST[0:?] OF edge;
LOCAL
  edges : LIST[0:?] OF edge := [];
END_LOCAL;

IF 'STRUCTURAL_FRAME_SCHEMA.EDGE_LOOP' IN TYPEOF(l) THEN
  REPEAT i := 1 TO SIZEOF(l.path.edge_list);
    edges := edges + (l.path.edge_list[i].edge_element);
  END_REPEAT;
END_IF;

RETURN(edges);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION list_of_topology_reversed
  (a_list : list_of_reversible_topology_item) : list_of_reversible_topology_item;
LOCAL
  the_reverse : list_of_reversible_topology_item;
END_LOCAL;

the_reverse := [];
REPEAT i := 1 TO SIZEOF (a_list);
  the_reverse := topology_reversed (a_list [i]) + the_reverse;
END_REPEAT;

RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION list_to_array
  (lis : LIST [0:?] OF GENERIC : T; low,u : INTEGER) : ARRAY[low:u] OF GENERIC :
  T;
LOCAL

```

```

    n : INTEGER;
    res : ARRAY [low:u] OF GENERIC : T;
END_LOCAL;

n := SIZEOF(lis);
IF (n <> (u-low + 1)) THEN
    RETURN(?);
ELSE
    REPEAT i := 1 TO n;
        res[low+i-1] := lis[i];
    END_REPEAT;
    RETURN(res);
END_IF;
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION list_to_set
    (l : LIST [0:?] OF GENERIC:T) : SET OF GENERIC:T;
LOCAL
    s : SET OF GENERIC:T := [];
END_LOCAL;

REPEAT i := 1 TO SIZEOF(l);
    s := s + l[i];
END_REPEAT;

RETURN(s);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION make_array_of_array
    (lis : LIST[1:?] OF LIST [1:?] OF GENERIC : T;
     low1, u1, low2, u2 : INTEGER);
    ARRAY OF ARRAY OF GENERIC : T;
LOCAL
    res : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
END_LOCAL;

```

(* Check input dimensions for consistency *)

```

IF (u1-low1+1) <> SIZEOF(lis) THEN
    RETURN (?);
END_IF;
IF (u2 - low2 + 1 ) <> SIZEOF(lis[1]) THEN
    RETURN (?) ;
END_IF;

```

(* Initialise res with values from lis[1] *)

```

    res := [list_to_array(lis[1], low2, u2) : (u1-low1 + 1)];
    REPEAT i := 2 TO HIINDEX(lis);
        IF (u2-low2+1) <> SIZEOF(lis[i]) THEN
            RETURN (?);
        END_IF;
        res[low1+i-1] := list_to_array(lis[i], low2, u2);
    END_REPEAT;

    RETURN (res);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* New for LPM/6 *)

```

FUNCTION make_array_of_array_of_array
    (lis : LIST[1:?] OF
        LIST [1:?] OF LIST [1:?] OF GENERIC : T;
        low1, u1, low2, u2, low3, u3 : INTEGER);
    ARRAY OF ARRAY OF ARRAY OF GENERIC : T;
LOCAL
    res : ARRAY[low1:u1] OF ARRAY [low2:u2] OF
        ARRAY[low3:u3] OF GENERIC : T;
END_LOCAL;

(* Check input dimensions for consistency *)
IF (u1-low1+1) <> SIZEOF(lis) THEN
    RETURN (?);
END_IF;
IF (u2-low2+1) <> SIZEOF(lis[1]) THEN
    RETURN (?);
END_IF;

(* Initialise res with values from lis[1] *)
res := [make_array_of_array(lis[1], low2, u2, low3, u3) : (u1-low1 + 1)];
REPEAT i := 2 TO HIINDEX(lis);
    IF (u2-low2+1) <> SIZEOF(lis[i]) THEN
        RETURN (?);
    END_IF;
    res[low1+i-1] := make_array_of_array(lis[i], low2, u2, low3, u3);
END_REPEAT;
RETURN (res);
END_FUNCTION; (* STEP Part 42 (new for 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION mixed_loop_type_set
    (l: SET[0:?] OF loop): LOGICAL;
LOCAL
    poly_loop_type: LOGICAL;

```

```

END_LOCAL;
IF(SIZEOF(l) <= 1) THEN
  RETURN(FALSE);
END_IF;
poly_loop_type := ('STRUCTURAL_FRAME_SCHEMA.POLY_LOOP' IN TYPEOF(l[1]));
REPEAT i := 2 TO SIZEOF(l);
  IF(('STRUCTURAL_FRAME_SCHEMA.POLY_LOOP' IN TYPEOF(l[i])) <>
    poly_loop_type) THEN
    RETURN(TRUE);
  END_IF;
END_REPEAT;
RETURN(FALSE);
END_FUNCTION; (* STEP Part 42 (modified for 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION normalise
  (arg : vector_or_direction) : vector_or_direction;
LOCAL
  ndim : INTEGER;
  v : direction;
  result : vector_or_direction;
  vec : vector;
  mag : REAL;
END_LOCAL;

IF NOT EXISTS (arg) THEN
  result := ?;
(* When FUNCTION is called with invalid data a NULL result is returned *)
ELSE
  ndim := arg.dim;
  IF 'STRUCTURAL_FRAME_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
    BEGIN
      v := dummy_gri || direction(arg.orientation.direction_ratios);
      IF arg.magnitude = 0.0 THEN
        RETURN(?);
      ELSE
        vec := dummy_gri || vector (v, 1.0);
        END_IF;
      END;
    ELSE
      v := dummy_gri || direction (arg.direction_ratios);
      END_IF;
      mag := 0.0;
      REPEAT i := 1 TO ndim;
        mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
      END_REPEAT;

```

```

IF mag > 0.0 THEN
  mag := SQRT(mag);
  REPEAT i := 1 TO ndim;
    v.direction_ratios[i] := v.direction_ratios[i]/mag;
  END_REPEAT;
  IF 'STRUCTURAL_FRAME_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
    vec.orientation := v;
    result := vec;
  ELSE
    result := v;
  END_IF;
ELSE
  RETURN(?);
END_IF;
END_IF;
RETURN (result);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* New for LPM/6 *)

```

FUNCTION open_shell_reversed
  ( a_shell : open_shell ) : oriented_open_shell;
LOCAL
  the_reverse : oriented_open_shell;
END_LOCAL;
IF ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF (a_shell)
) THEN
  the_reverse := dummy_tri ||
    connected_face_set (
      a_shell\connected_face_set.cfs_faces) ||
    open_shell () || oriented_open_shell(
      a_shell\oriented_open_shell.open_shell_element,
      (NOT (a_shell\oriented_open_shell.orientation)));
ELSE
  the_reverse := dummy_tri ||
    connected_face_set (
      a_shell\connected_face_set.cfs_faces) ||
    open_shell () || oriented_open_shell (a_shell, FALSE);
END_IF;
RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (new for 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION orthogonal_complement
  (vec : direction) : direction;
LOCAL

```

```

    result : direction ;
END_LOCAL;

IF (vec.dim <> 2) OR NOT EXISTS (vec) THEN
    RETURN(?);
ELSE
    result := dummy_gri || direction([-vec.direction_ratios[2],
                                     vec.direction_ratios[1]]);
    RETURN(result);
END_IF;
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```

FUNCTION path_head_to_tail
    (a_path : path) : LOGICAL;
LOCAL
    n : INTEGER;
    p : LOGICAL := TRUE;
END_LOCAL;

    n := SIZEOF (a_path.edge_list);
    REPEAT i := 2 TO n;
    p := p AND (a_path.edge_list[i-1].edge_end :=:
                a_path.edge_list[i].edge_start);
    END_REPEAT;

    RETURN (p);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION path_reversed
    (a_path : path) : oriented_path;
LOCAL
    the_reverse : oriented_path ;
END_LOCAL;
IF ('STRUCTURAL_FRAME_SCHEMA.ORIENTED_PATH' IN TYPEOF (a_path) ) THEN
    the_reverse := dummy_tri ||
        path(list_of_topology_reversed (a_path.edge_list)) ||
        oriented_path(a_path\oriented_path.path_element,
                      NOT(a_path\oriented_path.orientation)) ;
ELSE
    the_reverse := dummy_tri ||
        path(list_of_topology_reversed (a_path.edge_list)) ||
        oriented_path(a_path, FALSE);
END_IF;

```

```

RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```
(* New for LPM/6 *)
```

```

FUNCTION same_side
  (plane_pts : LIST [3:3] of cartesian_point;
   test_points : LIST [2:?] of cartesian_point) : BOOLEAN;
LOCAL
  val1, val2 : REAL;
  n : INTEGER;
END_LOCAL;

IF (plane_pts[1].dim = 2) OR (test_points[1].dim = 2) THEN
  RETURN(?);
END_IF;
n := SIZEOF(test_points);
val1 := above_plane(plane_pts[1], plane_pts[2], plane_pts[3],
  test_points[1] );
REPEAT i := 2 TO n;
  val2 := above_plane(plane_pts[1], plane_pts[2], plane_pts[3],
    test_points[i] );
  IF (val1*val2 <= 0.0) THEN
    RETURN(FALSE);
  END_IF;
END_REPEAT;
RETURN(TRUE);
END_FUNCTION;

```

```
(* Modified for LPM/6 *)
```

```

FUNCTION scalar_times_vector
  (scalar : REAL; vec : vector_or_direction) : vector;
LOCAL
  v : direction;
  mag : REAL;
  result : vector;
END_LOCAL;

IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
  RETURN (?) ;
ELSE
  IF 'STRUCTURAL_FRAME_SCHEMA.VECTOR' IN TYPEOF (vec) THEN
    v := dummy_gri || direction(vec.orientation.direction_ratios);
    mag := scalar * vec.magnitude;
  ELSE
    v := dummy_gri || direction(vec.direction_ratios);

```

```

    mag := scalar;
  END_IF;
  IF (mag < 0.0 ) THEN
    REPEAT i := 1 TO SIZEOF(v.direction_ratios);
      v.direction_ratios[i] := -v.direction_ratios[i];
    END_REPEAT;
    mag := -mag;
  END_IF;
  result := dummy_gri || vector(normalise(v), mag);
END_IF;
RETURN (result);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION second_proj_axis
  (z_axis, x_axis, arg: direction) : direction;
LOCAL
  y_axis : vector;
  v      : direction;
  temp   : vector;
END_LOCAL;

  IF NOT EXISTS(arg) THEN
    v := dummy_gri || direction([0.0,1.0,0.0]);
  ELSE
    v := arg;
  END_IF;

  temp := scalar_times_vector(dot_product(v, z_axis), z_axis);
  y_axis := vector_difference(v, temp);
  temp := scalar_times_vector(dot_product(v, x_axis), x_axis);
  y_axis := vector_difference(y_axis, temp);
  y_axis := normalise(y_axis);
  RETURN(y_axis.orientation);
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```

FUNCTION set_of_topology_reversed
  (a_set : set_of_reversible_topology_item) : set_of_reversible_topology_item;
LOCAL
  the_reverse : set_of_reversible_topology_item;
END_LOCAL;

  the_reverse := [];
  REPEAT i := 1 TO SIZEOF (a_set);
    the_reverse := the_reverse + topology_reversed (a_set [i]);
  END_REPEAT;

```



```

END_REPEAT;

RETURN (the_reverse);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* Modified for LPM/6 *)

```

FUNCTION shell_reversed
  (a_shell : shell) : shell;
  IF ('STRUCTURAL_FRAME_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
    RETURN (open_shell_reversed (a_shell));
  ELSE
    IF ('STRUCTURAL_FRAME_SCHEMA.CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
      RETURN (closed_shell_reversed (a_shell));
    ELSE
      RETURN (?);
    END_IF;
  END_IF;
END_FUNCTION; (* STEP Part 42 (modified in 2nd edition) *)

```

```

FUNCTION surface_weights_positive
  (b: rational_b_spline_surface) : BOOLEAN;
  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;

  REPEAT i := 0 TO b.u_upper;
    REPEAT j := 0 TO b.v_upper;
      IF (b.weights[i][j] <= 0.0) THEN
        result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
END_REPEAT;
RETURN(result);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

```

FUNCTION topology_reversed
  (an_item : reversible_topology) : reversible_topology;

  IF ('STRUCTURAL_FRAME_SCHEMA.EDGE' IN TYPEOF (an_item)) THEN
    RETURN (edge_reversed (an_item));
  END_IF;

```

```

IF ('STRUCTURAL_FRAME_SCHEMA.PATH' IN TYPEOF (an_item)) THEN
    RETURN (path_reversed (an_item));
END_IF;

IF ('STRUCTURAL_FRAME_SCHEMA.FACE_BOUND' IN TYPEOF (an_item)) THEN
    RETURN (face_bound_reversed (an_item));
END_IF;

IF ('STRUCTURAL_FRAME_SCHEMA.FACE' IN TYPEOF (an_item)) THEN
    RETURN (face_reversed (an_item));
END_IF;

IF ('STRUCTURAL_FRAME_SCHEMA.SHELL' IN TYPEOF (an_item)) THEN
    RETURN (shell_reversed (an_item));
END_IF;

IF ('SET' IN TYPEOF (an_item)) THEN
    RETURN (set_of_topology_reversed (an_item));
END_IF;

IF ('LIST' IN TYPEOF (an_item)) THEN
    RETURN (list_of_topology_reversed (an_item));
END_IF;

RETURN (?);
END_FUNCTION; (* STEP Part 42 (unchanged in 2nd edition) *)

```

(* **New for LPM/6** - required for FUNCTION using_representations *)

```

FUNCTION using_items
    (item : founded_item_select;
     checked_items: SET OF founded_item_select)
    : SET OF founded_item_select;
LOCAL
    new_check_items : SET OF founded_item_select;
    result_items : SET OF founded_item_select;
    next_items : SET OF founded_item_select;
END_LOCAL;
result_items := [];
new_check_items := checked_items + item;
(* Find the set of representation_items or founded_items
   in which item is used directly. *)
next_items := QUERY(z <* bag_to_set( USEDIN(item , "")) |
    ('STRUCTURAL_FRAME_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) OR
    ('STRUCTURAL_FRAME_SCHEMA.FOUNDED_ITEM' IN TYPEOF(z)));
(* If the set of next_items is not empty; *)
IF SIZEOF(next_items) > 0 THEN

```

```

(* For each element in the set, find the using_items recursively *)
REPEAT i := 1 TO HIINDEX(next_items);
  (* Check for loop in data model, i.e. one of the next_items
    occurred earlier in the set of check_items; *)
  IF NOT(next_items[i] IN new_check_items) THEN
    result_items := result_items + next_items[i] +
      using_items(next_items[i],new_check_items);
  END_IF;
END_REPEAT;
END_IF;
(* return the set of representation_items or founded_items
  in which the input item is used directly and indirectly. *)
RETURN (result_items);
END_FUNCTION; (* STEP Part 43 2nd Edition *)

```

(* Modified for LPM/6 *)

```

FUNCTION using_representations
  (item : founded_item_select) : SET OF representation;
LOCAL
  results      : SET OF representation;
  result_bag    : BAG OF representation;
  intermediate_items : SET OF founded_item_select;
END_LOCAL;
(* Find the representations in which the item is used and add to the
  results set. *)
results := [];
result_bag :=
  USEDIN(item,'STRUCTURAL_FRAME_SCHEMA.REPRESENTATION.ITEMS');
IF SIZEOF(result_bag) > 0 THEN
  REPEAT i := 1 TO HIINDEX(result_bag);
    results := results + result_bag[i];
  END_REPEAT;
END_IF;
(* Find all representation_items or founded_items
  by which item is referenced directly or indirectly. *)
intermediate_items := using_items(item,[]);
(* If the set of intermediate items is not empty; *)
IF SIZEOF(intermediate_items) > 0 THEN
  (* For each element in the set, add the
    representations of that element. *)
  REPEAT i := 1 TO HIINDEX(intermediate_items);
    result_bag := USEDIN(intermediate_items[i],
      'STRUCTURAL_FRAME_SCHEMA.REPRESENTATION.ITEMS');
    IF SIZEOF(result_bag) > 0 THEN
      REPEAT j := 1 TO HIINDEX(result_bag);
        results := results + result_bag[j];
      END_REPEAT;
    END_REPEAT;
  END_REPEAT;
END_IF;

```

```

    END_IF;
    END_REPEAT;
    END_IF;
    (* Return the set of representation in which the input item is
       used directly and indirectly (through intervening
       representation_items or founded items). *)
    RETURN (results);
    END_FUNCTION; (* STEP Part 43 (modified in 2nd edition) *)

```

```

FUNCTION unique_data_item
    (item : select_data_item): LOGICAL;
LOCAL
    bag_of_managed_items : BAG OF managed_data_item;
END_LOCAL;

IF ('STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_DELETED' IN TYPEOF(item))
    THEN
        RETURN (UNKNOWN);
    END_IF;

    (* find the managed_data_item in which the item is used
       and add to the bag_of_managed_items *)

    bag_of_managed_items := USEDIN(item,
        'STRUCTURAL_FRAME_SCHEMA.MANAGED_DATA_ITEM.DATA_ITEM');

    IF SIZEOF (bag_of_managed_items) = 1 THEN
        RETURN (TRUE);
    ELSE
        RETURN (FALSE);
    END_IF;

    END_FUNCTION; (* created for LPM/5 *)

```

```

(* Modified for LPM/6 *)
FUNCTION valid_calendar_date
    (date : calendar_date) : LOGICAL;
CASE date.month_component OF
    1 : RETURN({ 1 <= date.day_component <= 31 });
    2 : BEGIN
        IF (leap_year(date.year_component)) THEN
            RETURN({ 1 <= date.day_component <= 29 });
        ELSE
            RETURN({ 1 <= date.day_component <= 28 });
        END_IF;
    END;

```

```

3 : RETURN({ 1 <= date.day_component <= 31 });
4 : RETURN({ 1 <= date.day_component <= 30 });
5 : RETURN({ 1 <= date.day_component <= 31 });
6 : RETURN({ 1 <= date.day_component <= 30 });
7 : RETURN({ 1 <= date.day_component <= 31 });
8 : RETURN({ 1 <= date.day_component <= 31 });
9 : RETURN({ 1 <= date.day_component <= 30 });
10 : RETURN({ 1 <= date.day_component <= 31 });
11 : RETURN({ 1 <= date.day_component <= 30 });
12 : RETURN({ 1 <= date.day_component <= 31 });
END_CASE;
RETURN (FALSE);
END_FUNCTION; (* STEP Part 41 2nd edition *)

```

(* Added for LPM/6 *)

```

FUNCTION valid_measure_value
(m : measure_value) : BOOLEAN;
IF ('REAL' IN TYPEOF (m)) THEN
RETURN (m > 0.0);
ELSE
IF ('INTEGER' IN TYPEOF (m)) THEN
RETURN (m > 0);
ELSE
RETURN (TRUE);
END_IF;
END_IF;
END_FUNCTION; (* New in STEP Part 43 2nd edition *)

```

```

FUNCTION valid_time
(time: local_time) : BOOLEAN;
IF EXISTS (time.second_component) THEN
RETURN (EXISTS (time.minute_component));
ELSE
RETURN (TRUE);
END_IF;
END_FUNCTION; (* STEP Part 41 (unchanged in 2nd edition) *)

```

```

FUNCTION valid_units
( m : measure_with_unit ) : BOOLEAN;

IF 'STRUCTURAL_FRAME_SCHEMA.LENGTH_MEASURE' IN
TYPEOF ( m.value_component ) THEN
IF derive_dimensional_exponents ( m.unit_component ) <>
dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
RETURN (FALSE);
END_IF;

```

END_IF;

```
IF 'STRUCTURAL_FRAME_SCHEMA.MASS_MEASURE' IN
  TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
    dimensional_exponents ( 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
```

```
IF 'STRUCTURAL_FRAME_SCHEMA.TIME_MEASURE' IN
  TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
    dimensional_exponents ( 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
```

```
IF
'Structural_Frame_Schema.Thermodynamic_Temperature_Measure'
  IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
    dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
```

```
IF 'STRUCTURAL_FRAME_SCHEMA.PLANE_ANGLE_MEASURE' IN
  TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
    dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
```

```
IF 'STRUCTURAL_FRAME_SCHEMA.SOLID_ANGLE_MEASURE' IN
  TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
    dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
```

```
IF 'STRUCTURAL_FRAME_SCHEMA.AREA_MEASURE' IN
  TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
    dimensional_exponents ( 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
```

```

        RETURN (FALSE);
    END_IF;
END_IF;

IF 'STRUCTURAL_FRAME_SCHEMA.VOLUME_MEASURE' IN
    TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
        dimensional_exponents ( 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
    END_IF;
END_IF;

IF 'STRUCTURAL_FRAME_SCHEMA.RATIO_MEASURE' IN
    TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
        dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
    END_IF;
END_IF;

IF 'STRUCTURAL_FRAME_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
    TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
        dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
    END_IF;
END_IF;

IF 'STRUCTURAL_FRAME_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN
    TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
        dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
        RETURN (FALSE);
    END_IF;
END_IF;

RETURN (TRUE);
END_FUNCTION; (* STEP Part 41 (unchanged in 2nd edition) *)

```

(* [Modified for LPM/6](#) *)

```

FUNCTION vector_difference
    (arg1, arg2 : vector_or_direction) : vector;
LOCAL
    result      : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;

```

```

    ndim      : INTEGER;
END_LOCAL;

IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <> arg2.dim)
    THEN
    RETURN (?);
ELSE
    BEGIN
        IF 'STRUCTURAL_FRAME_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
            mag1 := arg1.magnitude;
            vec1 := arg1.orientation;
        ELSE
            mag1 := 1.0;
            vec1 := arg1;
        END_IF;
        IF 'STRUCTURAL_FRAME_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
            mag2 := arg2.magnitude;
            vec2 := arg2.orientation;
        ELSE
            mag2 := 1.0;
            vec2 := arg2;
        END_IF;
        vec1 := normalise (vec1);
        vec2 := normalise (vec2);
        ndim := SIZEOF(vec1.direction_ratios);
        mag := 0.0;
        res := dummy_gri || direction(vec1.direction_ratios);
        REPEAT i := 1 TO ndim;
            res.direction_ratios[i] := mag1*vec1.direction_ratios[i] +
                                    mag2*vec2.direction_ratios[i];
            mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
        END_REPEAT;
        IF (mag > 0.0 ) THEN
            result := dummy_gri || vector( res, SQRT(mag));
        ELSE
            result := dummy_gri || vector( vec1, 0.0);
        END_IF;
    END;
END_IF;
RETURN (result);
END_FUNCTION; (* STEP Part 42 (modified for 2nd edition) *)

```

```

FUNCTION volume_weights_positive
    (b: rational_b_spline_volume): BOOLEAN;
LOCAL

```



```
    result : BOOLEAN := TRUE;
END_LOCAL;

REPEAT i := 0 TO b.u_upper;
  REPEAT j := 0 TO b.v_upper;
    REPEAT k := 0 TO b.w_upper;
      IF (b.weights[i][j][k] <= 0.0) THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
  END_REPEAT;
END_REPEAT;
RETURN(result);
END_FUNCTION; (* STEP Part 42 (new in 2nd edition) *)

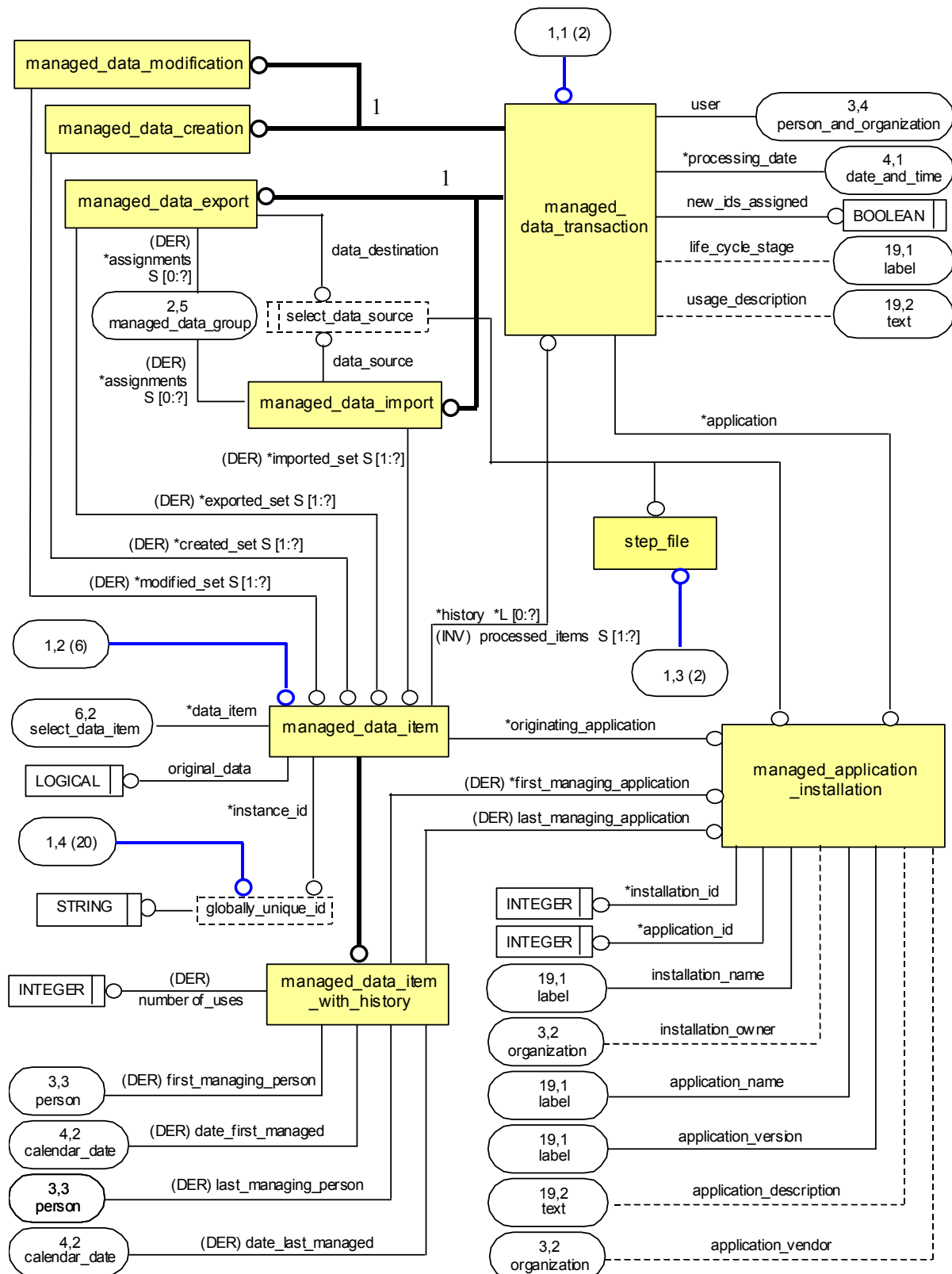
END_SCHEMA;
(* End of Structural Frame Schema (LPM/6) *)
```

This page is blank

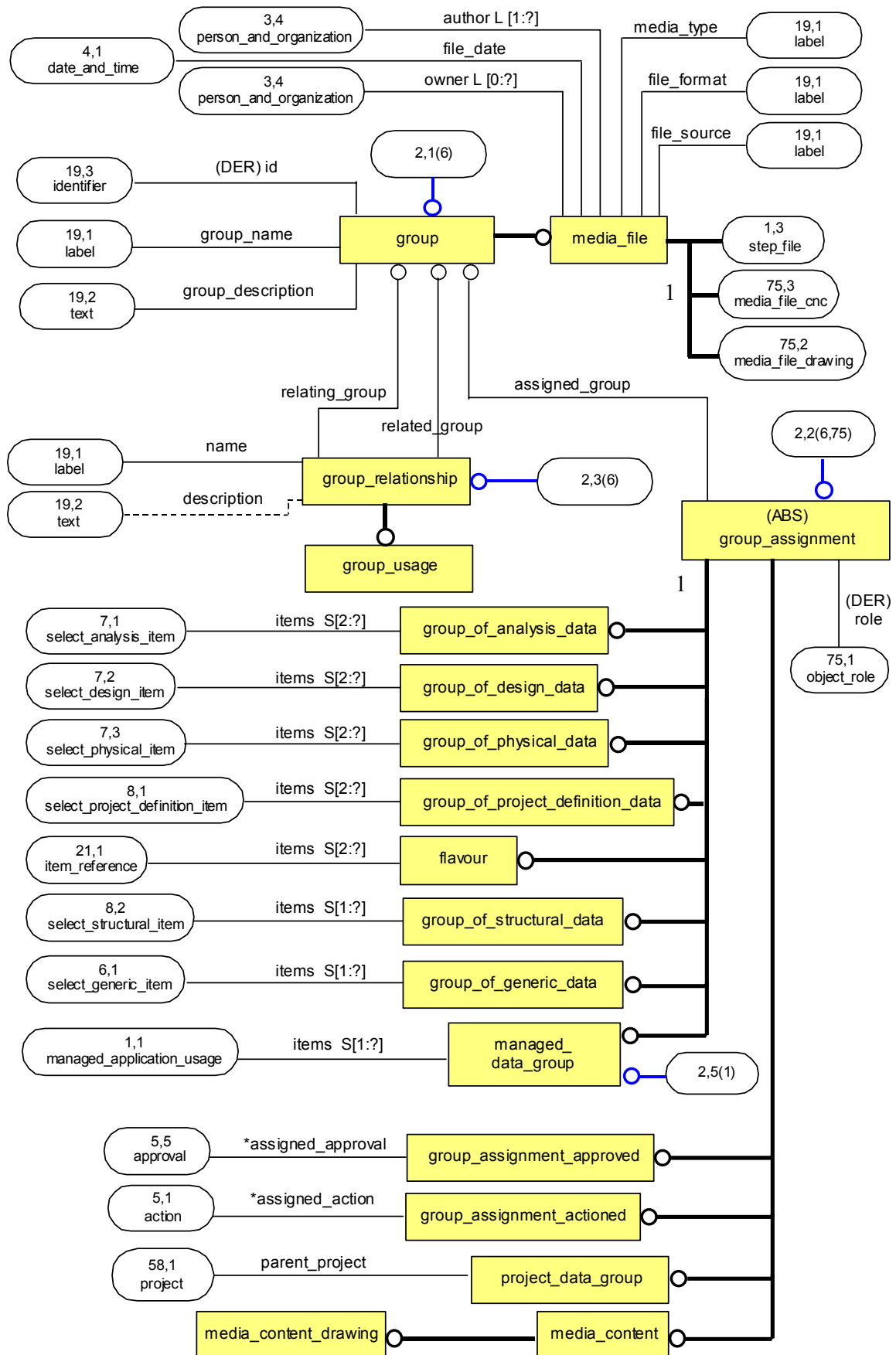
APPENDIX B LPM/6 EXPRESS-G DIAGRAMS

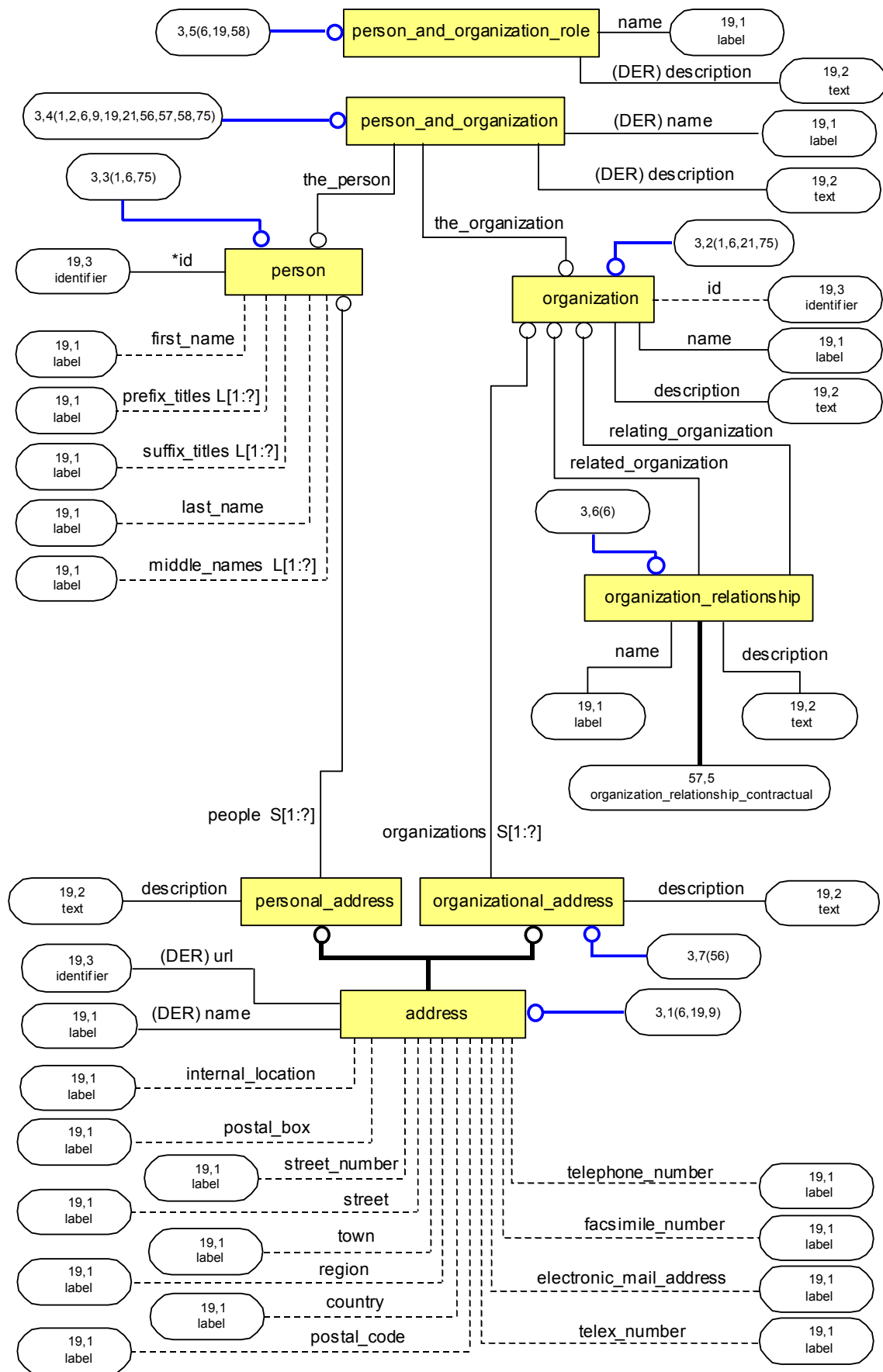
The following 82 pages illustrate the long form of the LPM/6 schema using the EXPRESS-G notation. See the *Implementation Guide* for details of the EXPRESS-G notation and its use in CIS/2.

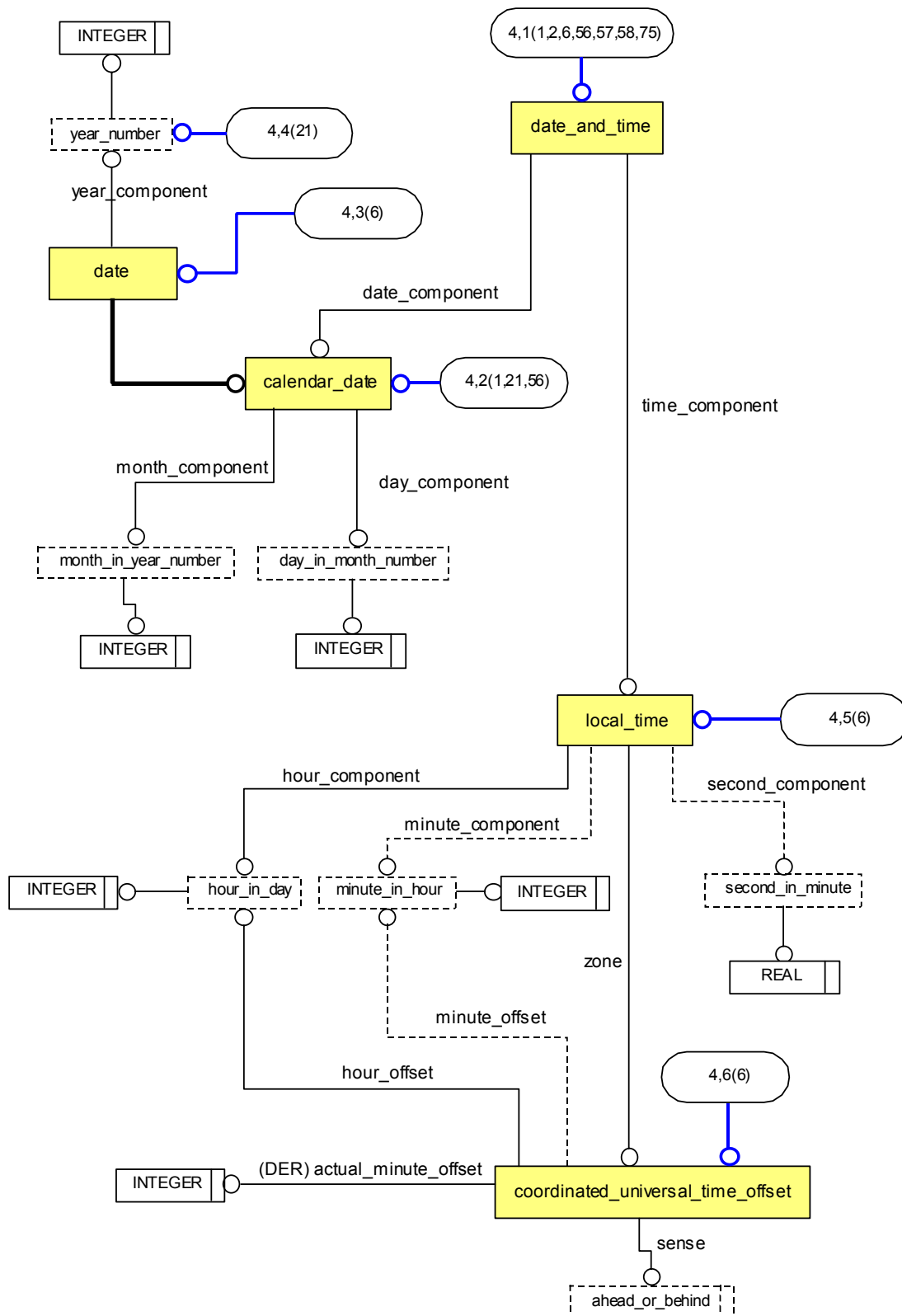
LPM/6 EXPRESS-G Diagram 1 of 82 (Modified for 2nd Edition)

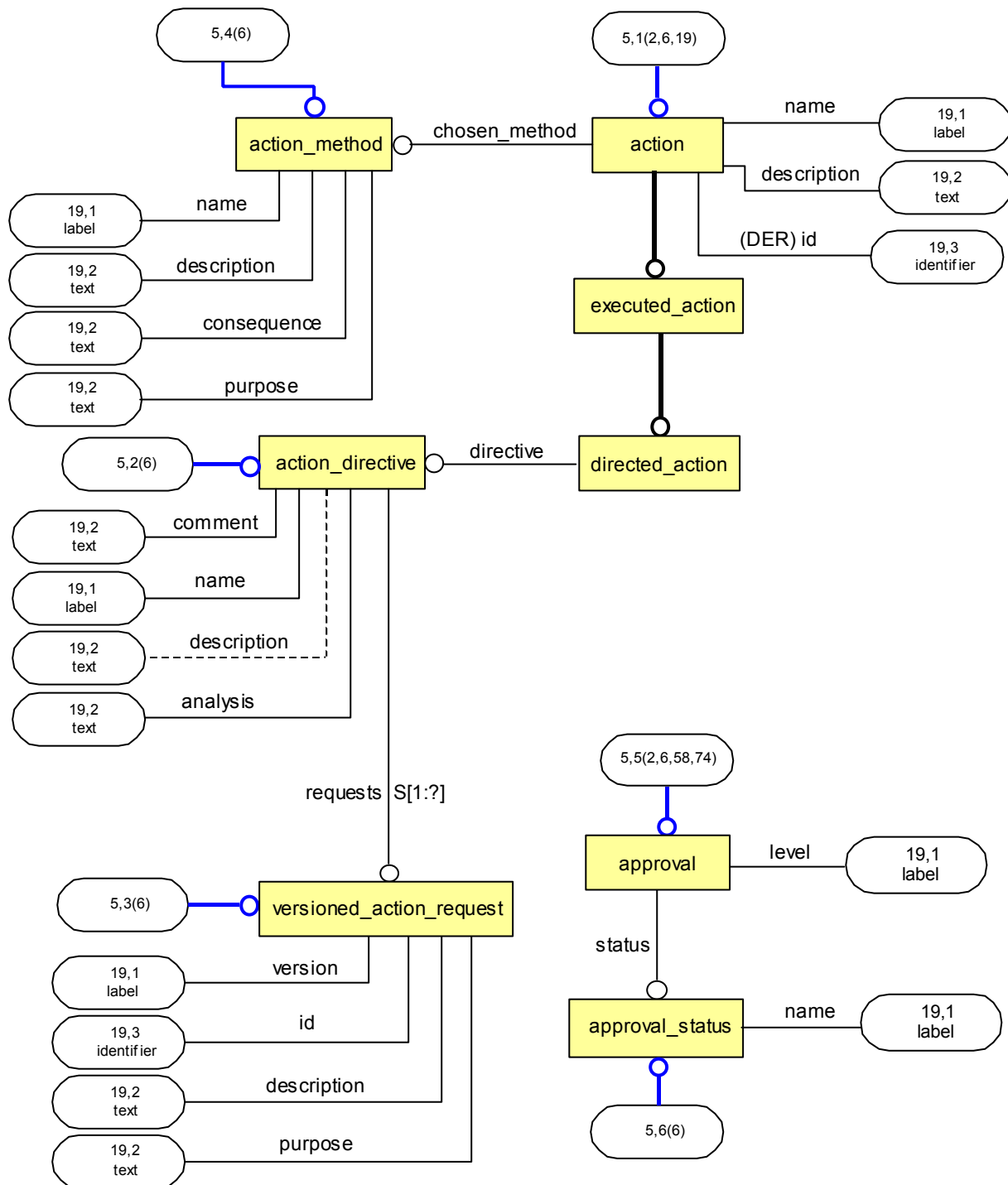


LPM/6 EXPRESS-G Diagram 2 of 82 (Modified for 2nd Edition)

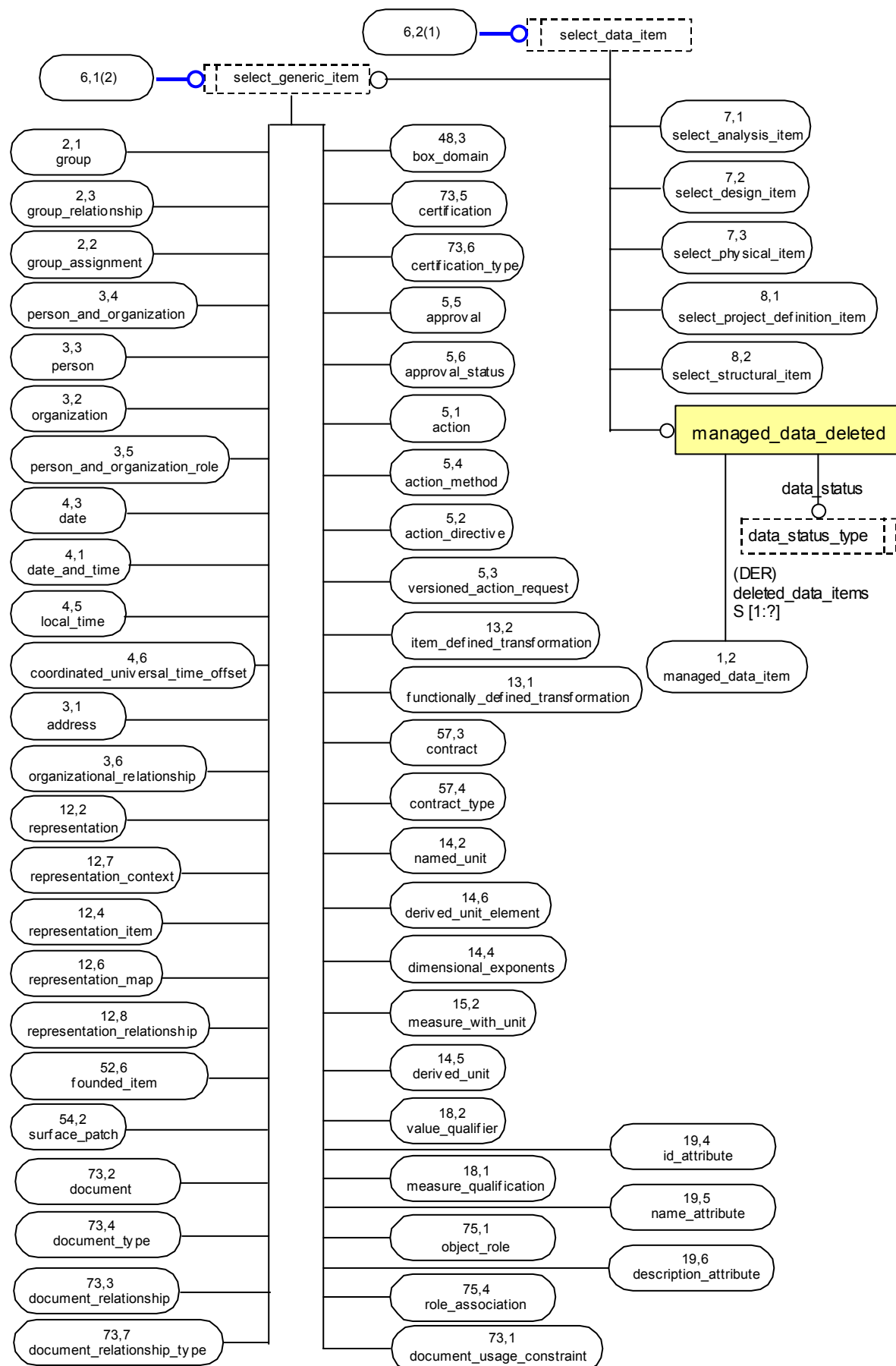


LPM/6 EXPRESS-G Diagram 3 of 82 (Modified for 2nd Edition)

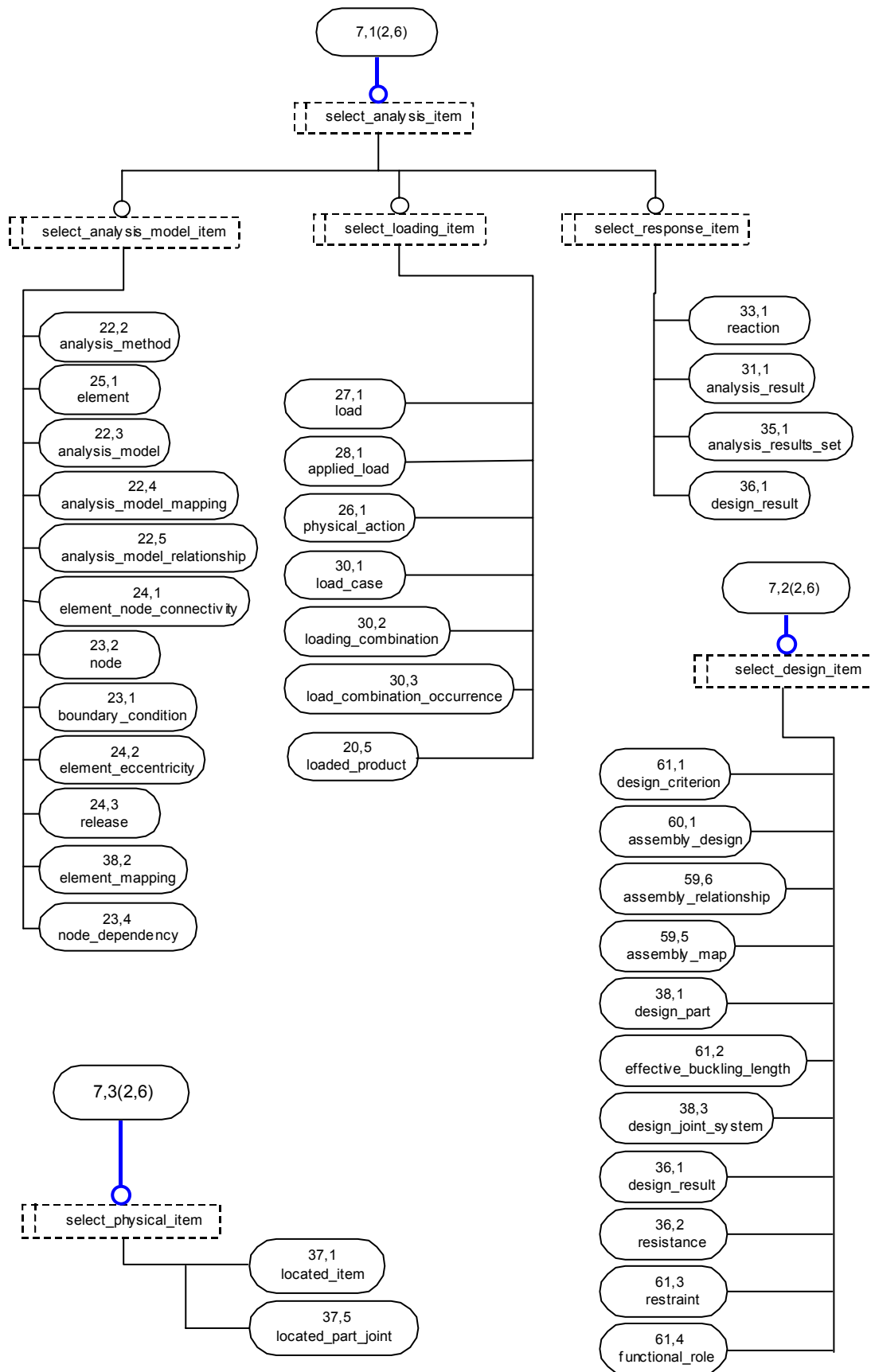
LPM/6 EXPRESS-G Diagram 4 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 5 of 82 (Modified for 2nd Edition)

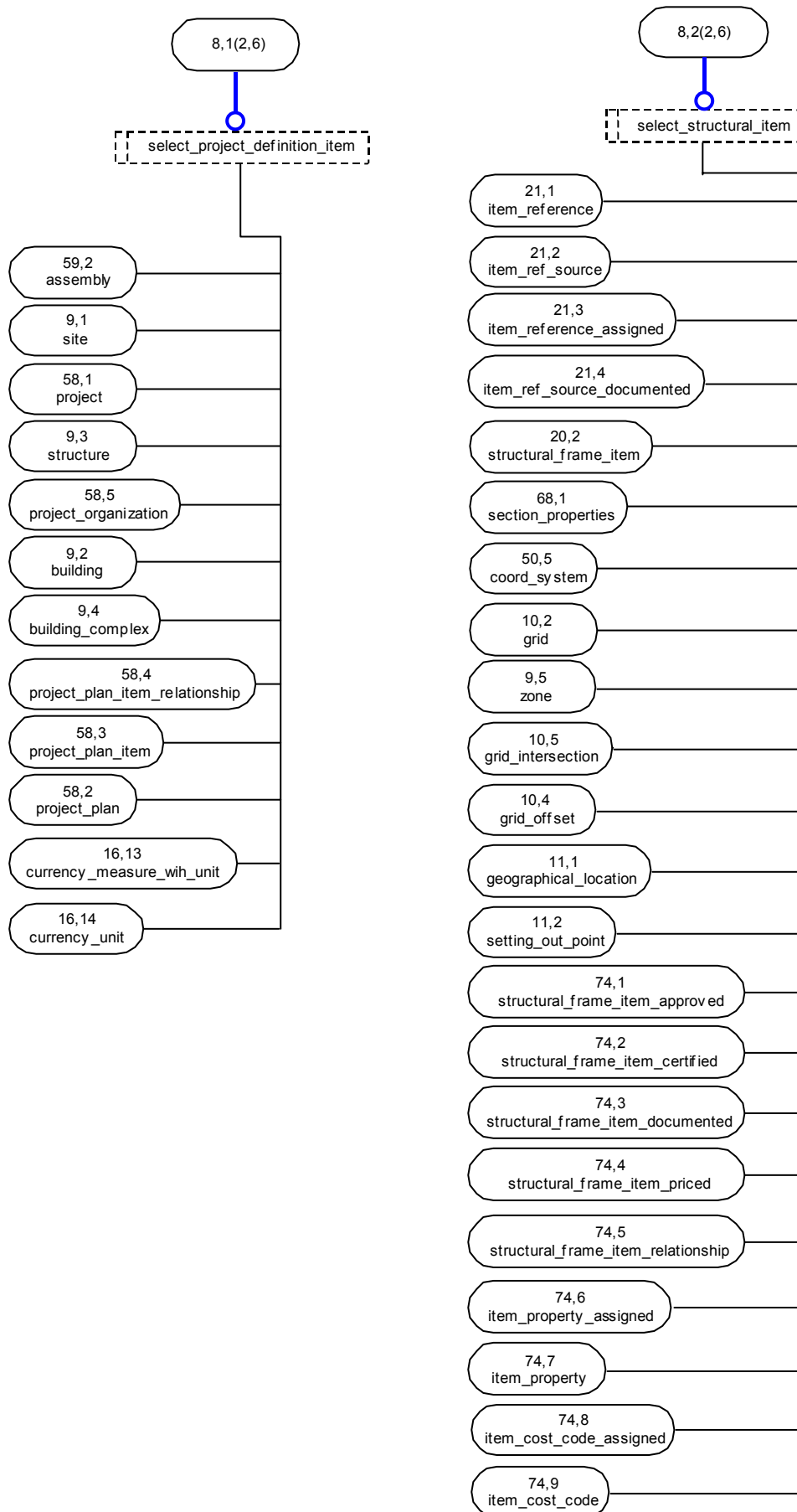
LPM/6 EXPRESS-G Diagram 6 of 82

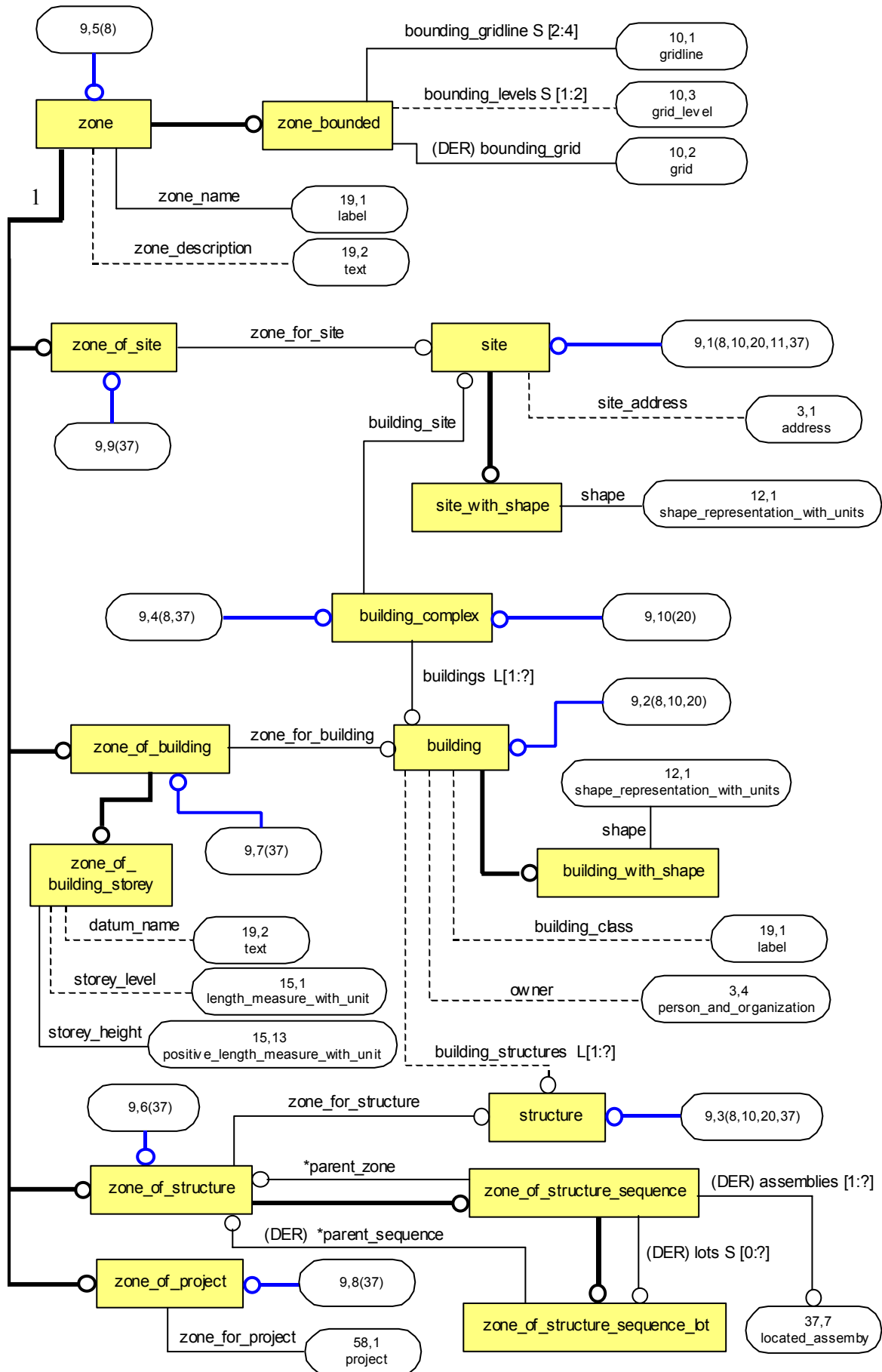


LPM/6 EXPRESS-G Diagram 7 of 82

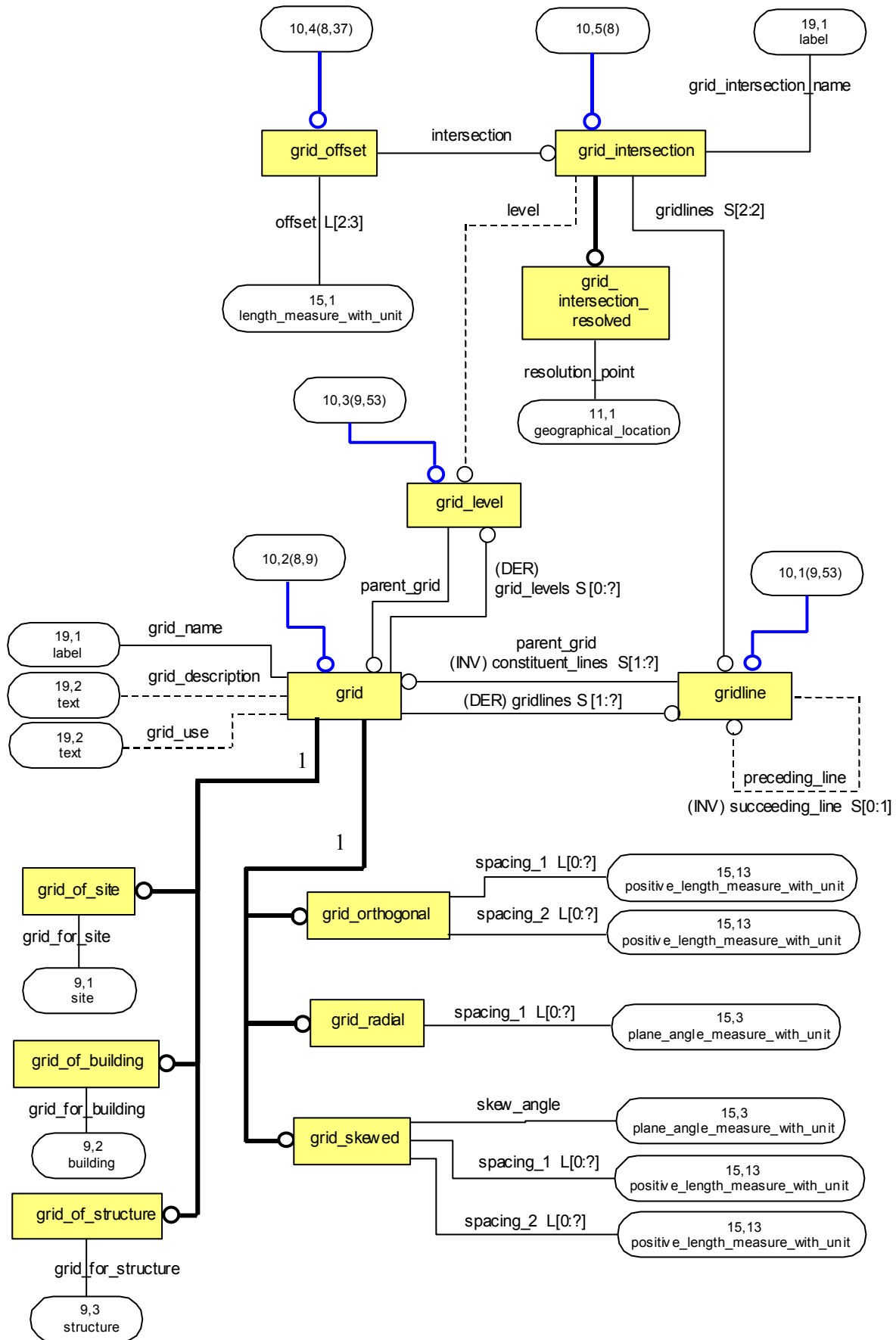


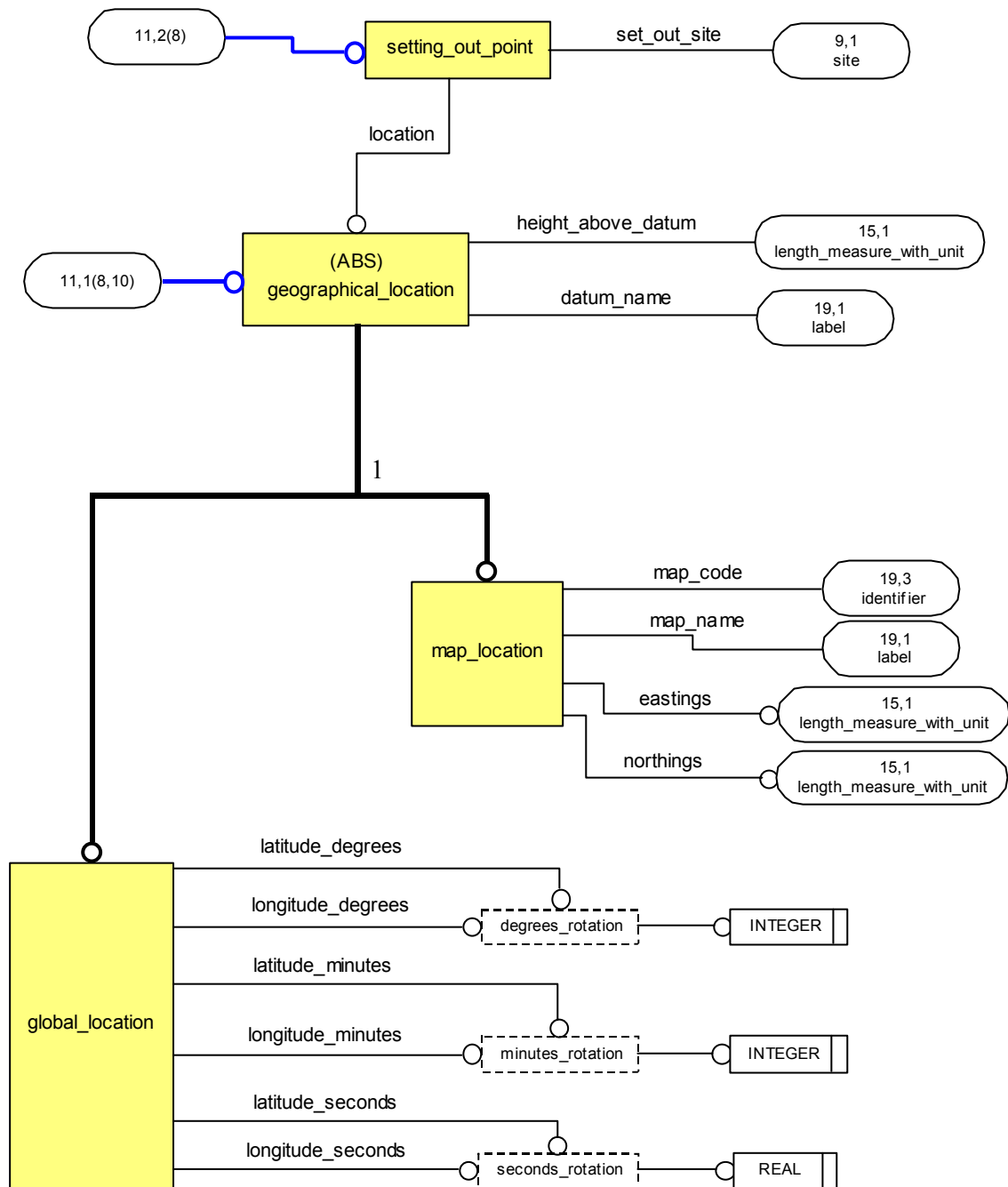
LPM/6 EXPRESS-G Diagram 8 of 82

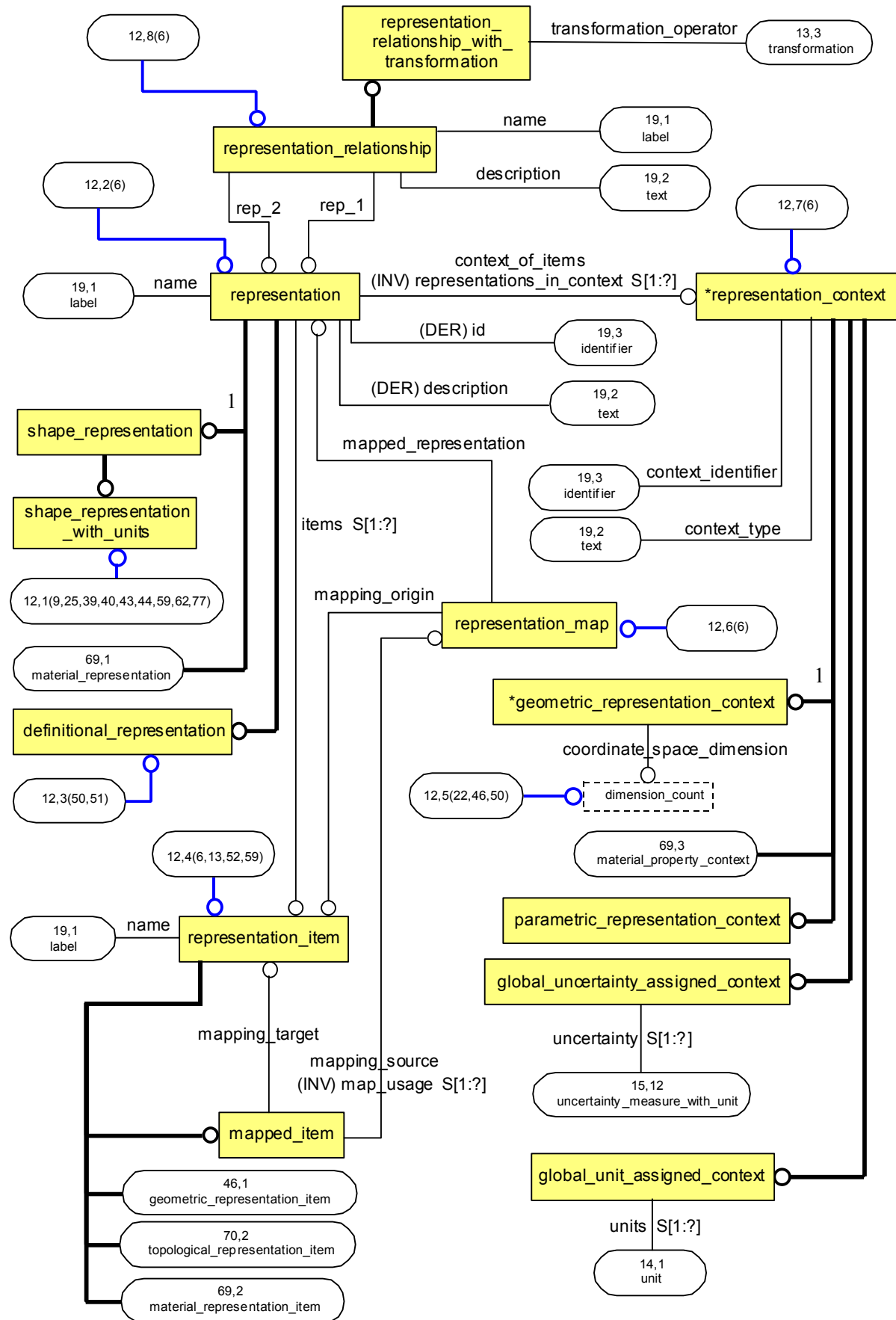


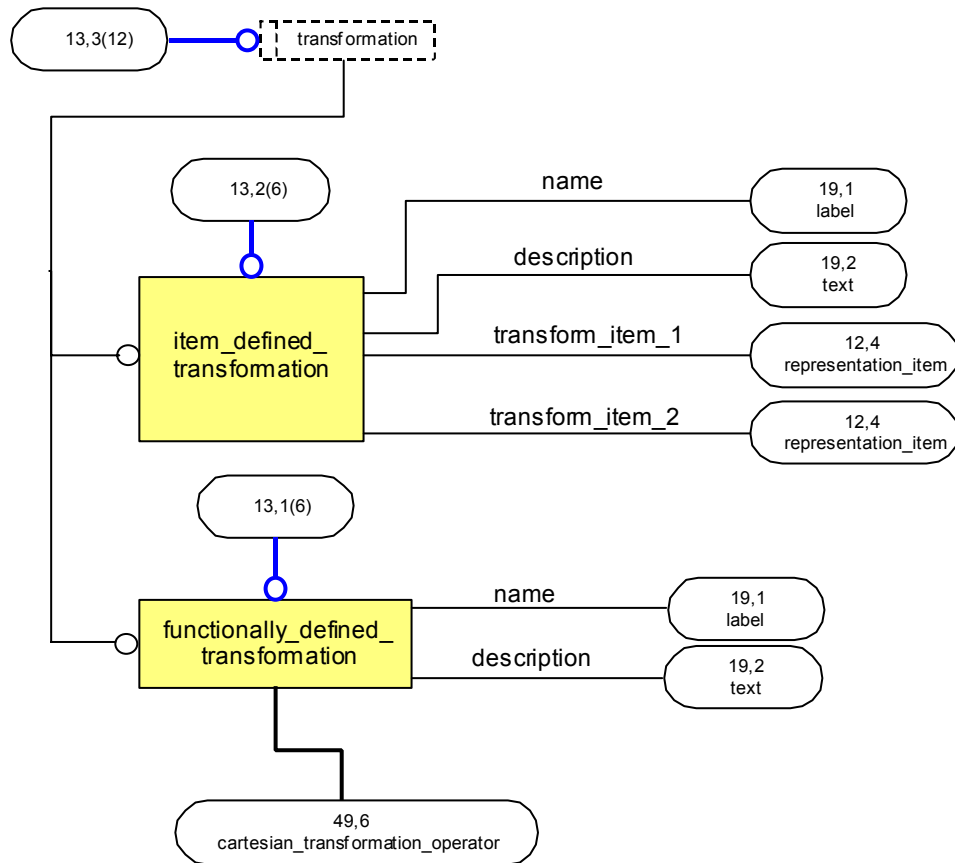
LPM/6 EXPRESS-G Diagram 9 of 82 (Modified for 2nd Edition)

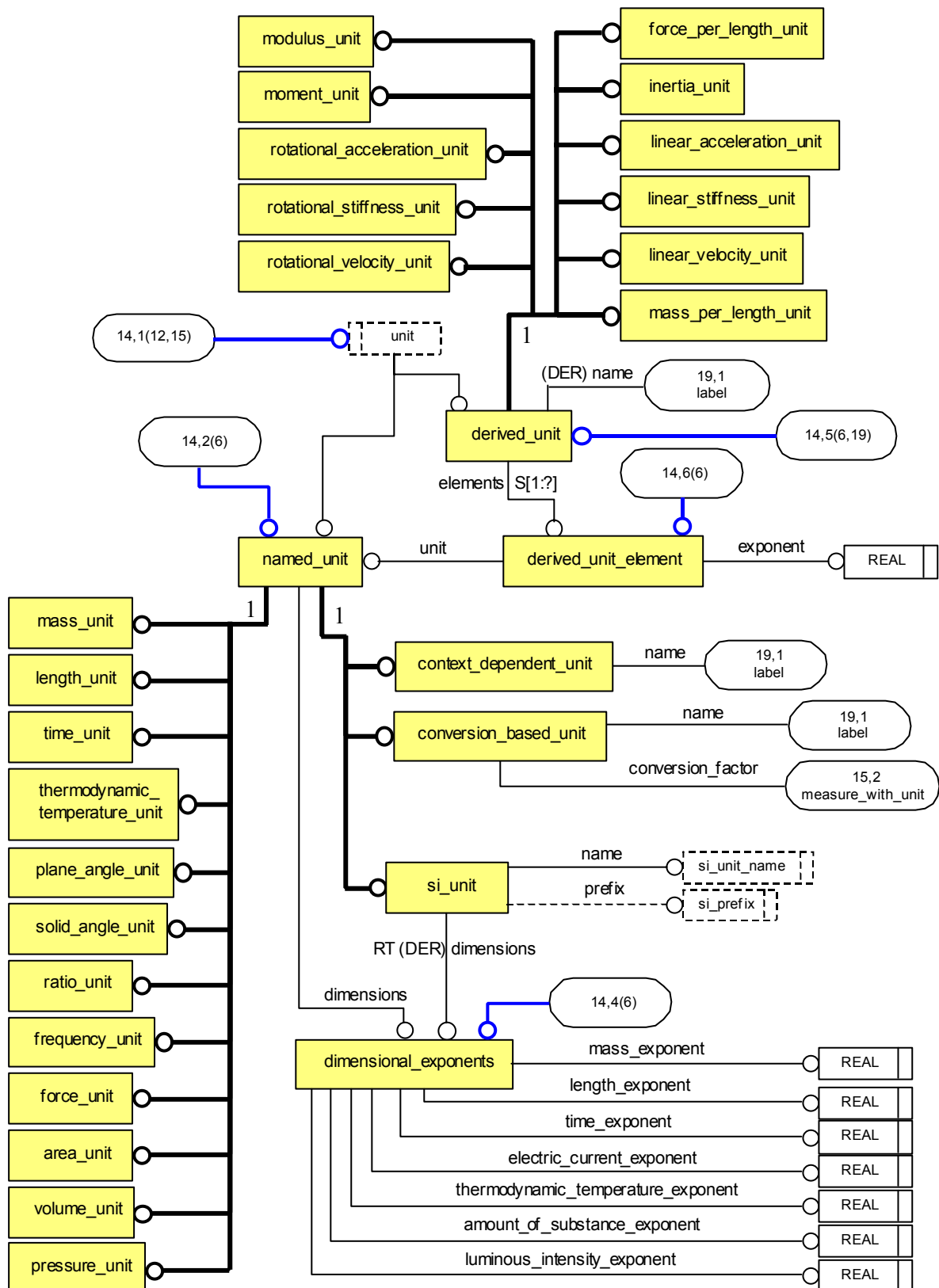
LPM/6 EXPRESS-G Diagram 10 of 82



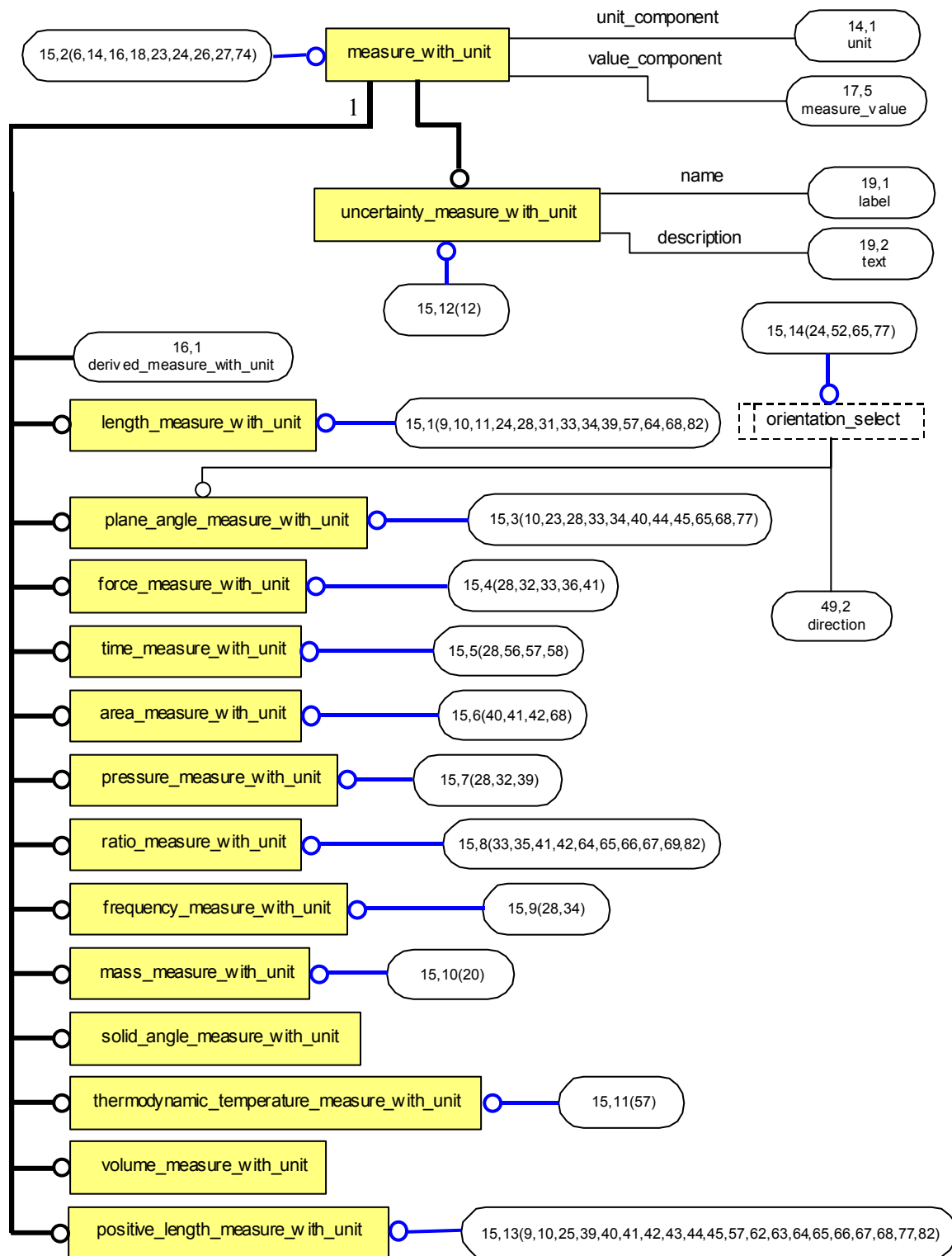
LPM/6 EXPRESS-G Diagram 11 of 82

LPM/6 EXPRESS-G Diagram 12 of 82 (Modified for 2nd Edition)

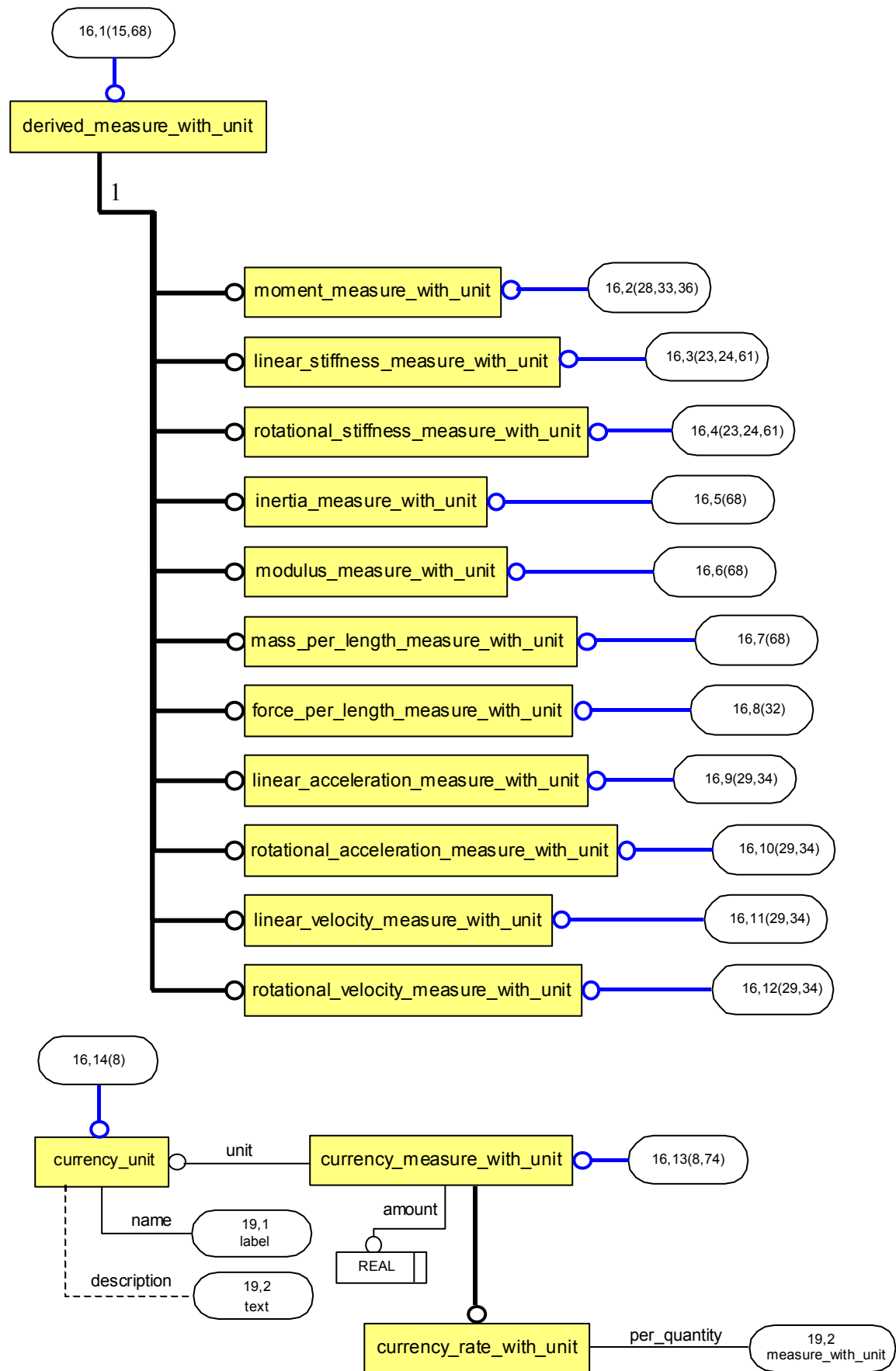
LPM/6 EXPRESS-G Diagram 13 of 82

LPM/6 EXPRESS-G Diagram 14 of 82 (Modified for 2nd Edition)

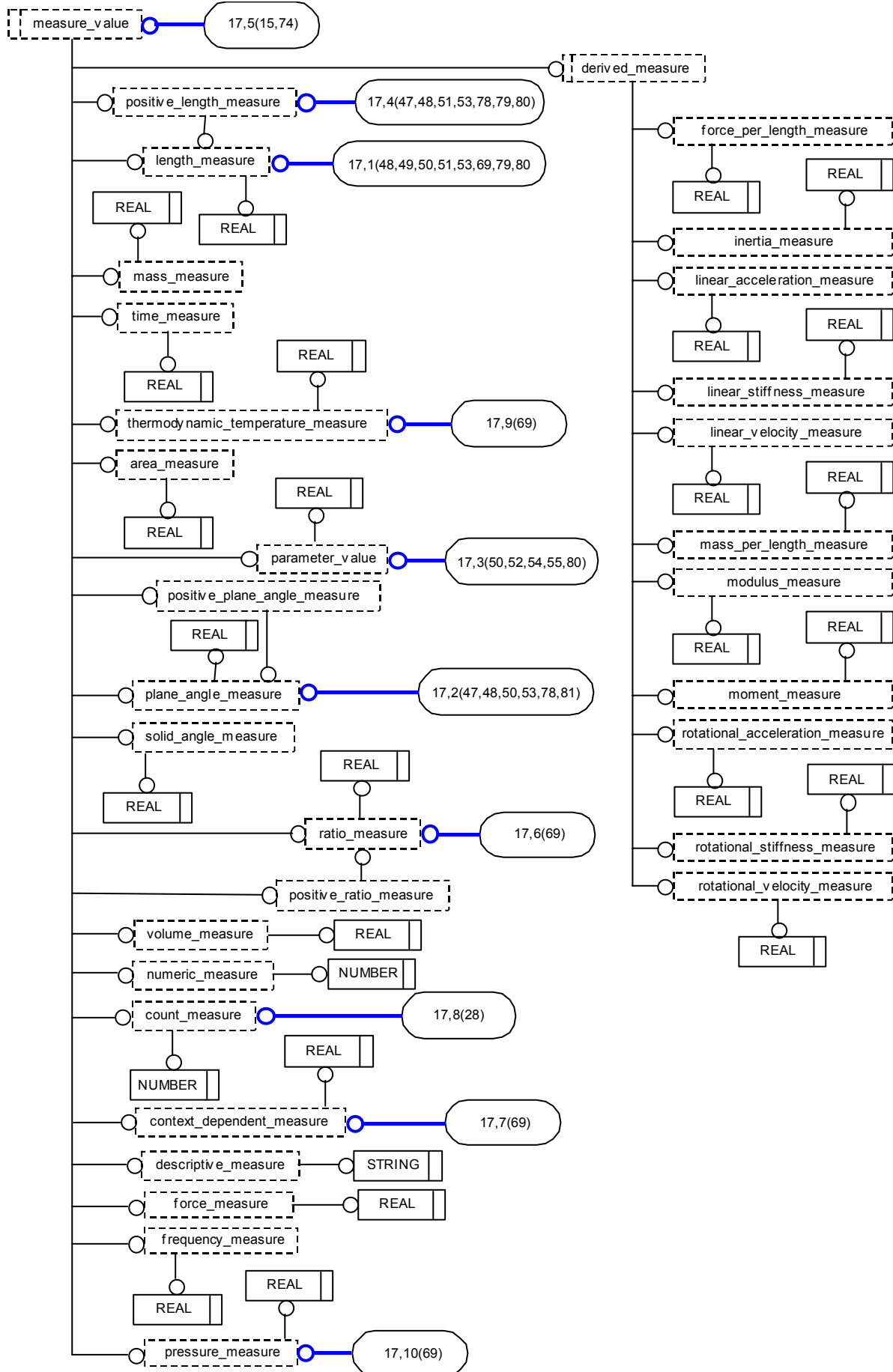
LPM/6 EXPRESS-G Diagram 15 of 82



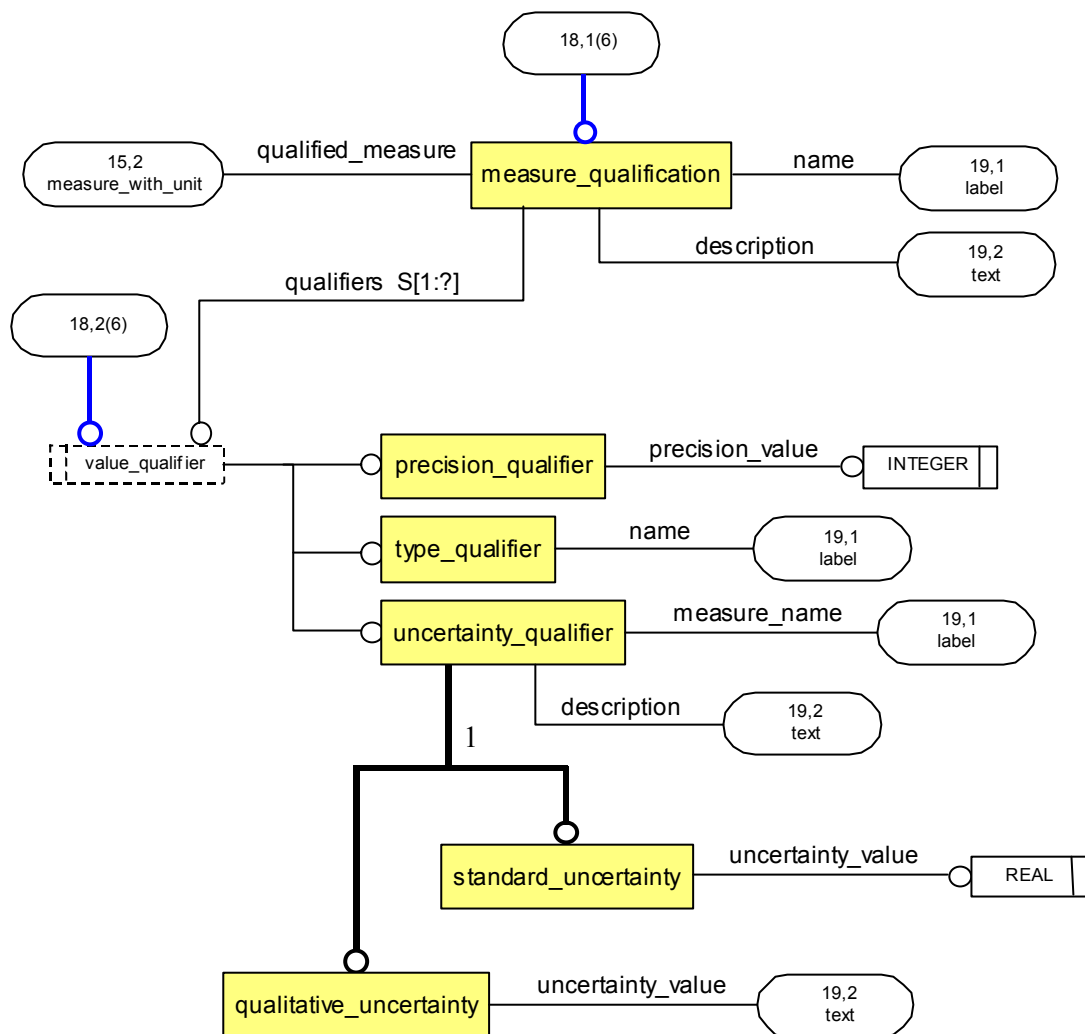
LPM/6 EXPRESS-G Diagram 16 of 82

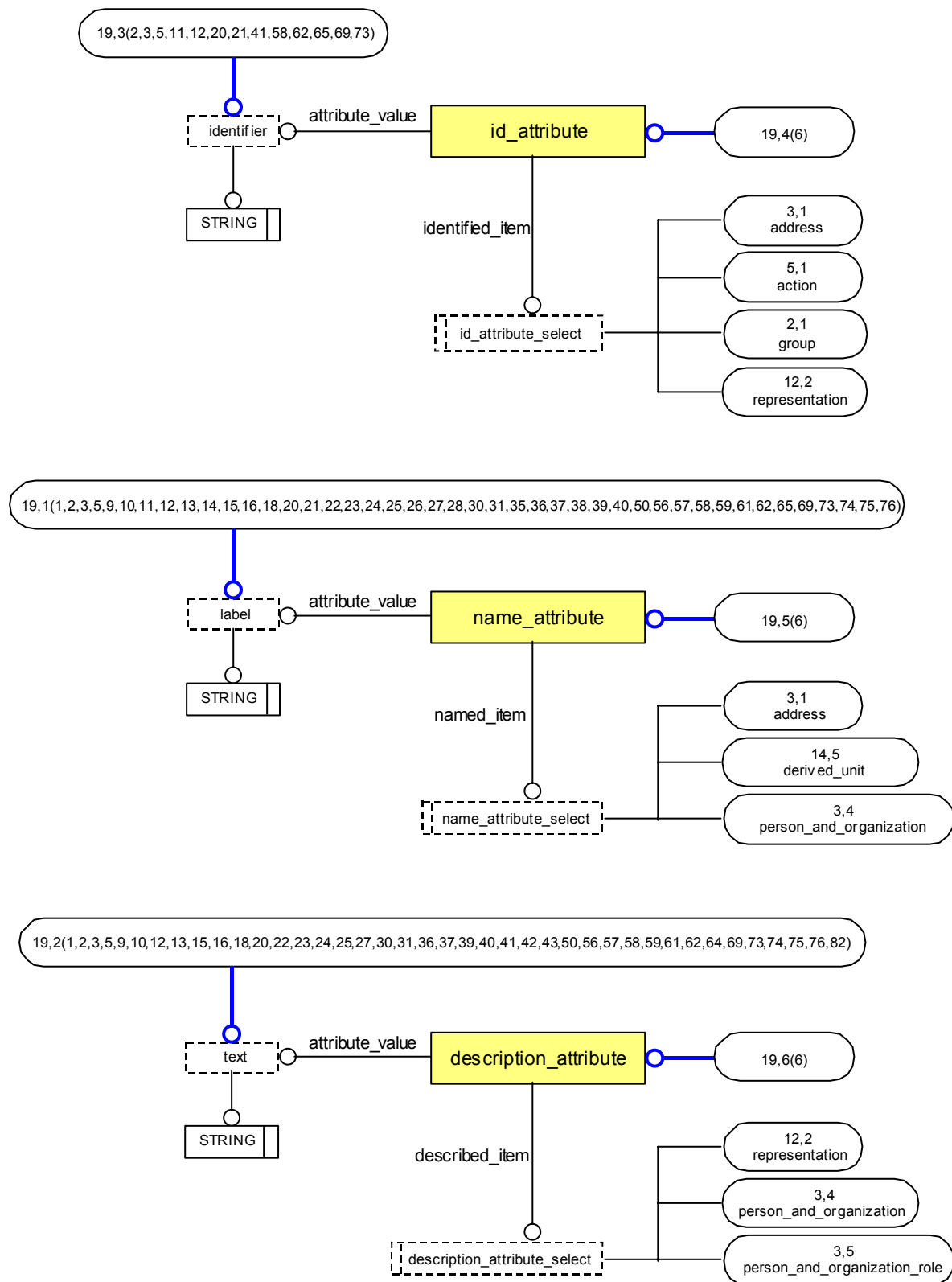


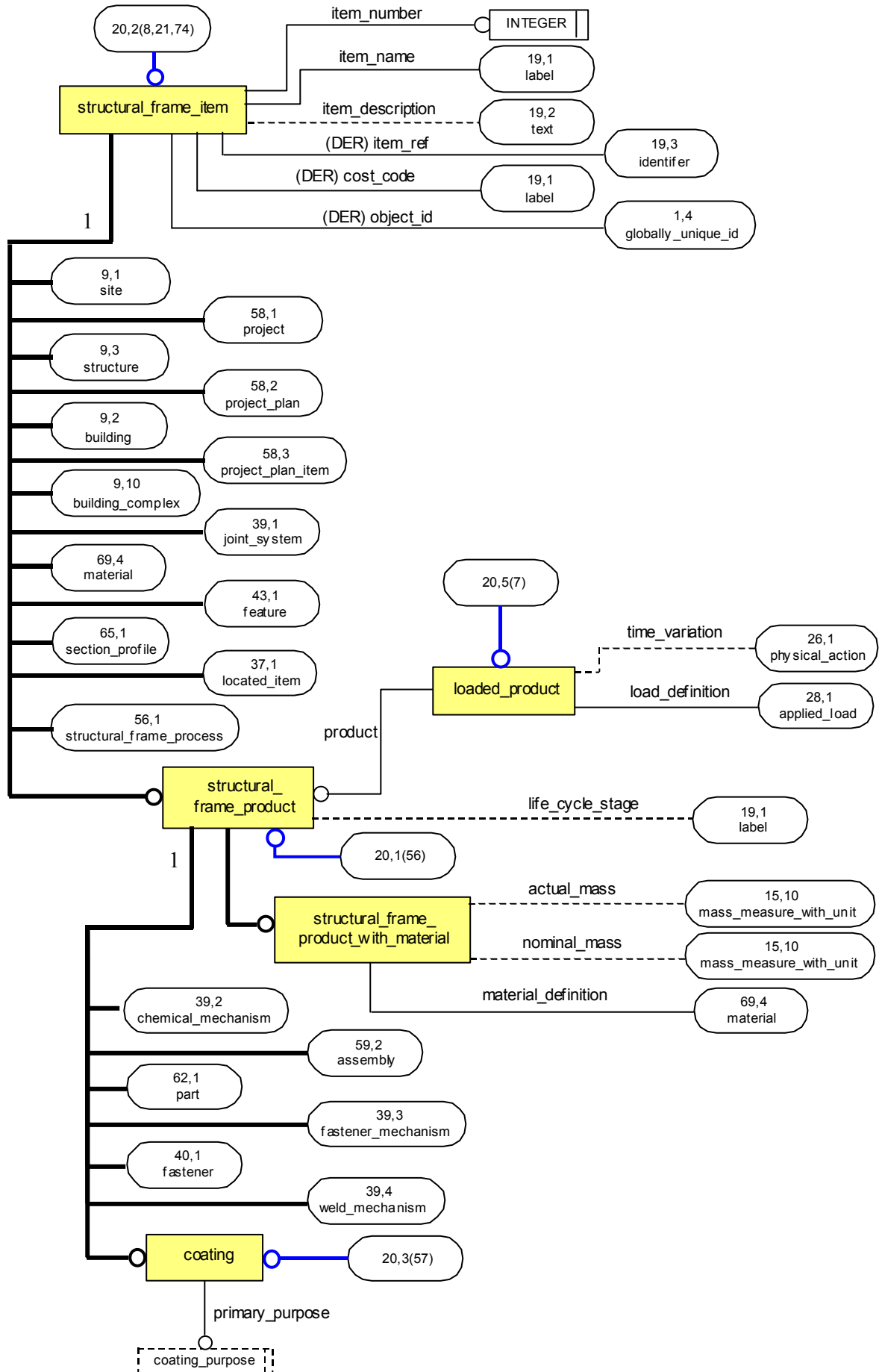
LPM/6 EXPRESS-G Diagram 17 of 82

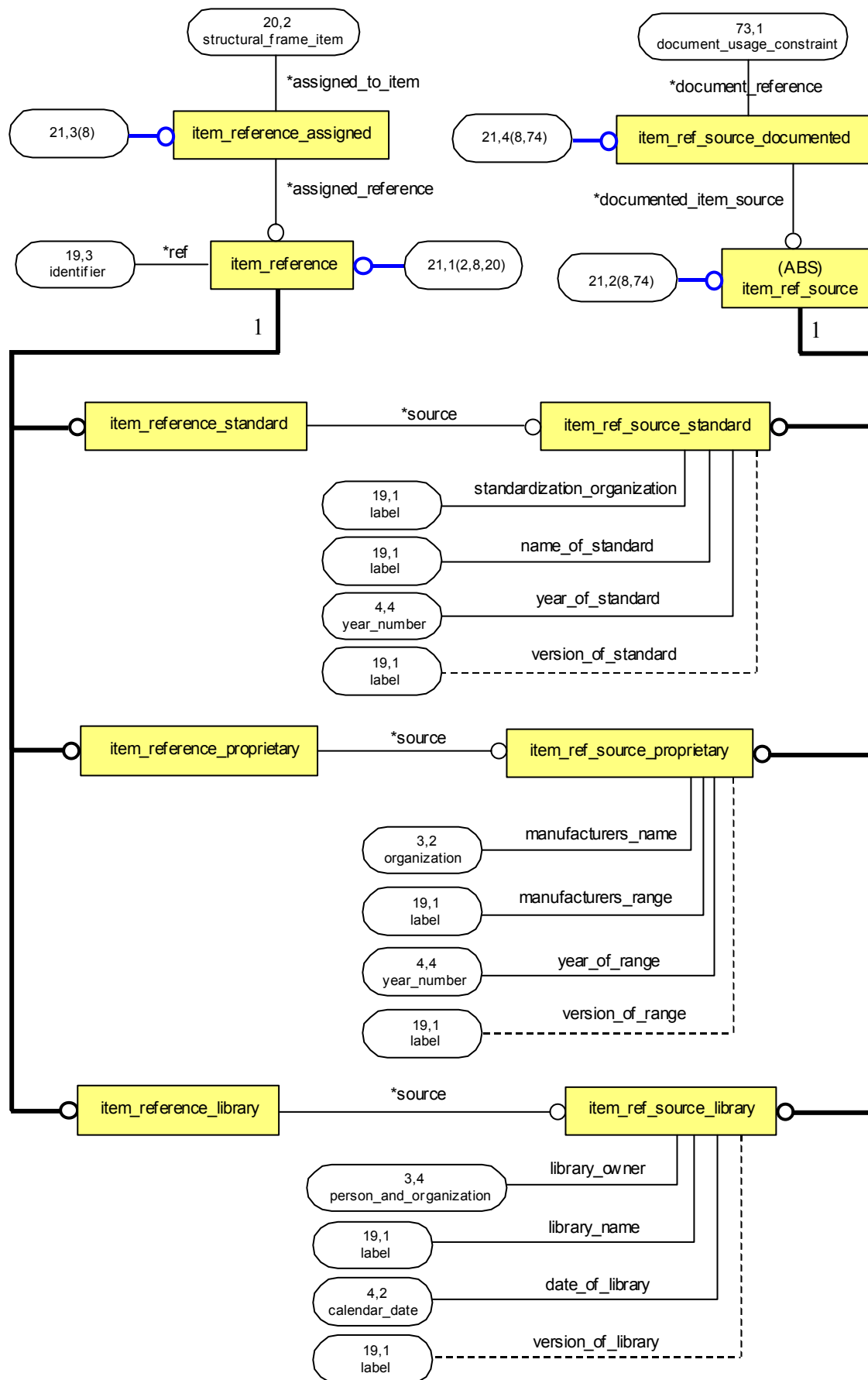


LPM/6 EXPRESS-G Diagram 18 of 82

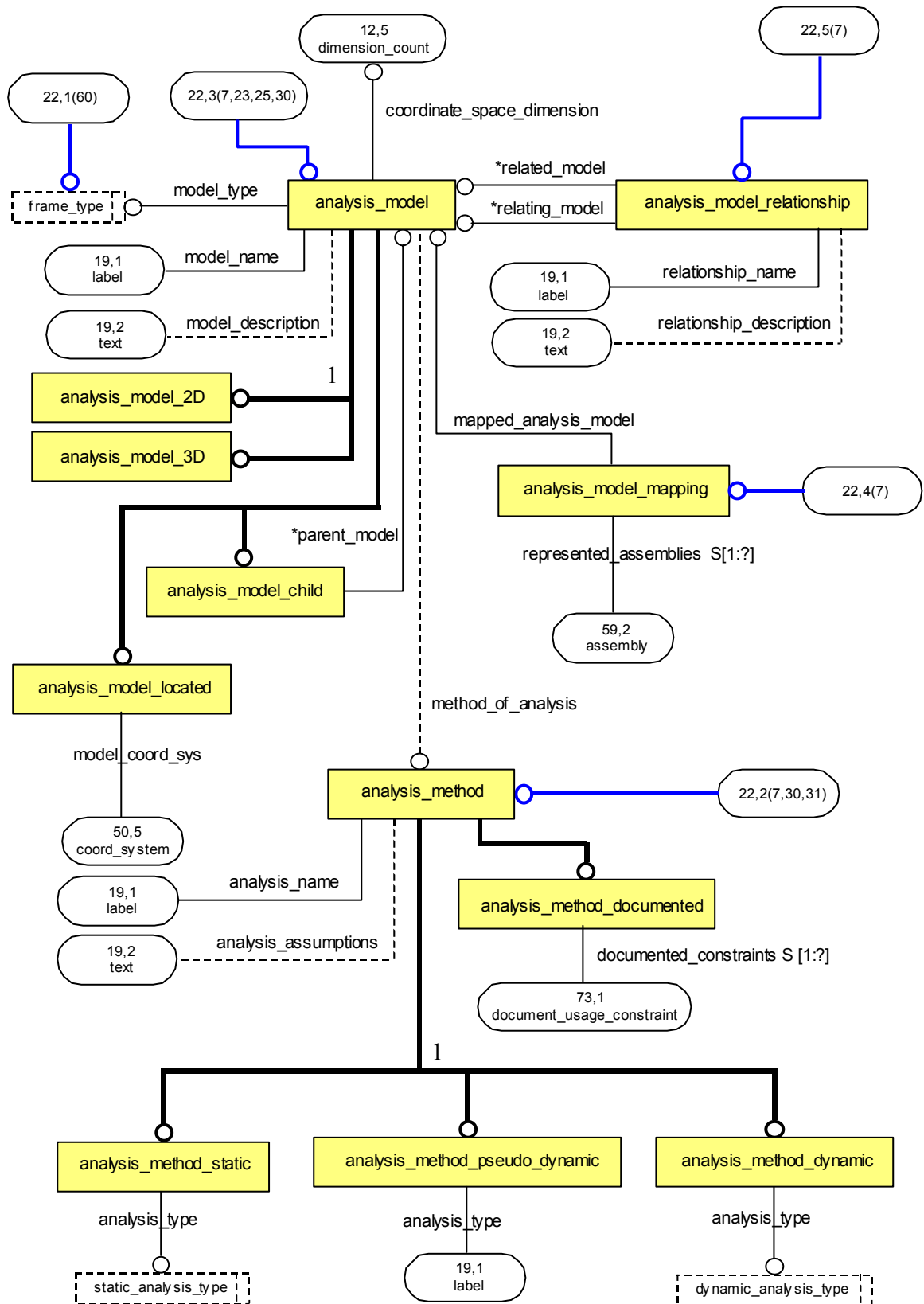


LPM/6 EXPRESS-G Diagram 19 of 82 (Modified for 2nd Edition)

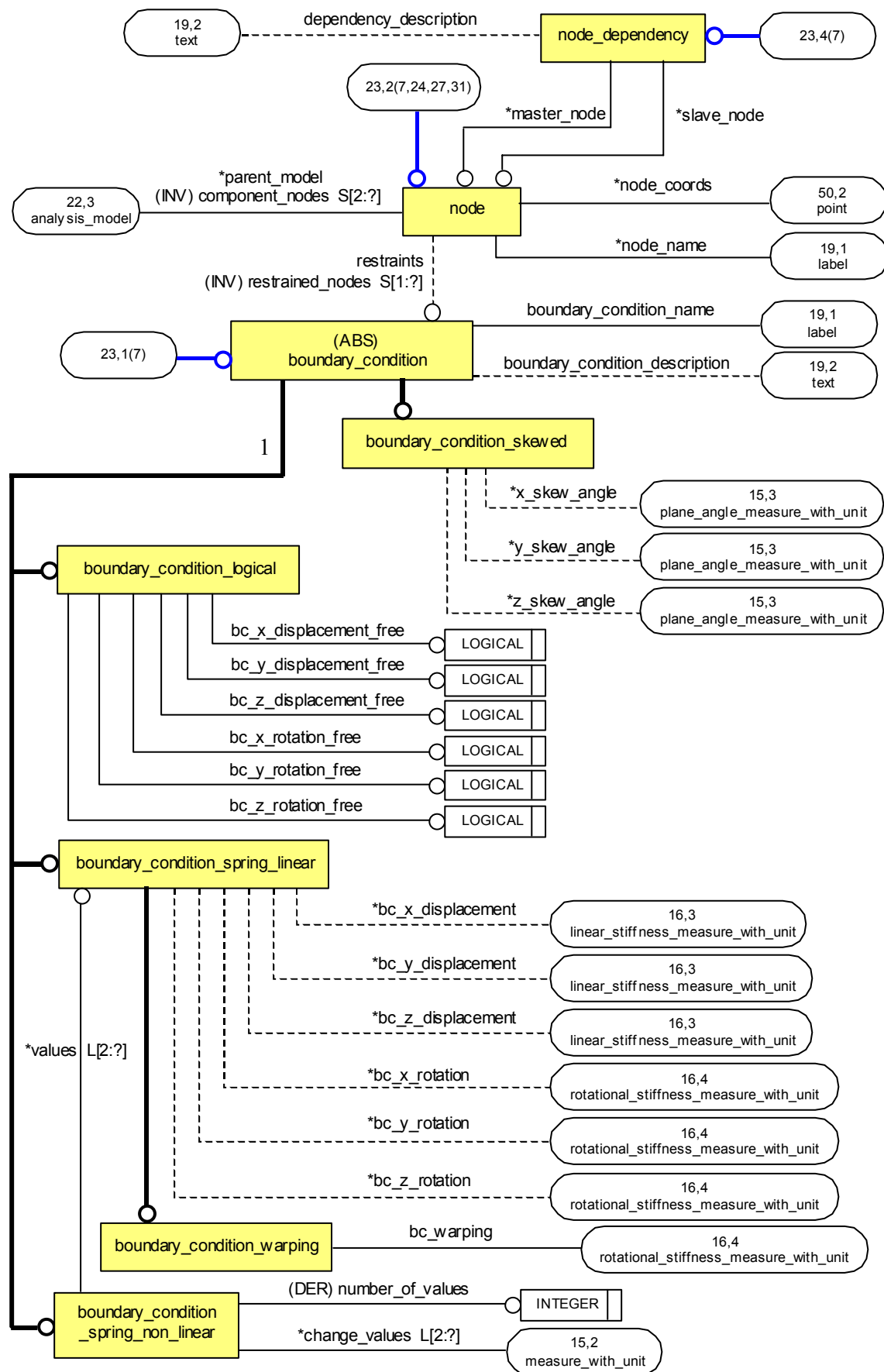
LPM/6 EXPRESS-G Diagram 20 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 21 of 82 (modified for 2nd edition)

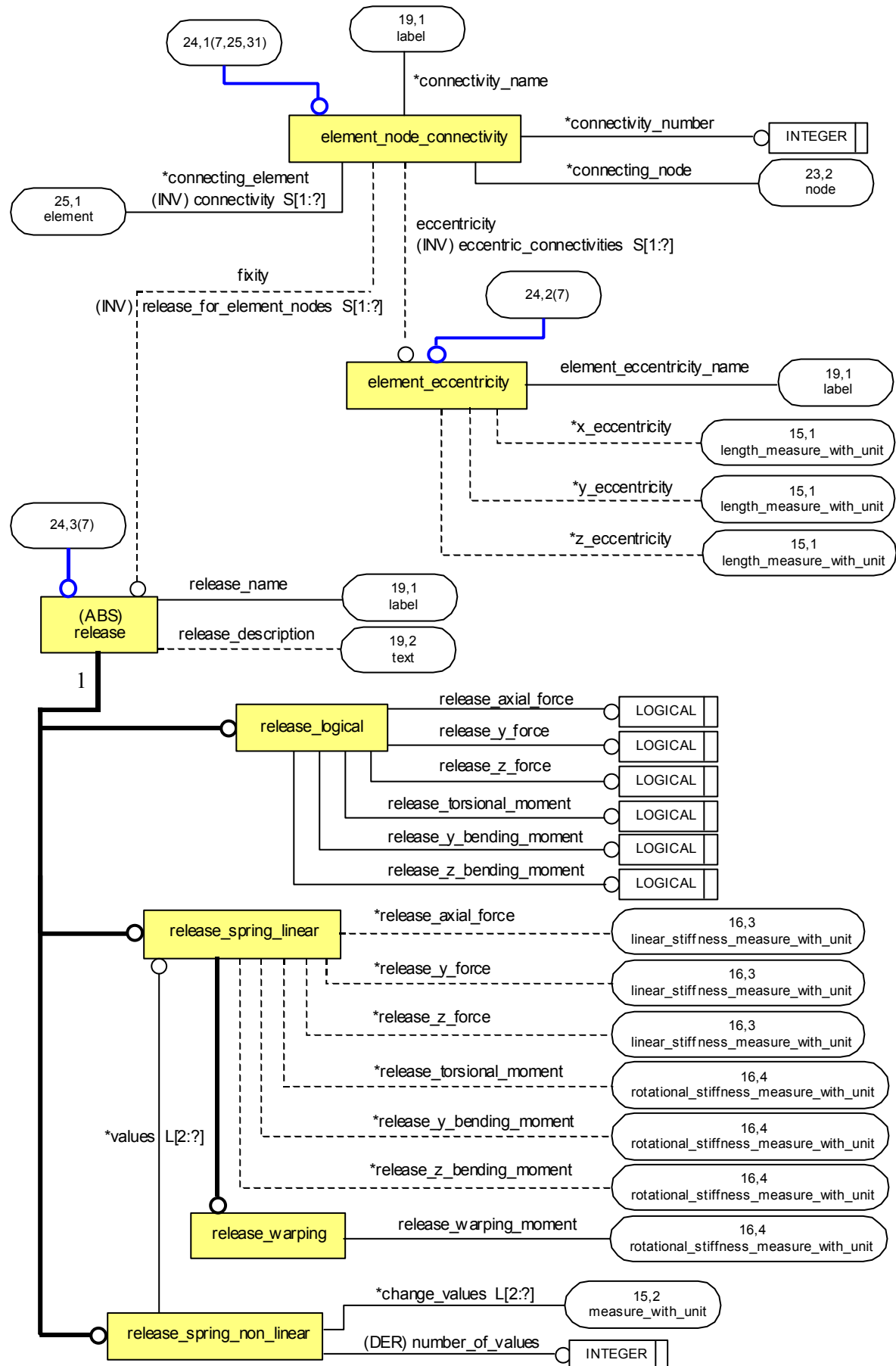
LPM/6 EXPRESS-G Diagram 22 of 82



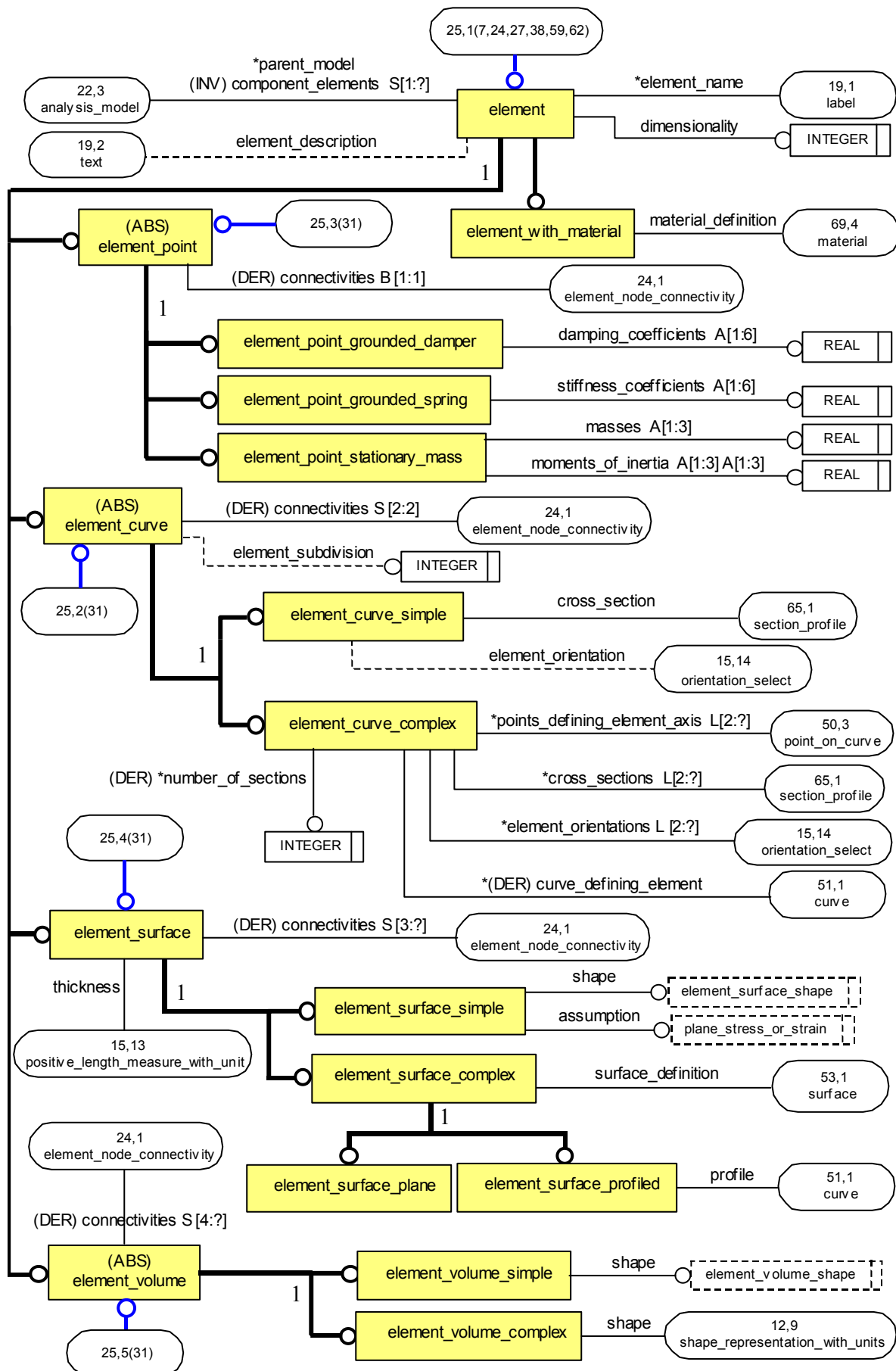
LPM/6 EXPRESS-G Diagram 23 of 82



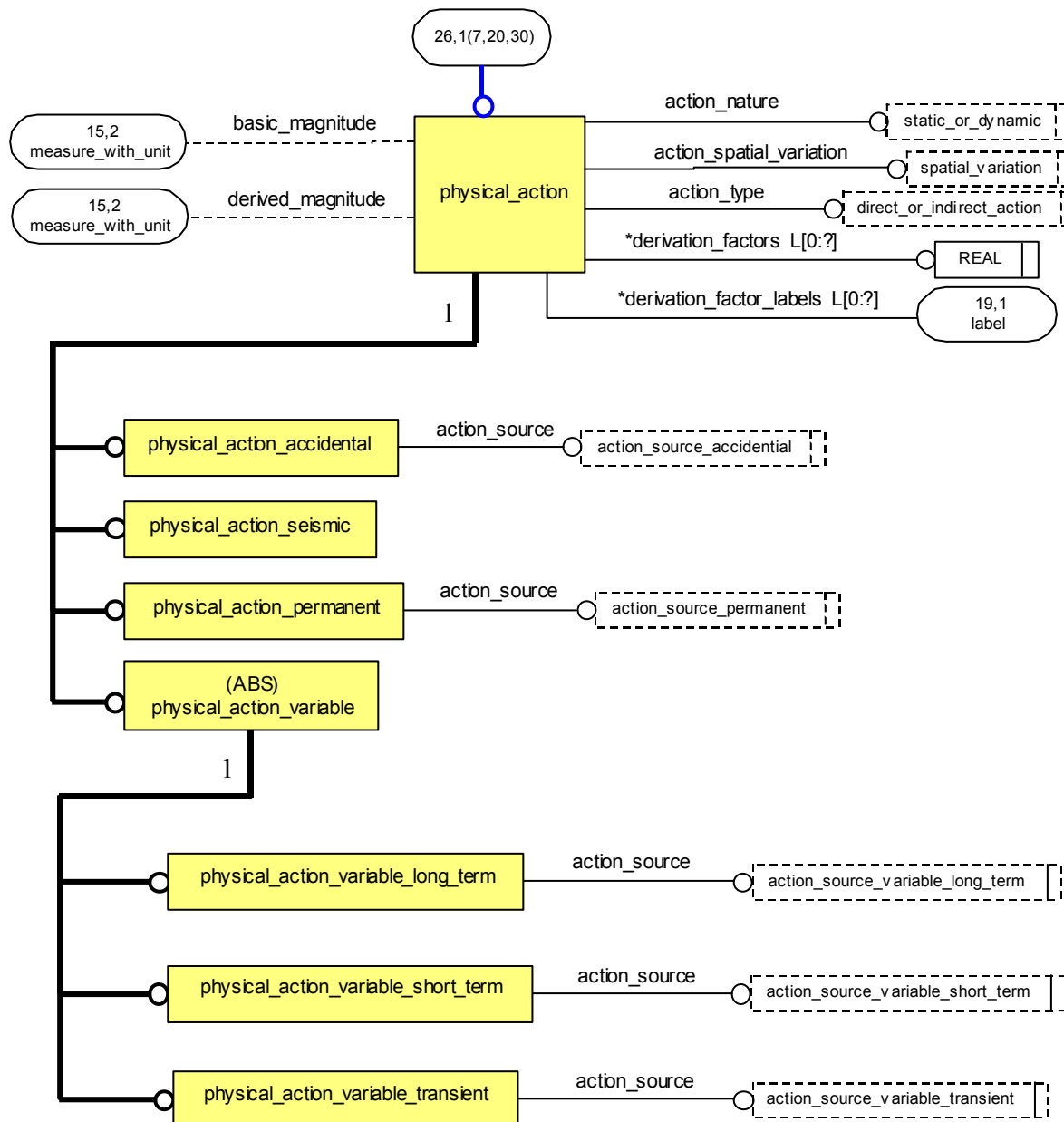
LPM/6 EXPRESS-G Diagram 24 of 82

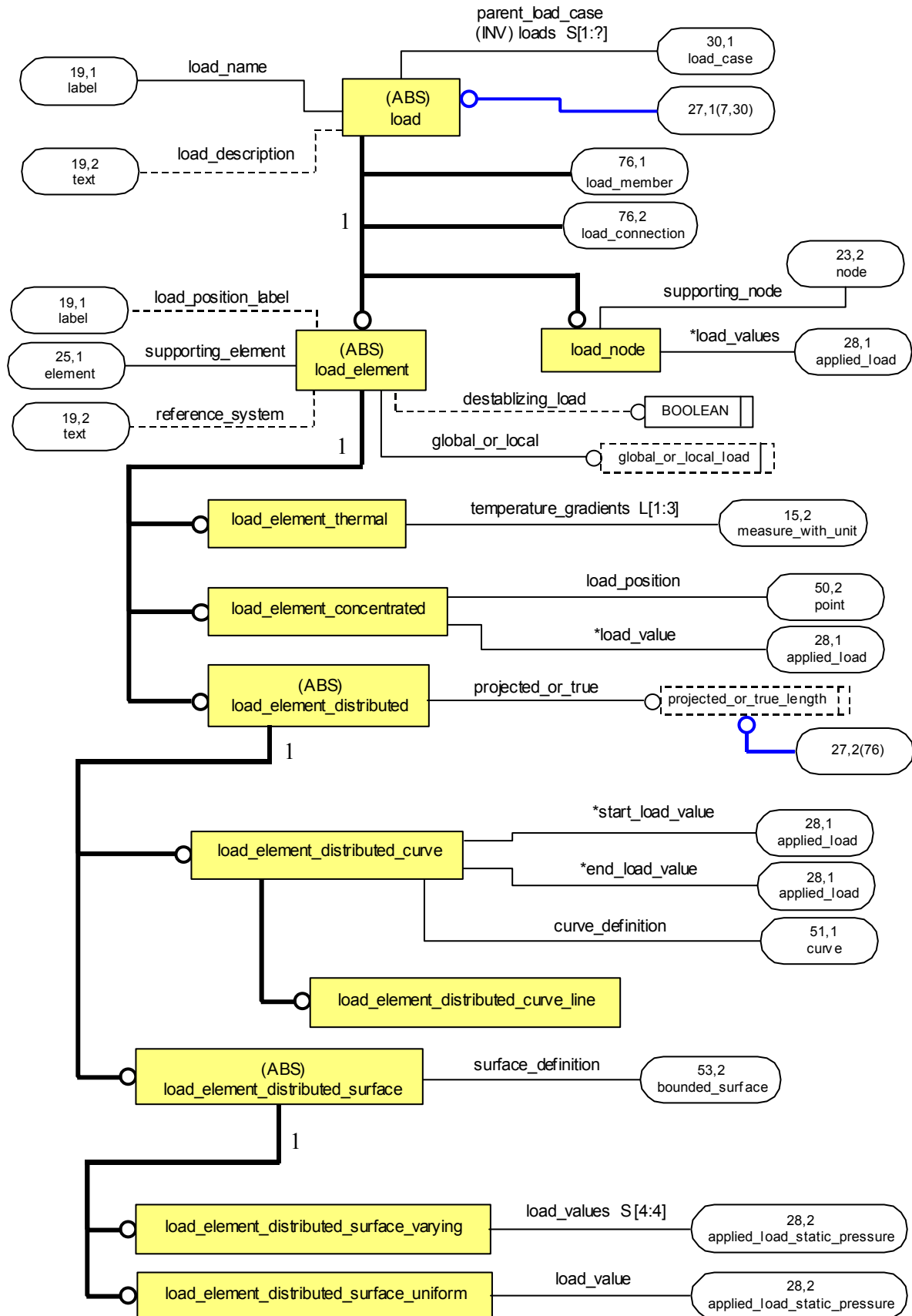


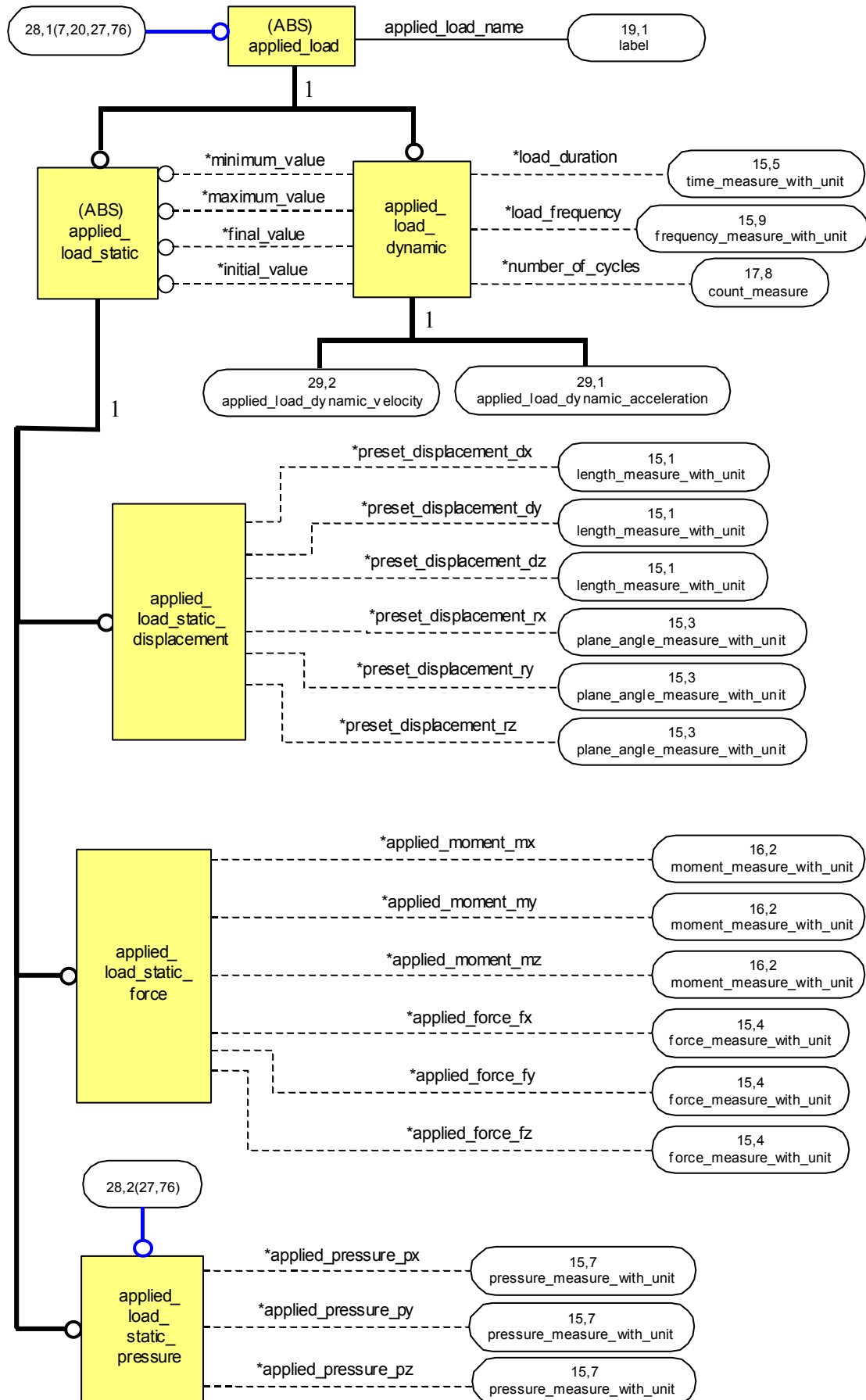
LPM/6 EXPRESS-G Diagram 25 of 82



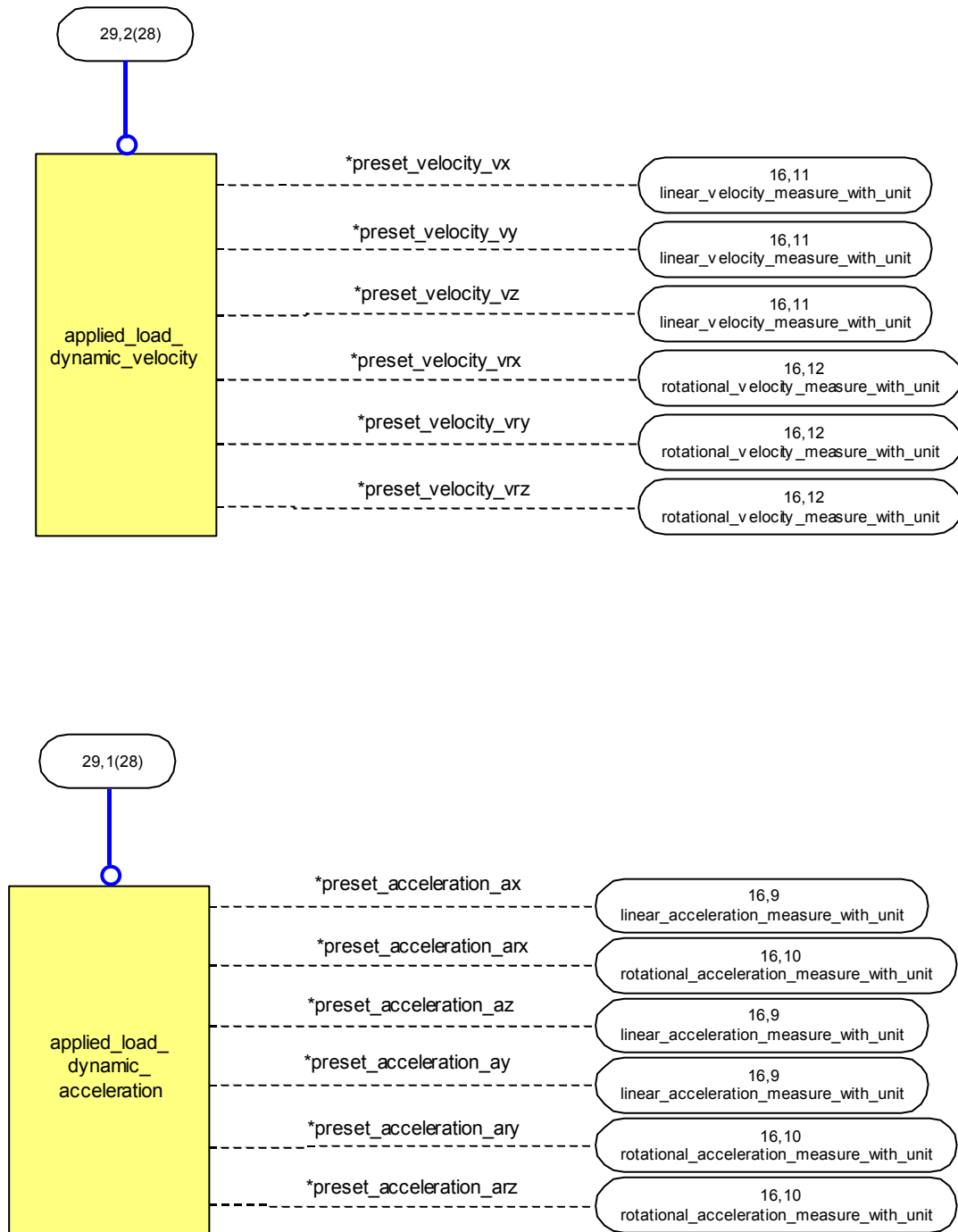
LPM/6 EXPRESS-G Diagram 26 of 82

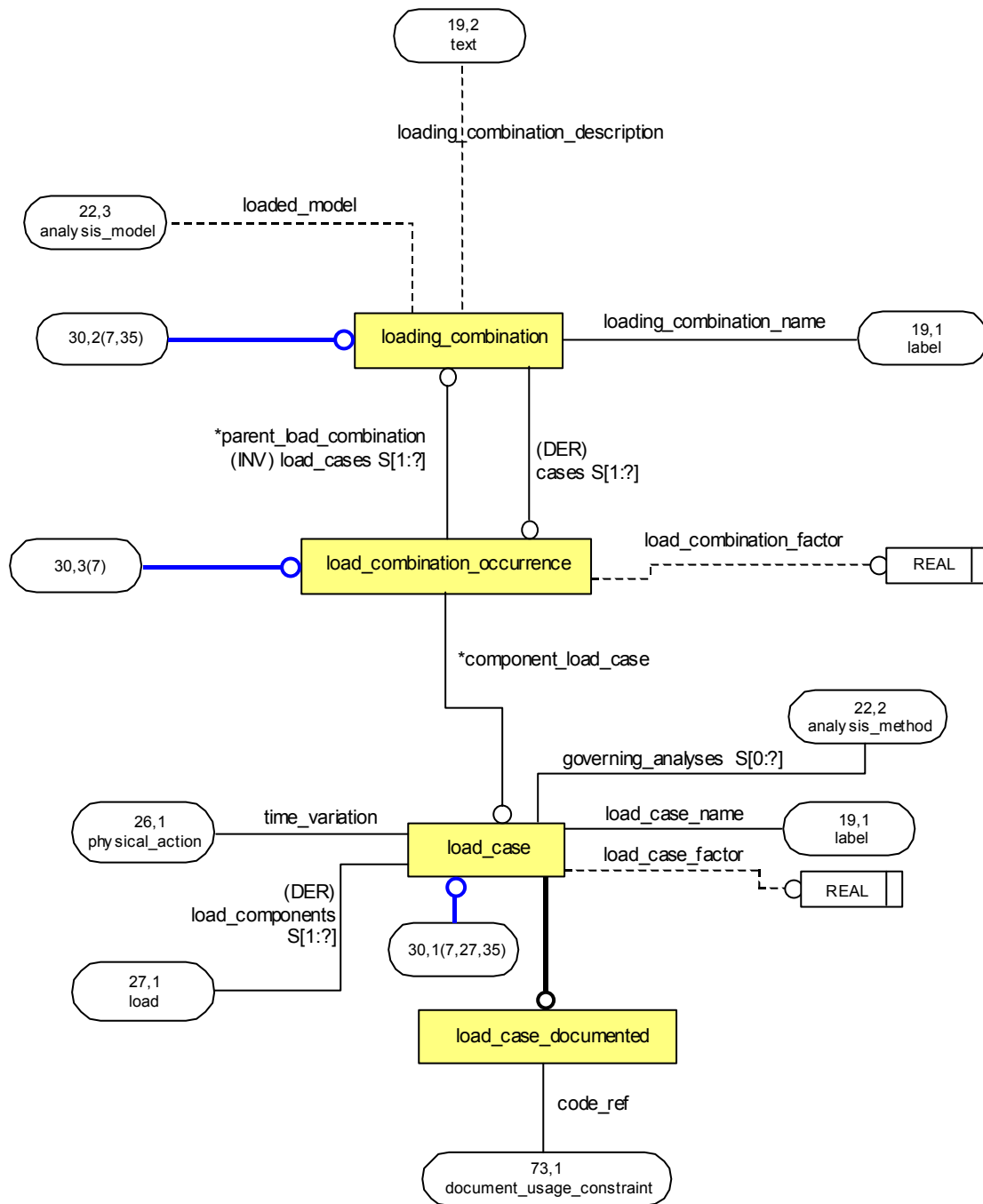


LPM/6 EXPRESS-G Diagram 27 of 82 (Modified for 2nd Edition)

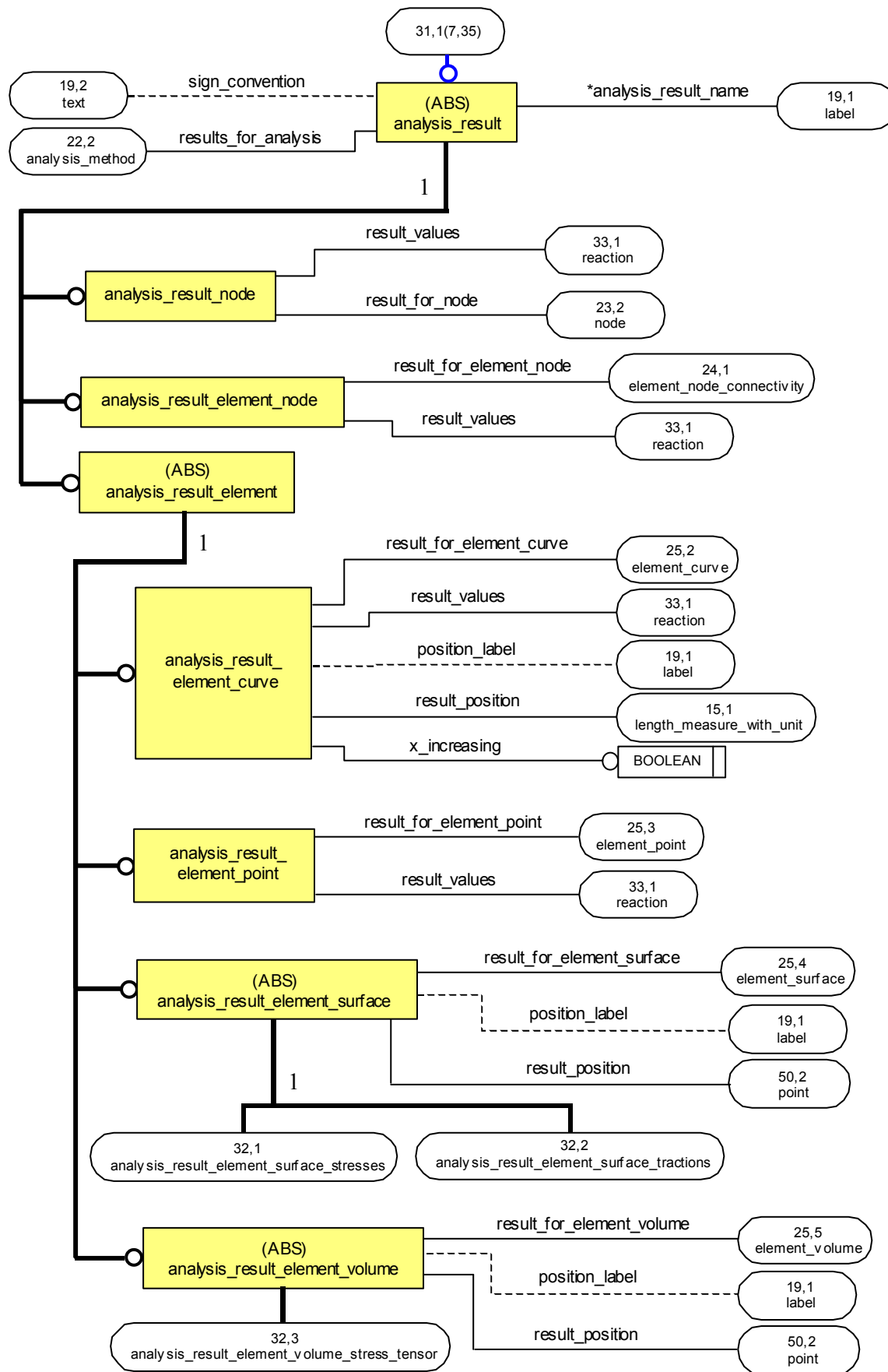
LPM/6 EXPRESS-G Diagram 28 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 29 of 82

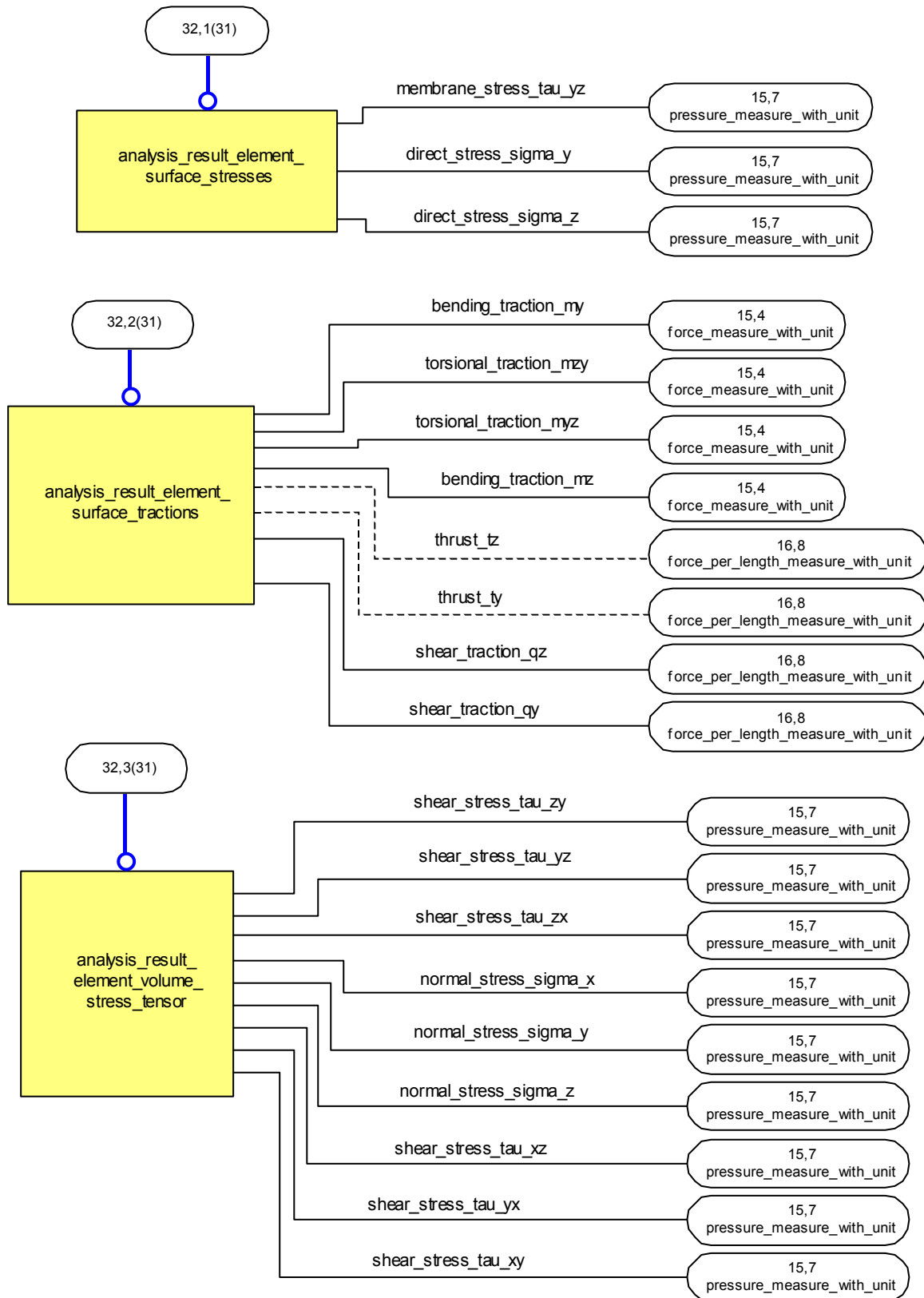


LPM/6 EXPRESS-G Diagram 30 of 82 (Modified for 2nd Edition)

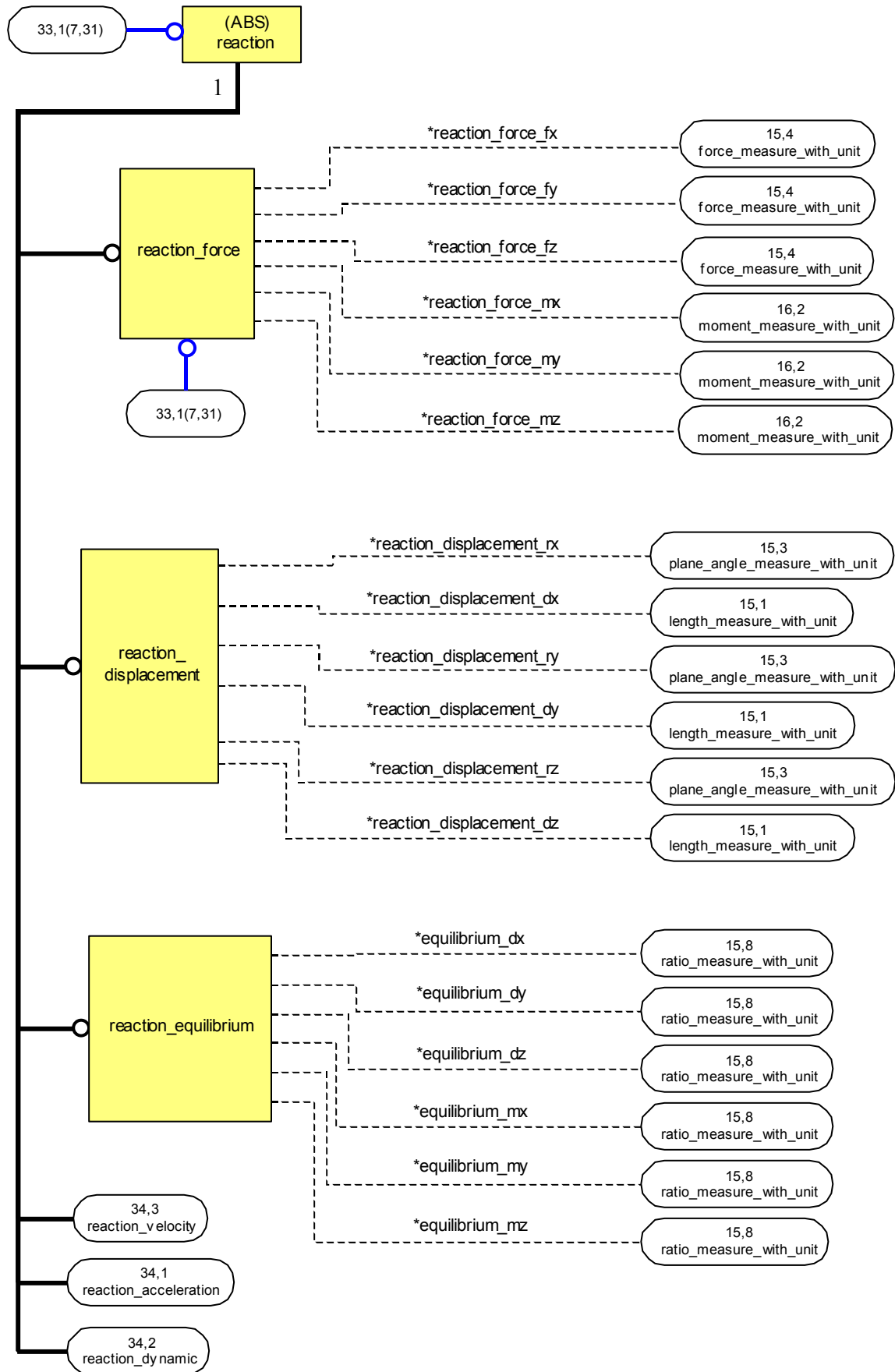
LPM/6 EXPRESS-G Diagram 31 of 82



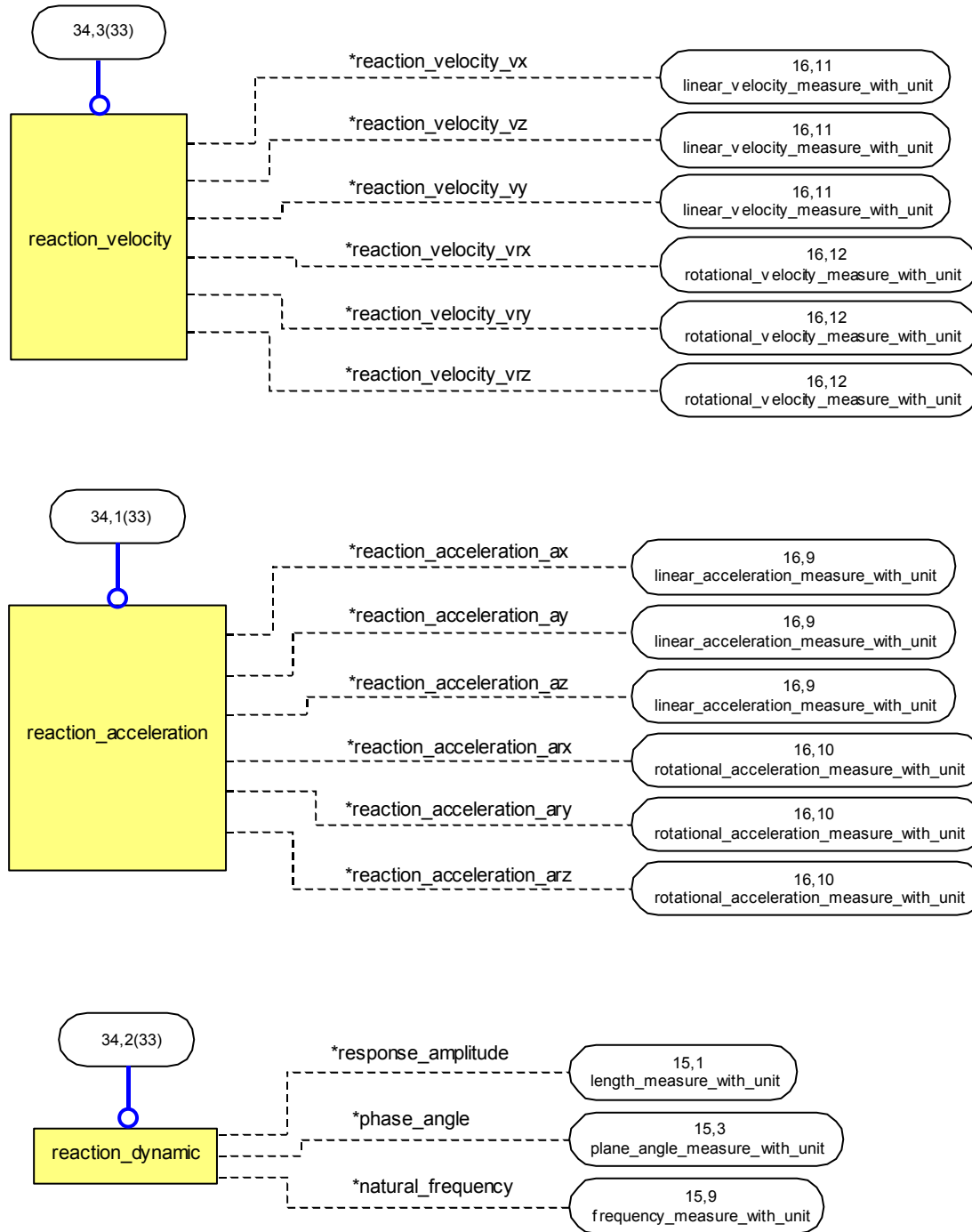
LPM/6 EXPRESS-G Diagram 32 of 82



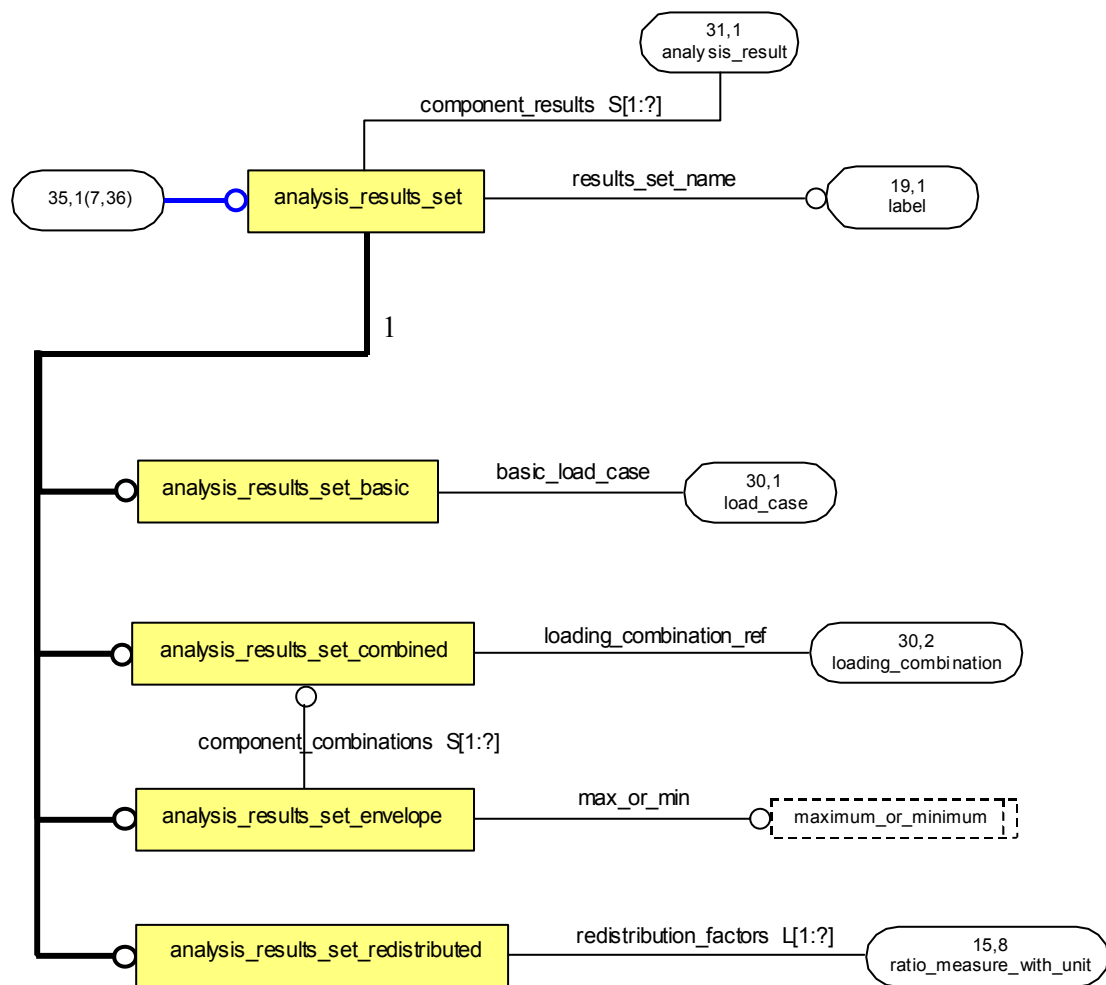
LPM/6 EXPRESS-G Diagram 33 of 82



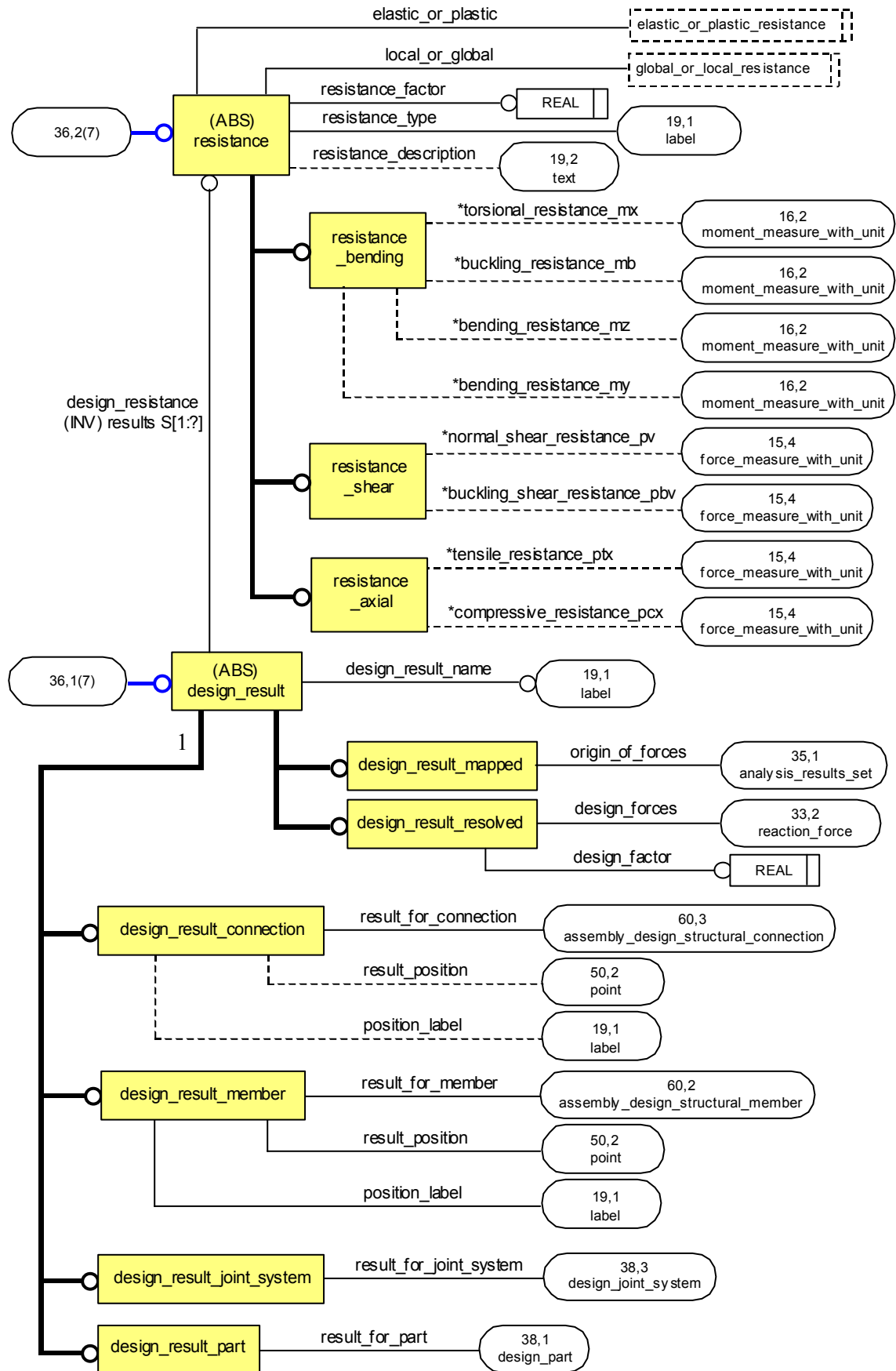
LPM/6 EXPRESS-G Diagram 34 of 82

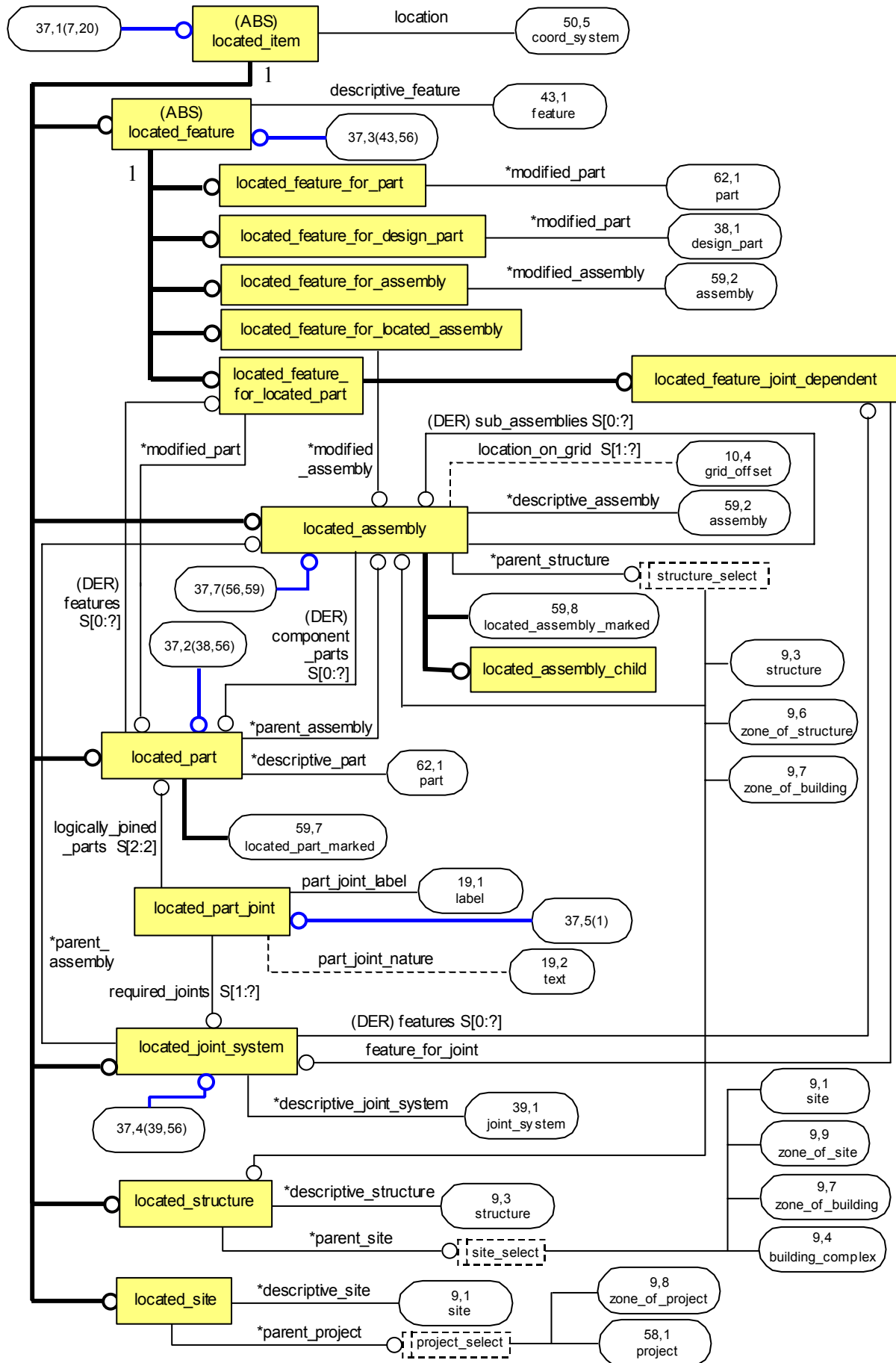


LPM/6 EXPRESS-G Diagram 35 of 82

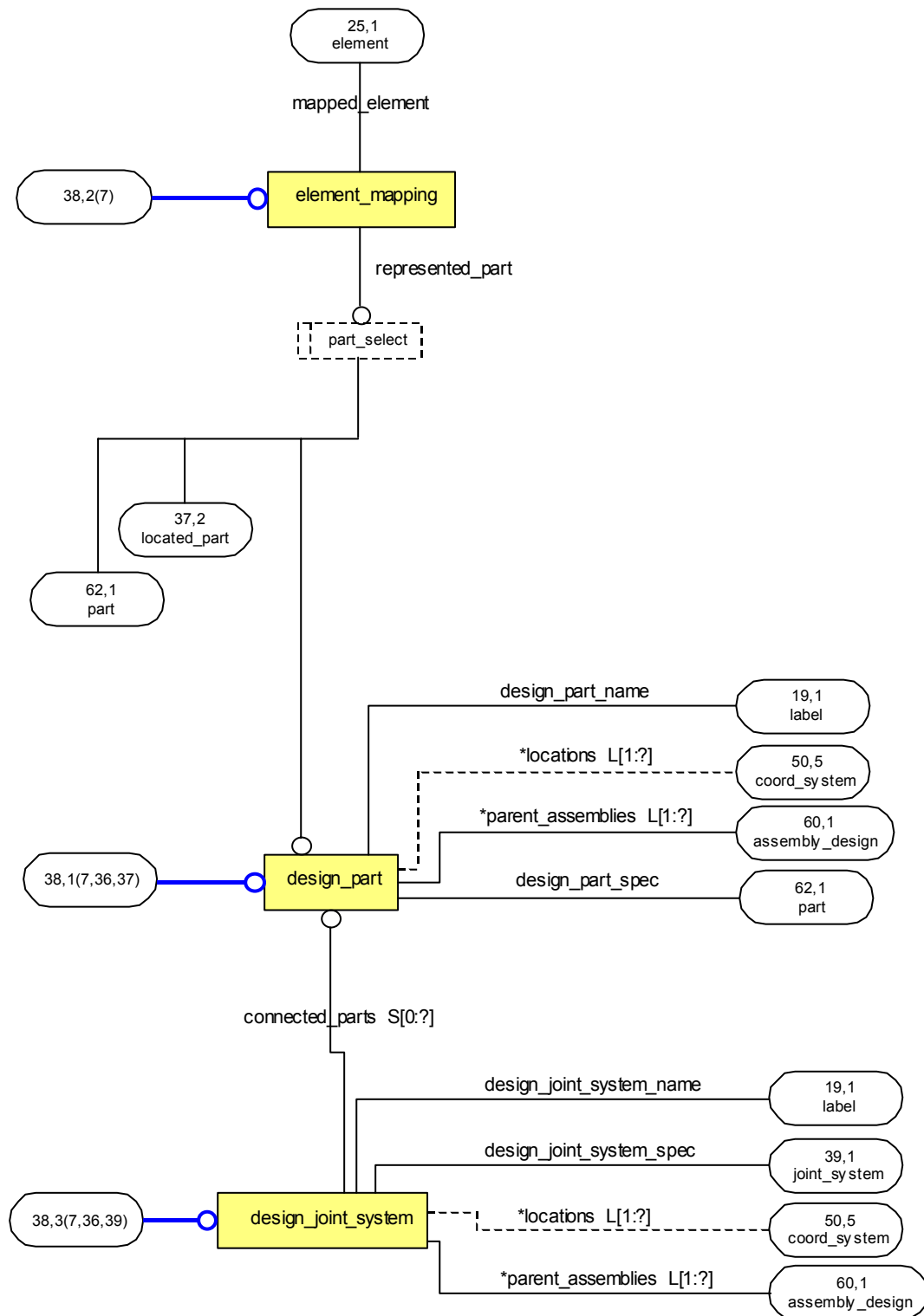


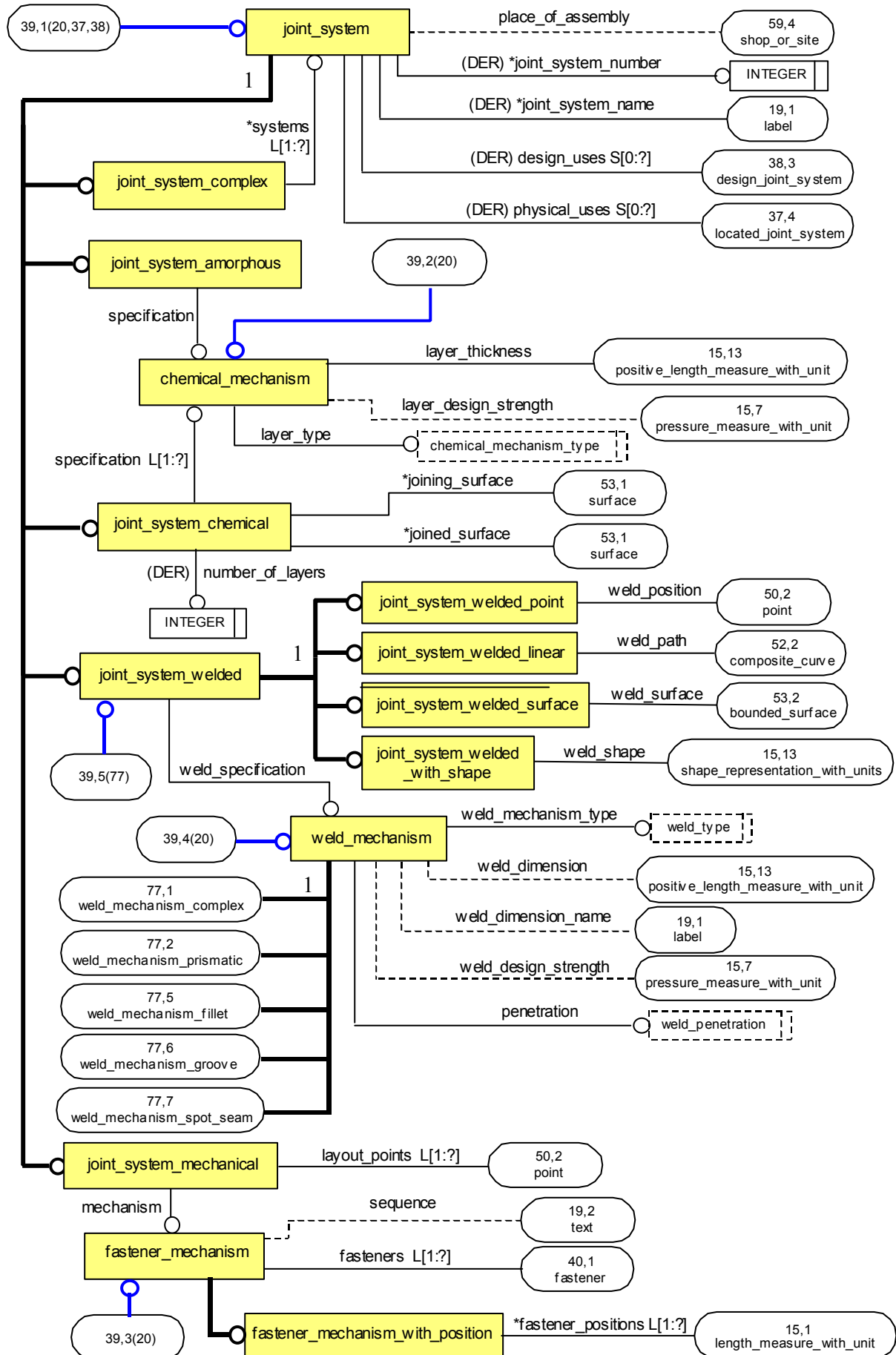
LPM/6 EXPRESS-G Diagram 36 of 82

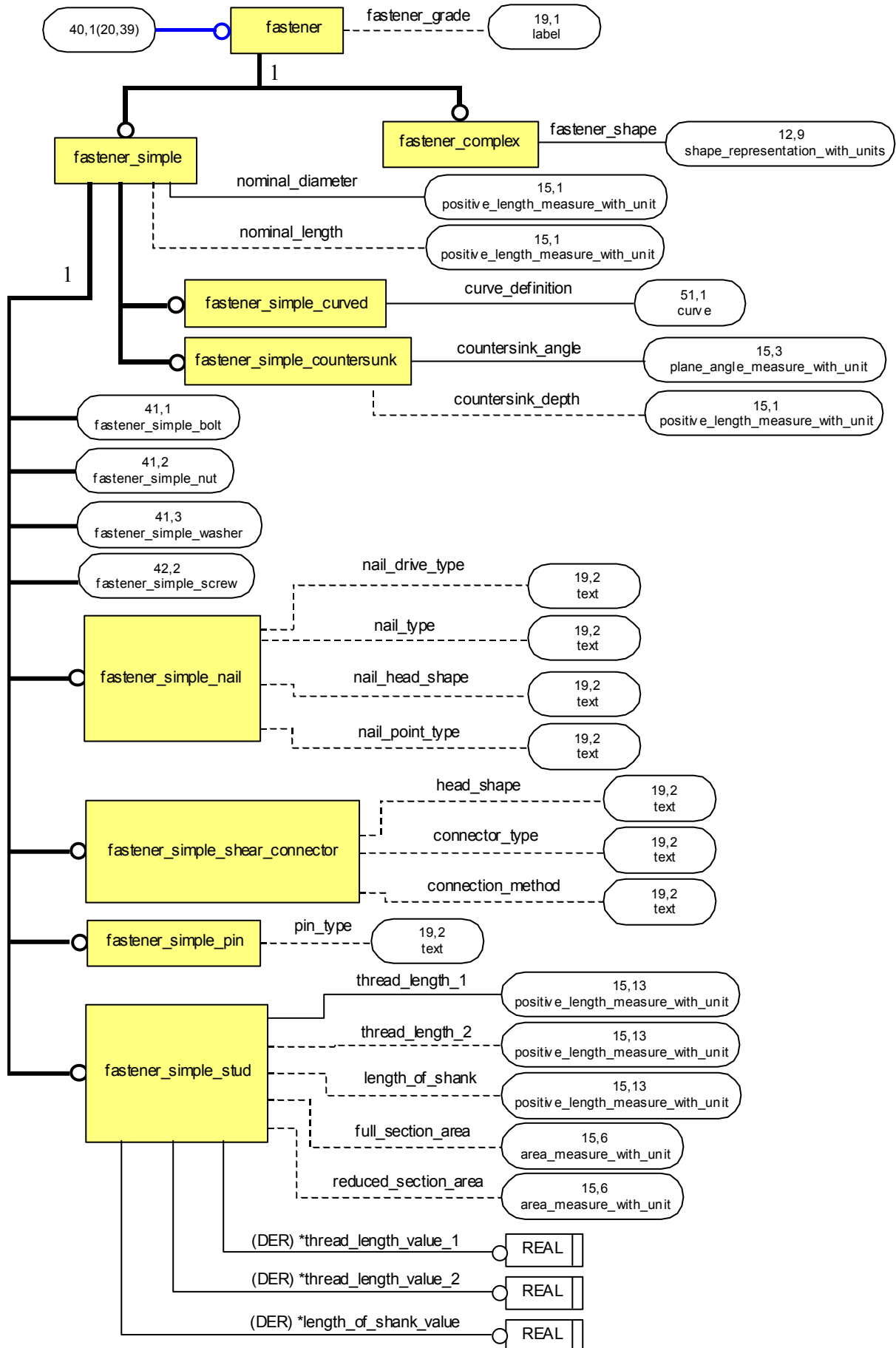


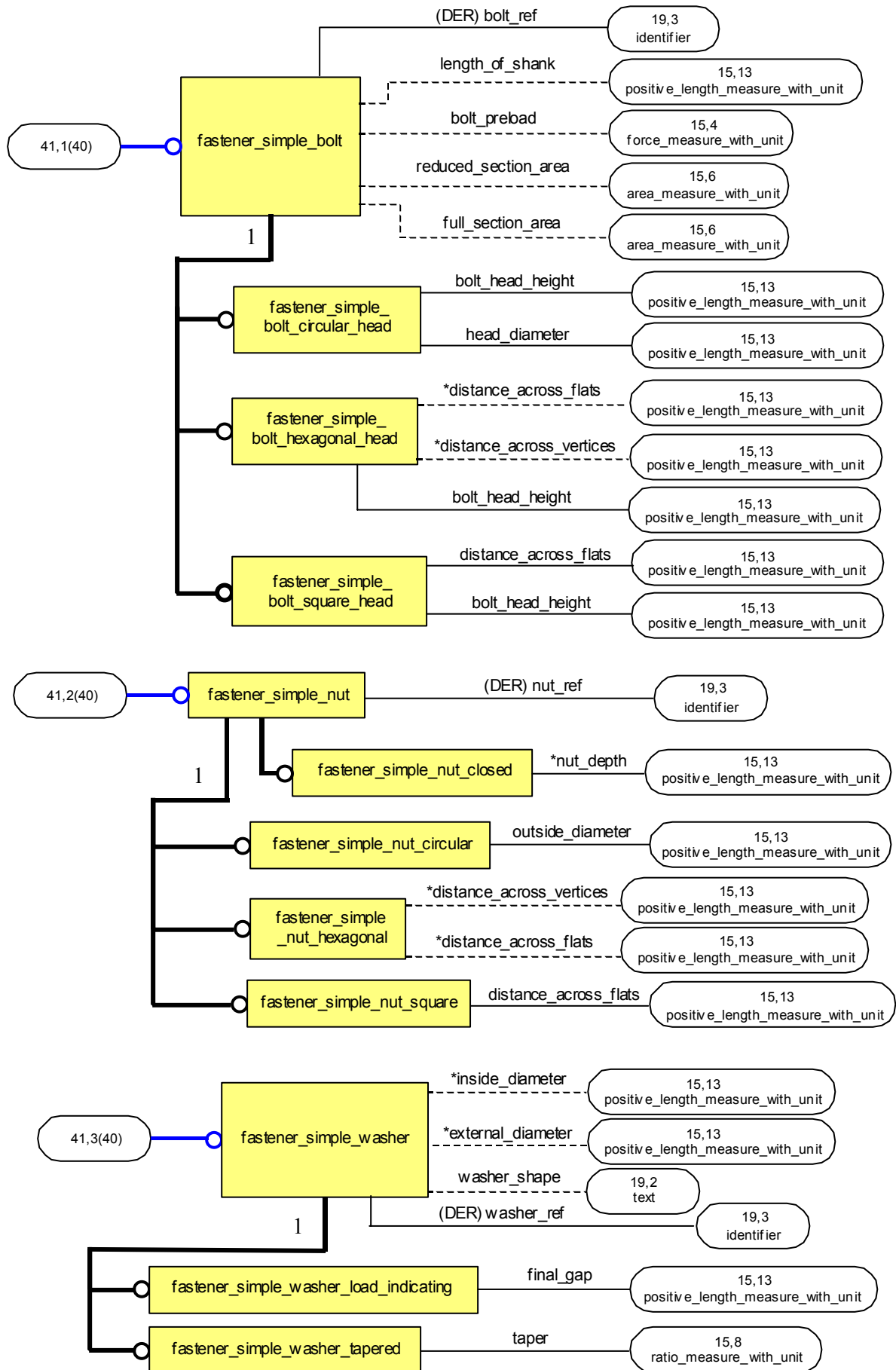
LPM/6 EXPRESS-G Diagram 37 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 38 of 82

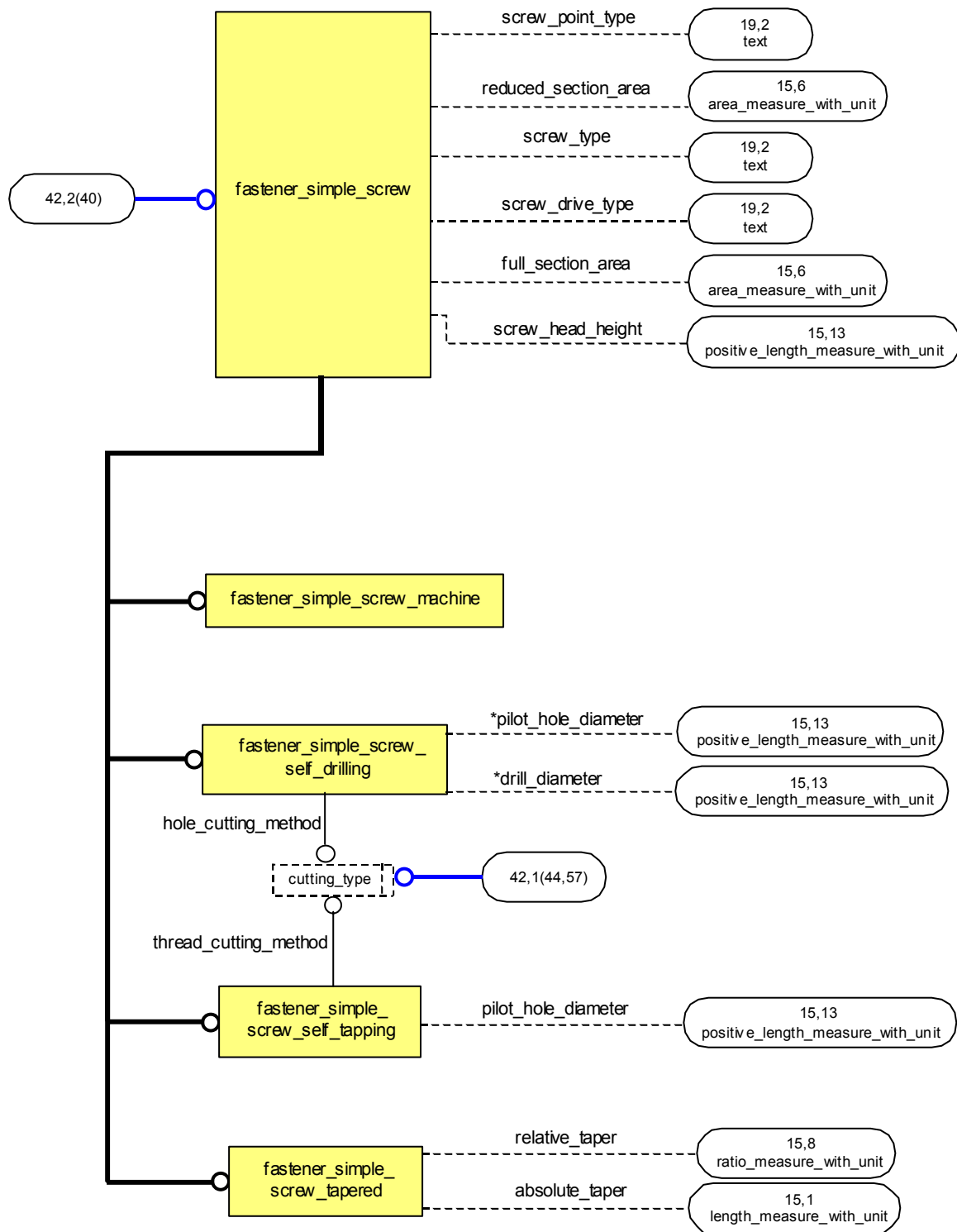


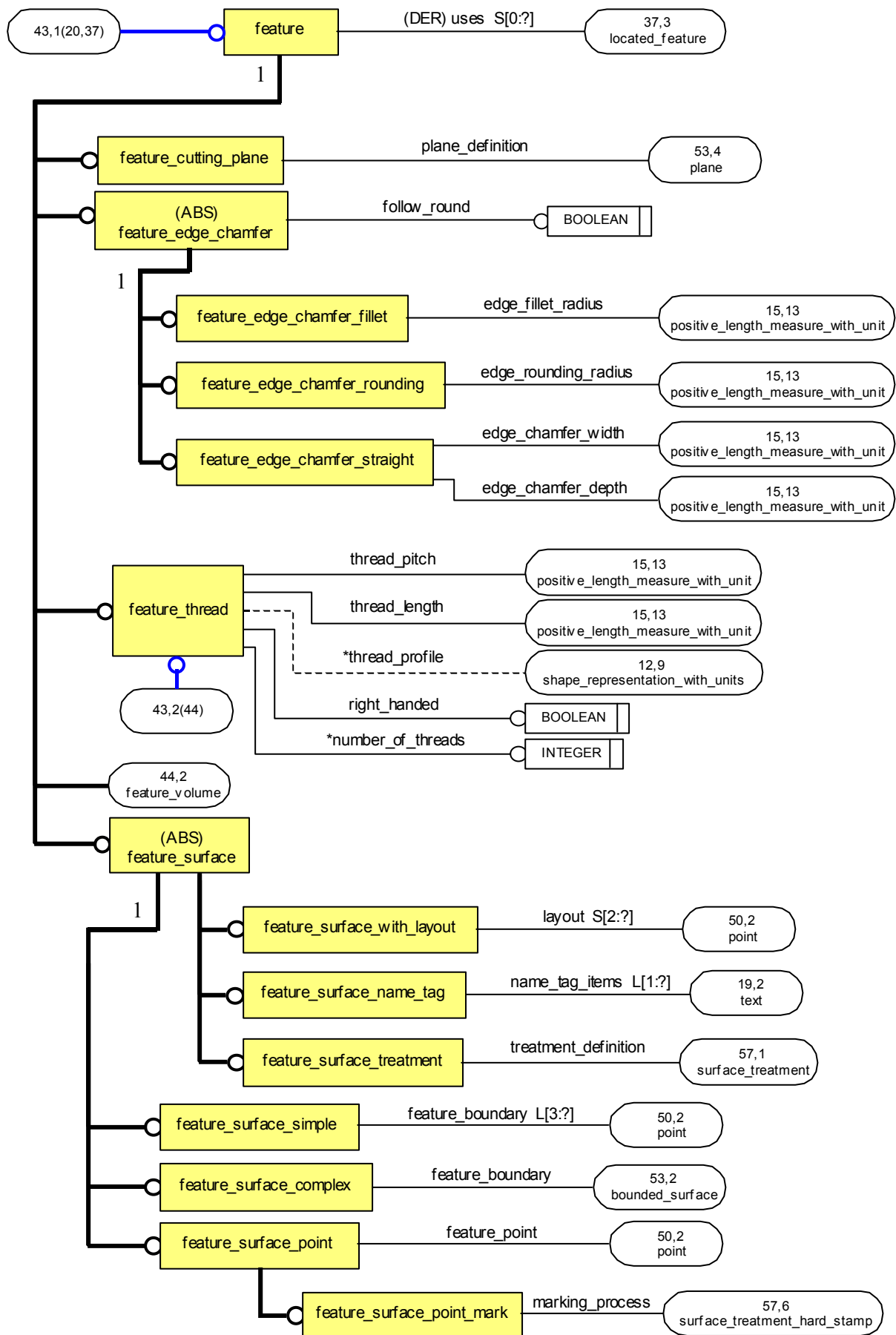
LPM/6 EXPRESS-G Diagram 39 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 40 of 82 (Modified for 2nd Edition)

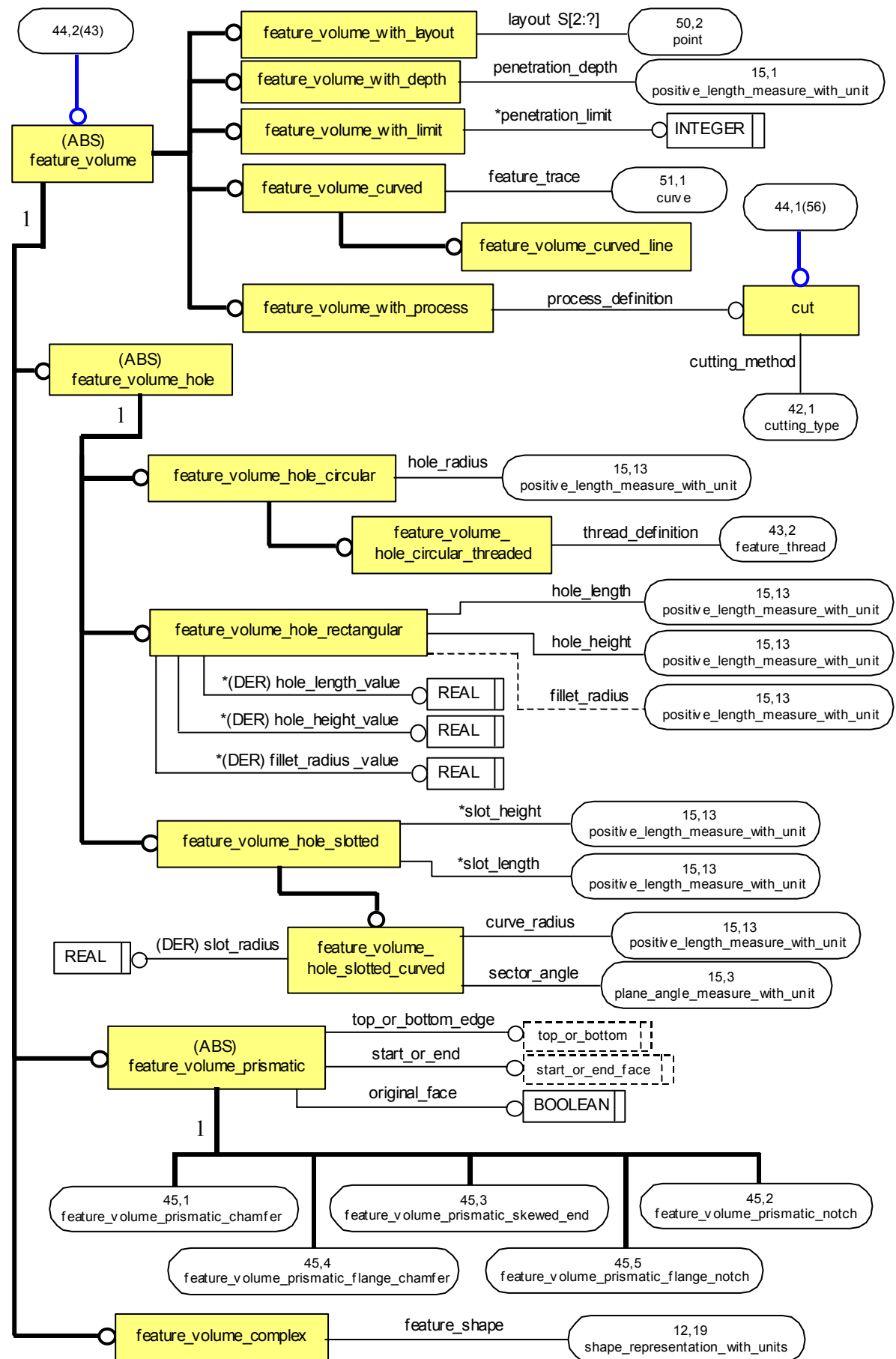
LPM/6 EXPRESS-G Diagram 41 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 42 of 82

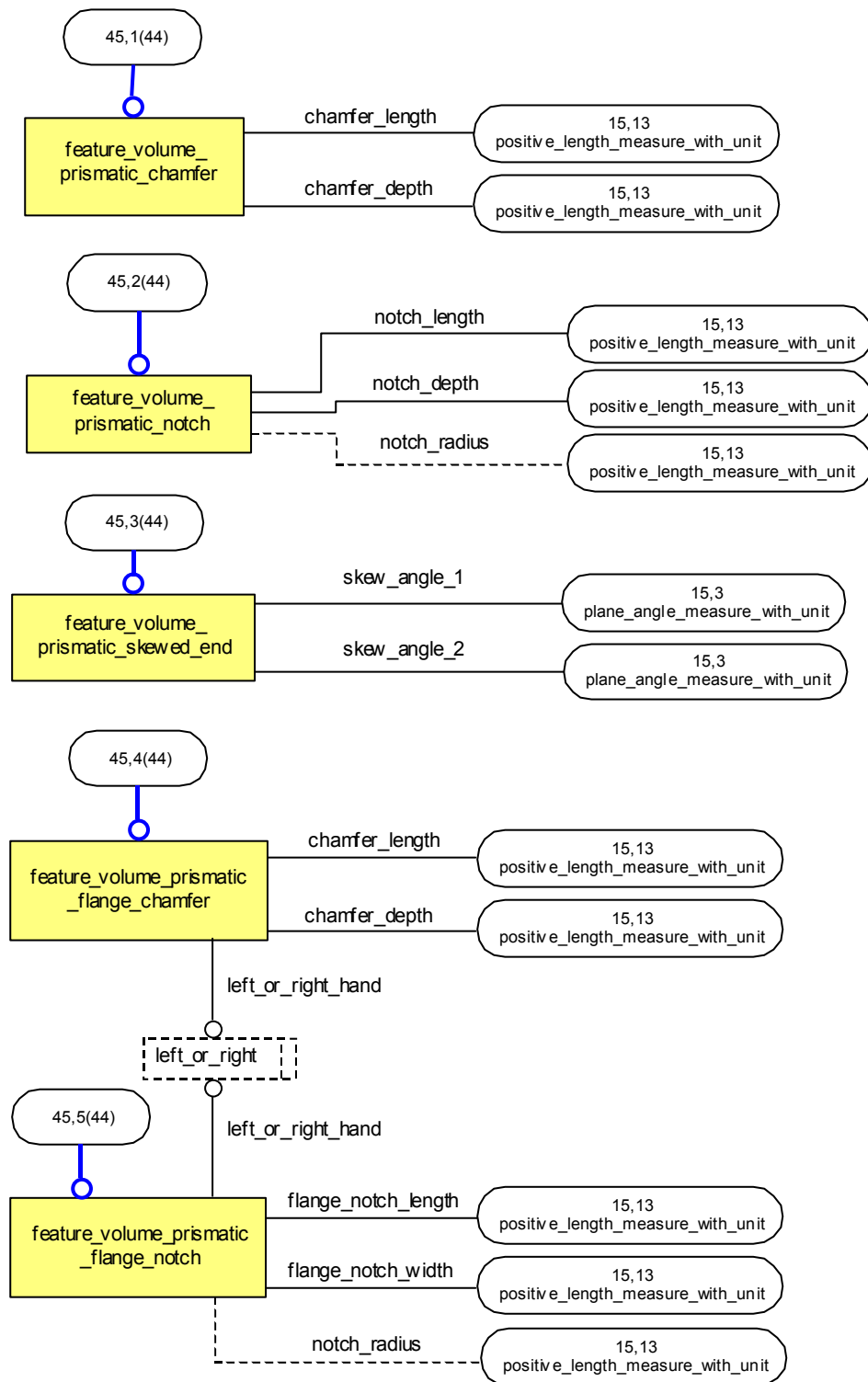


LPM/6 EXPRESS-G Diagram 43 of 82 (Modified for 2nd Edition)

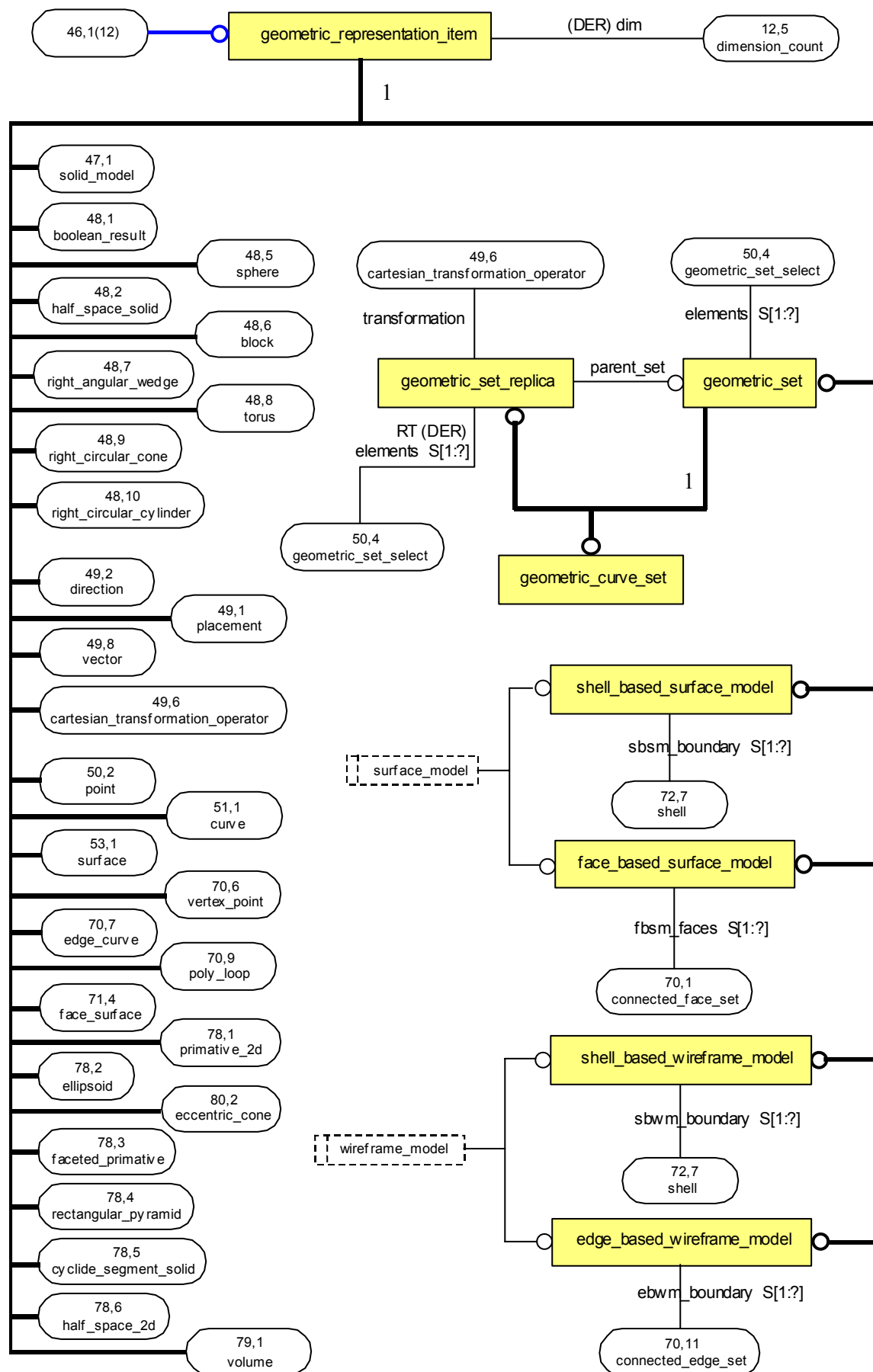
LPM/6 EXPRESS-G Diagram 44 of 82 (Modified for 2nd Edition)

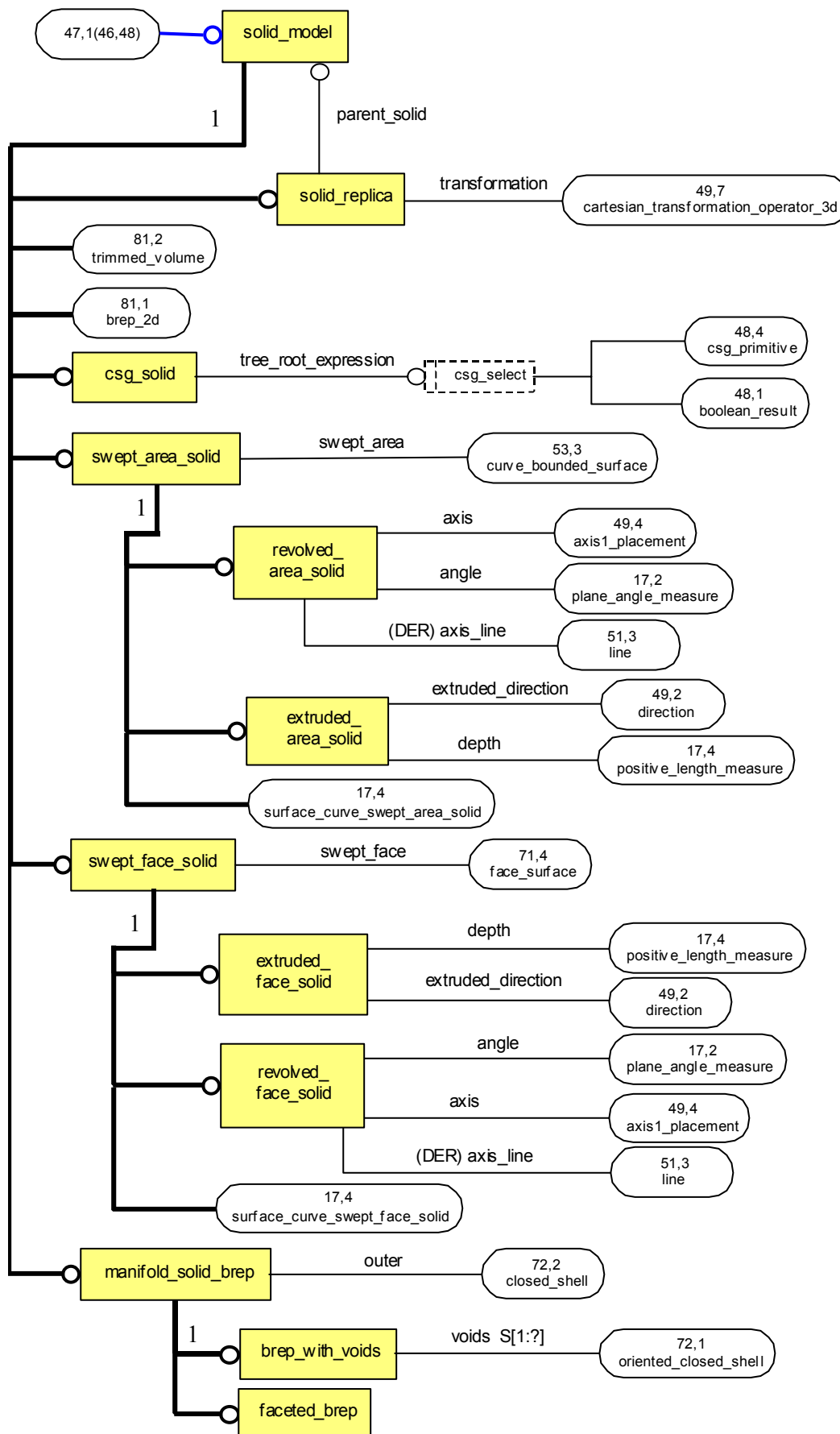


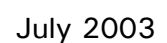
LPM/6 EXPRESS-G Diagram 45 of 82

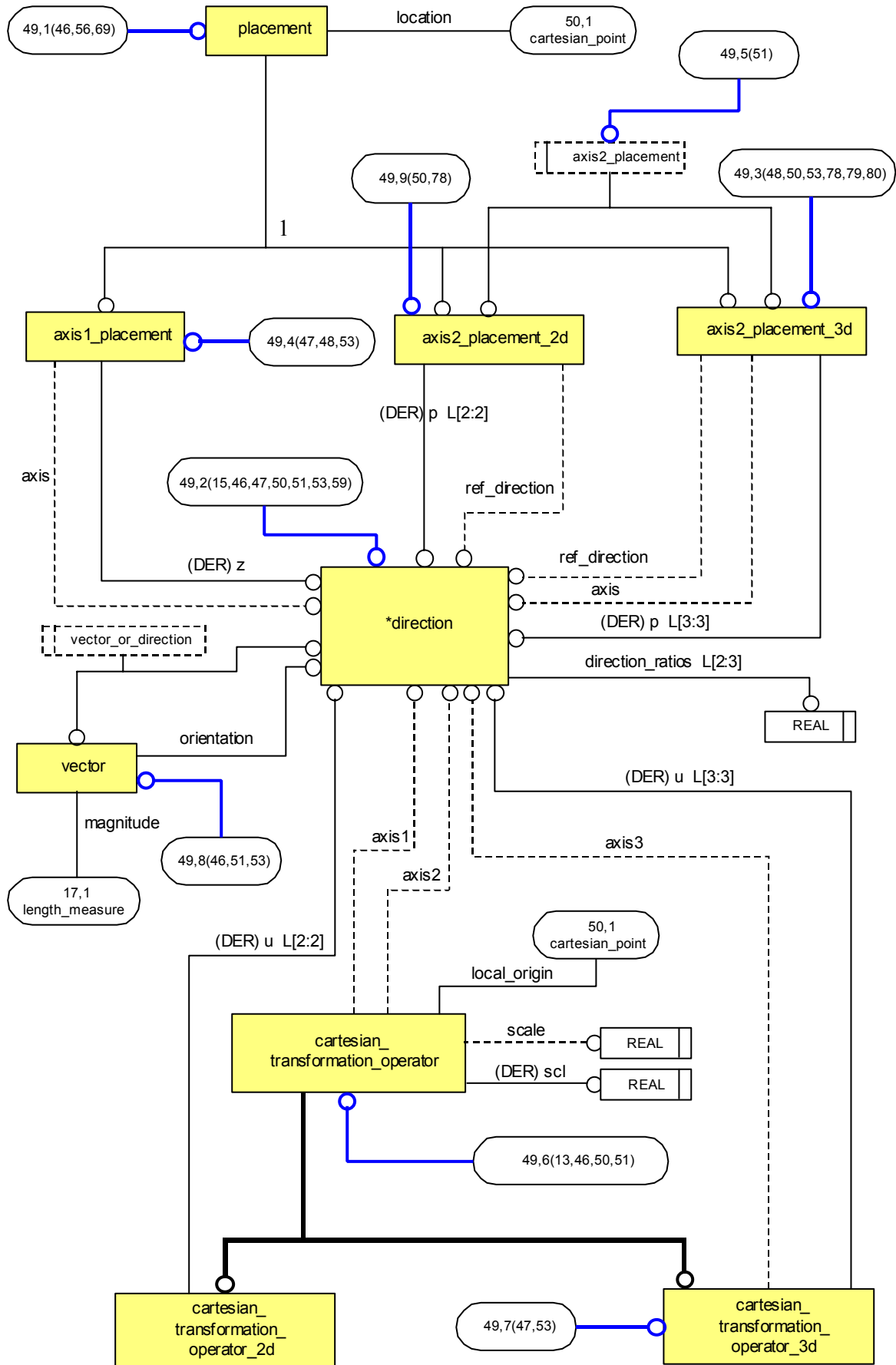


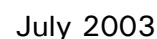
LPM/6 EXPRESS-G Diagram 46 of 82

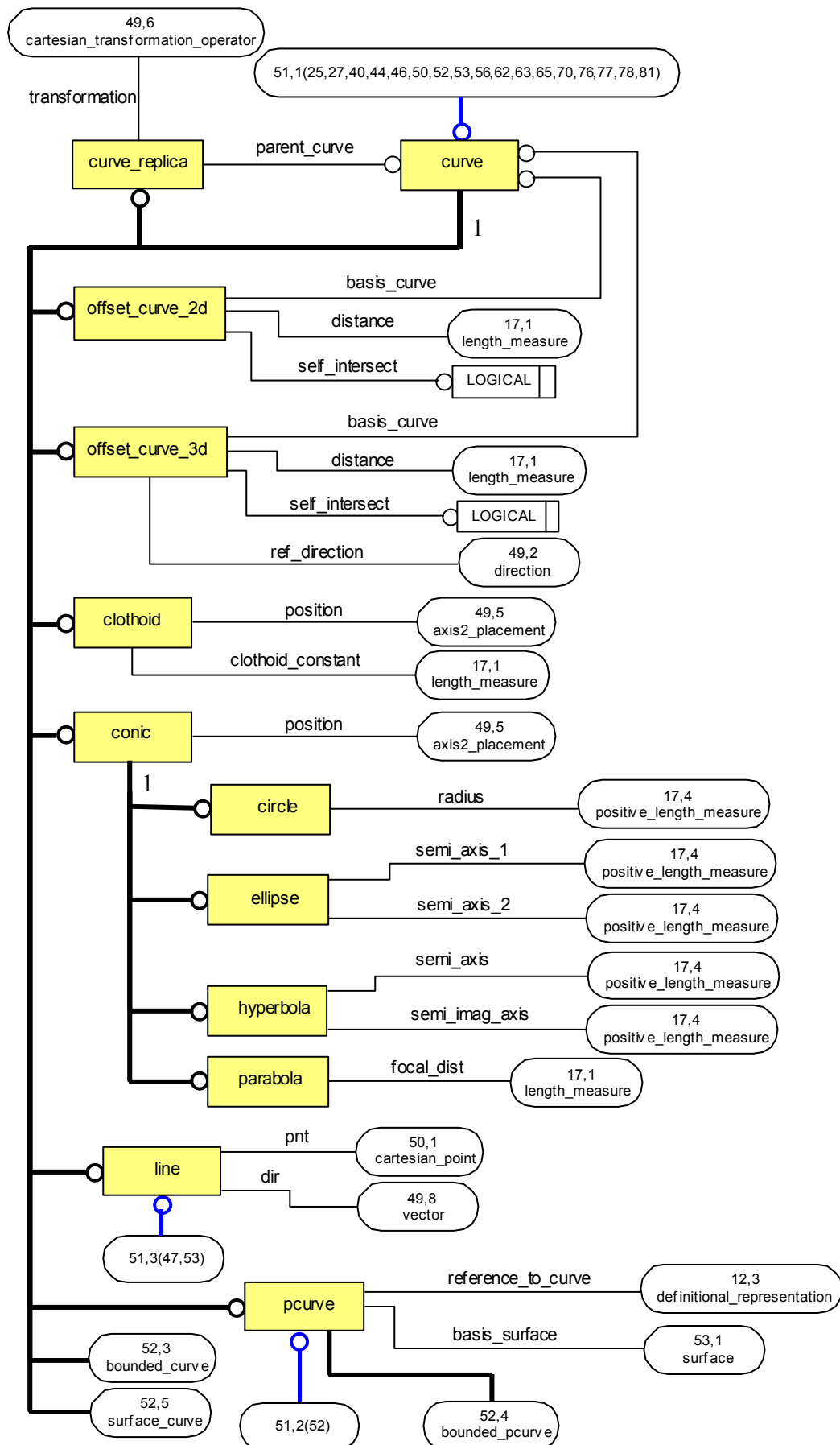


LPM/6 EXPRESS-G Diagram 47 of 82 (Modified for 2nd Edition)

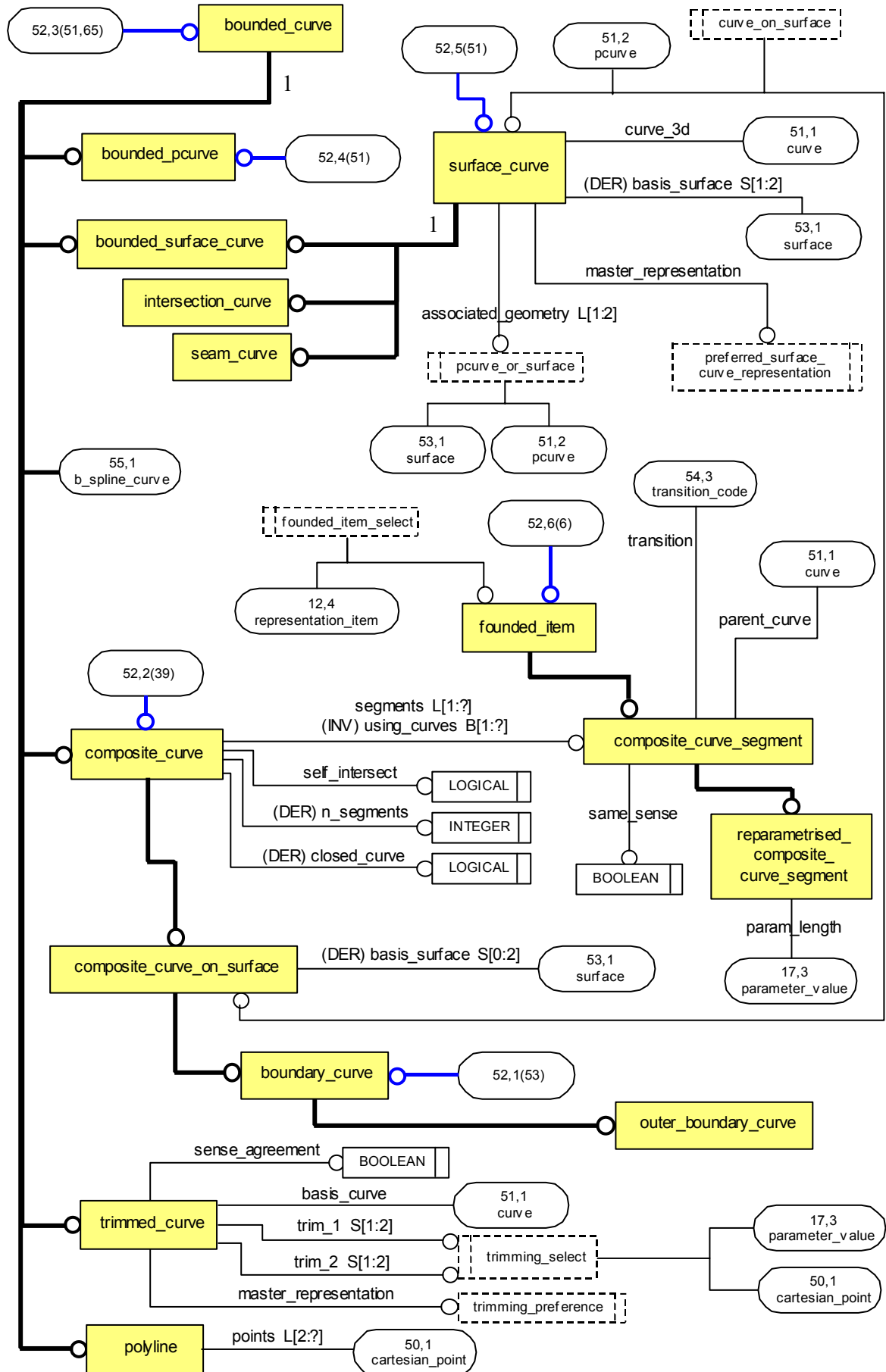


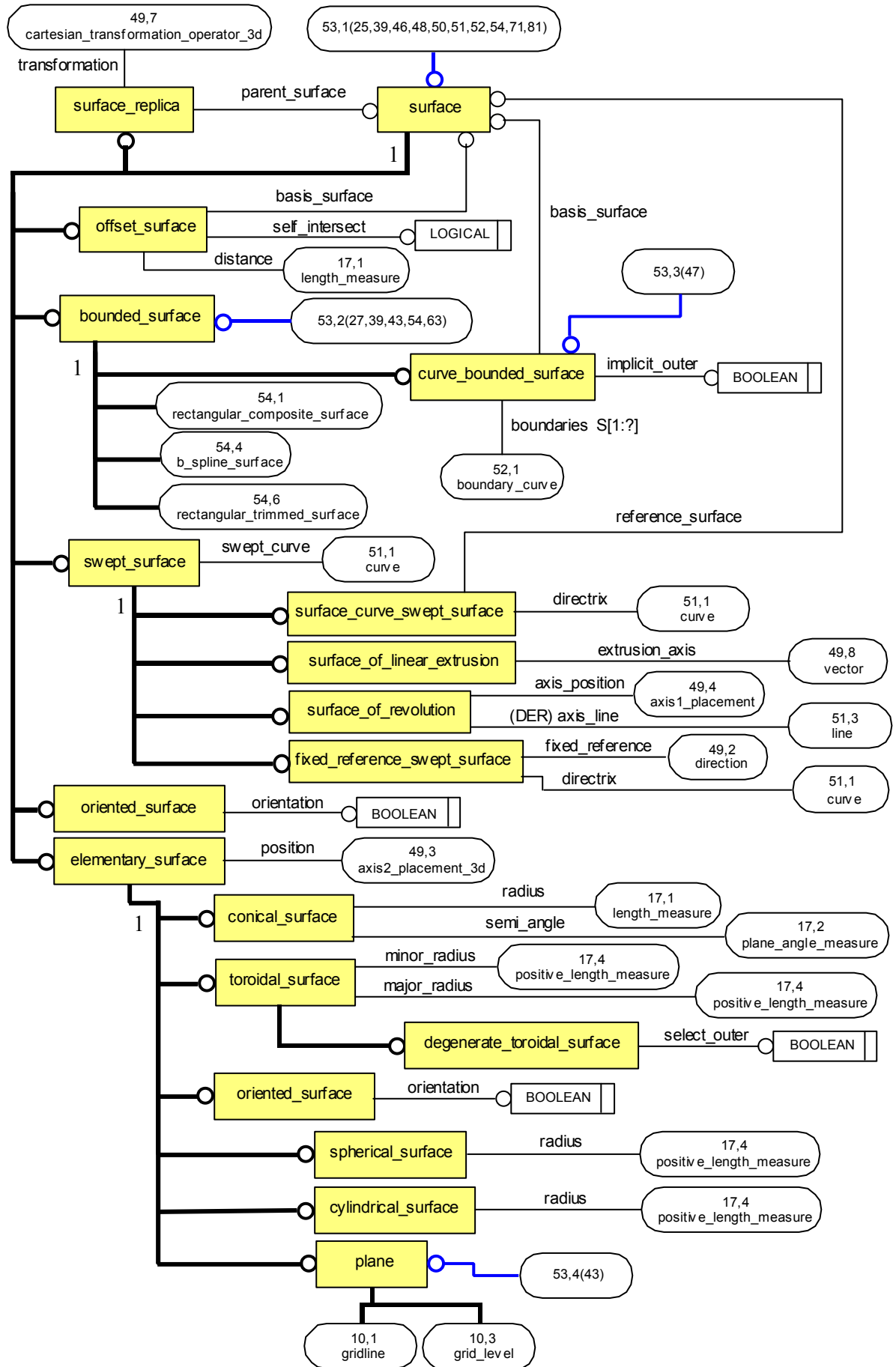
LPM/6 EXPRESS-G Diagram 49 of 82 (Modified for 2nd Edition)



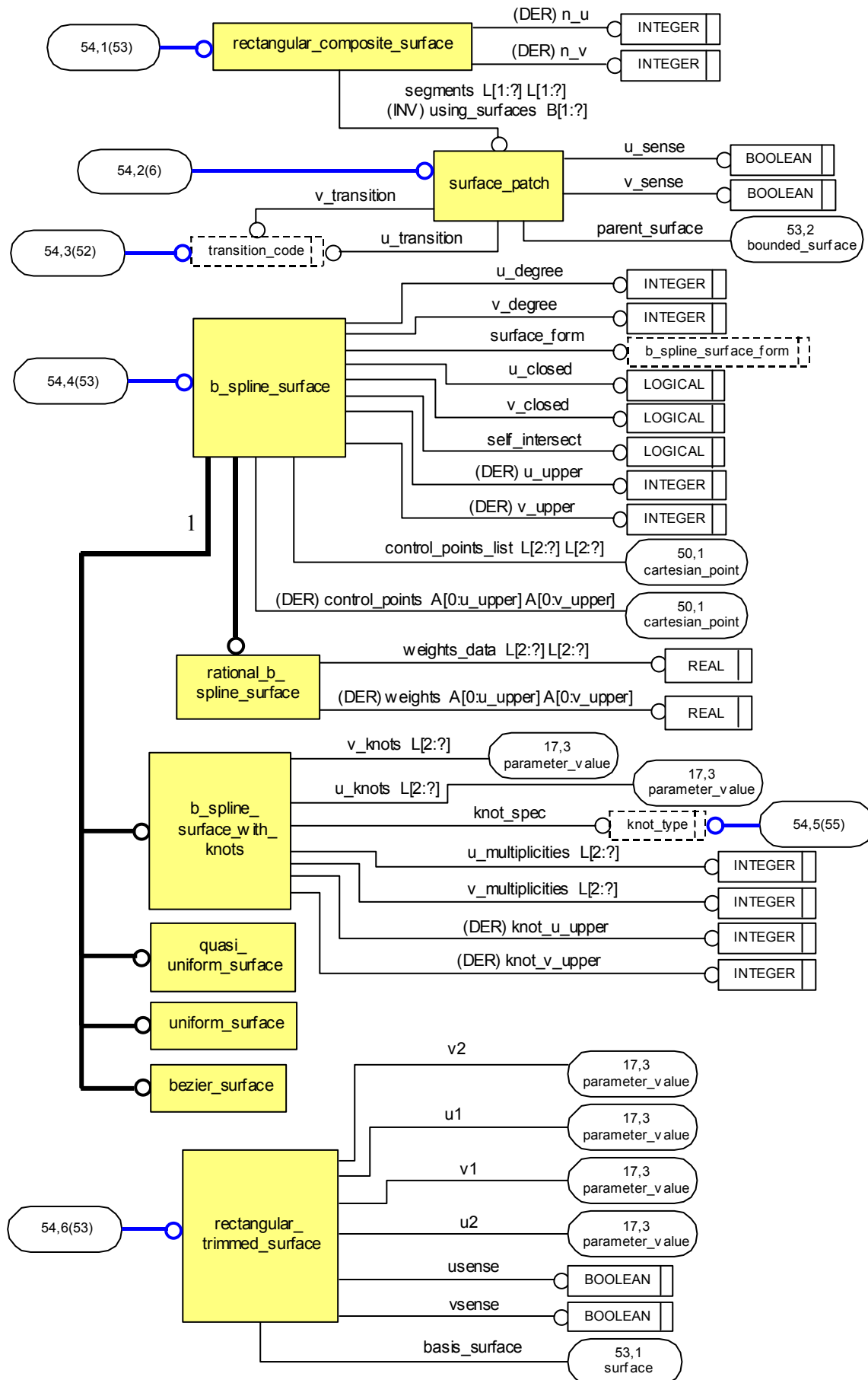
LPM/6 EXPRESS-G Diagram 51 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 52 of 82 (Modified for 2nd Edition)

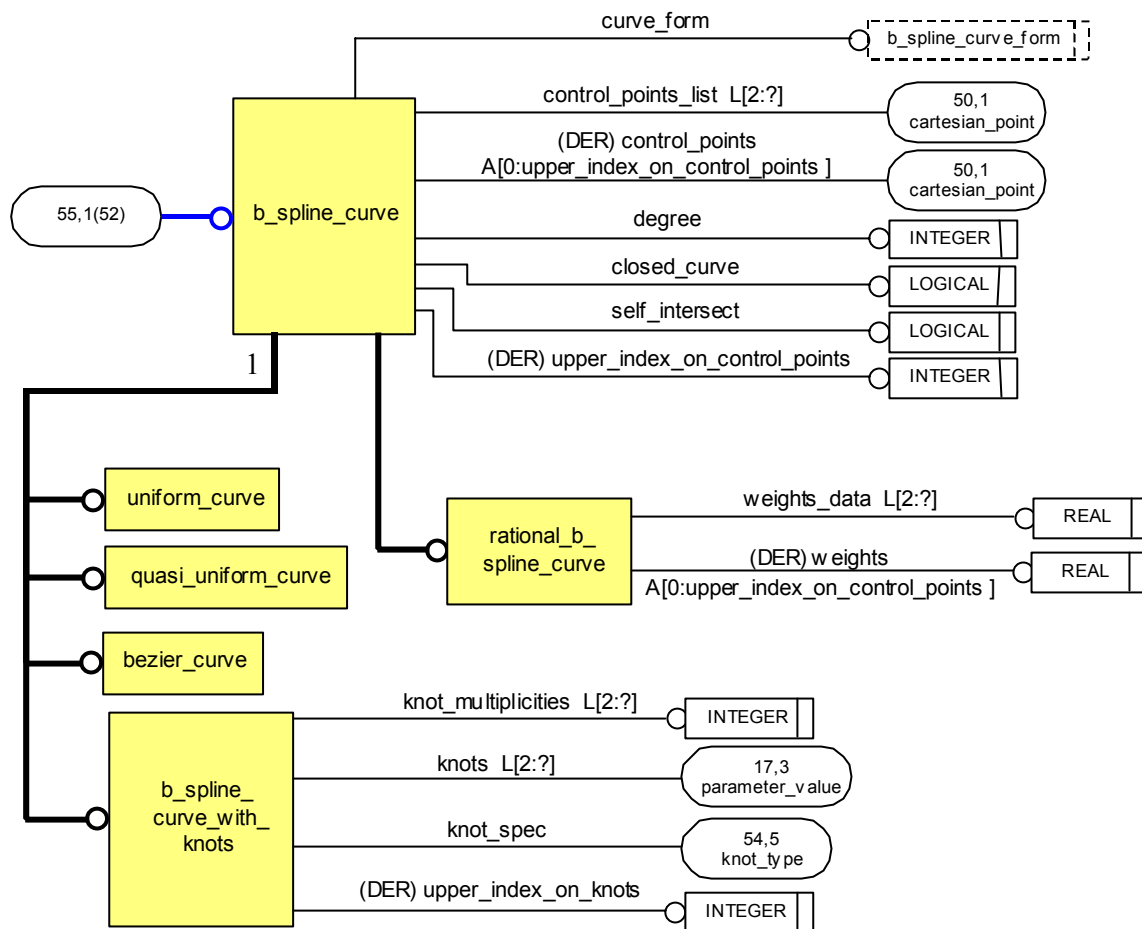


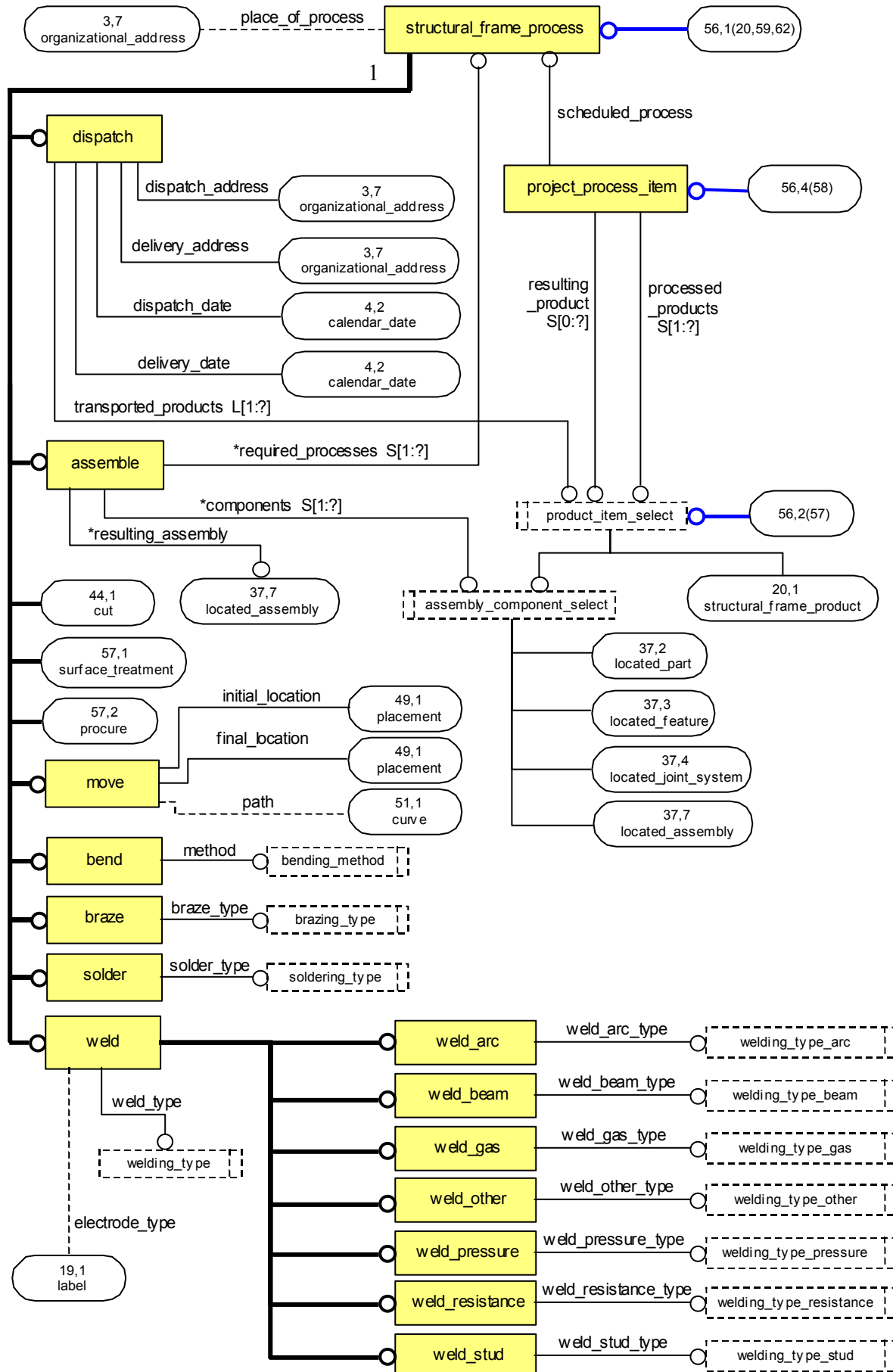
LPM/6 EXPRESS-G Diagram 53 of 82 (Modified for 2nd Edition)

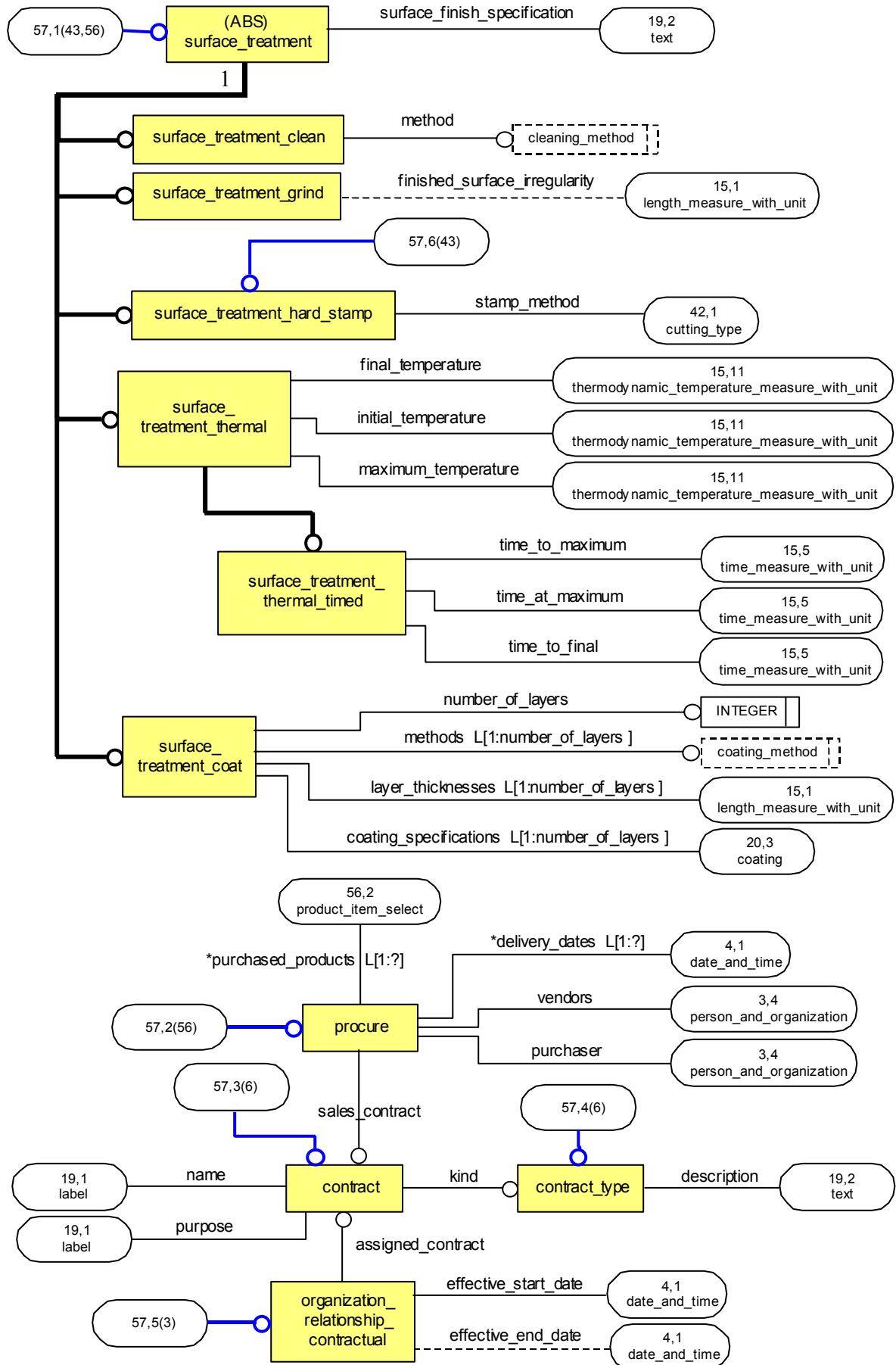
LPM/6 EXPRESS-G Diagram 54 of 82

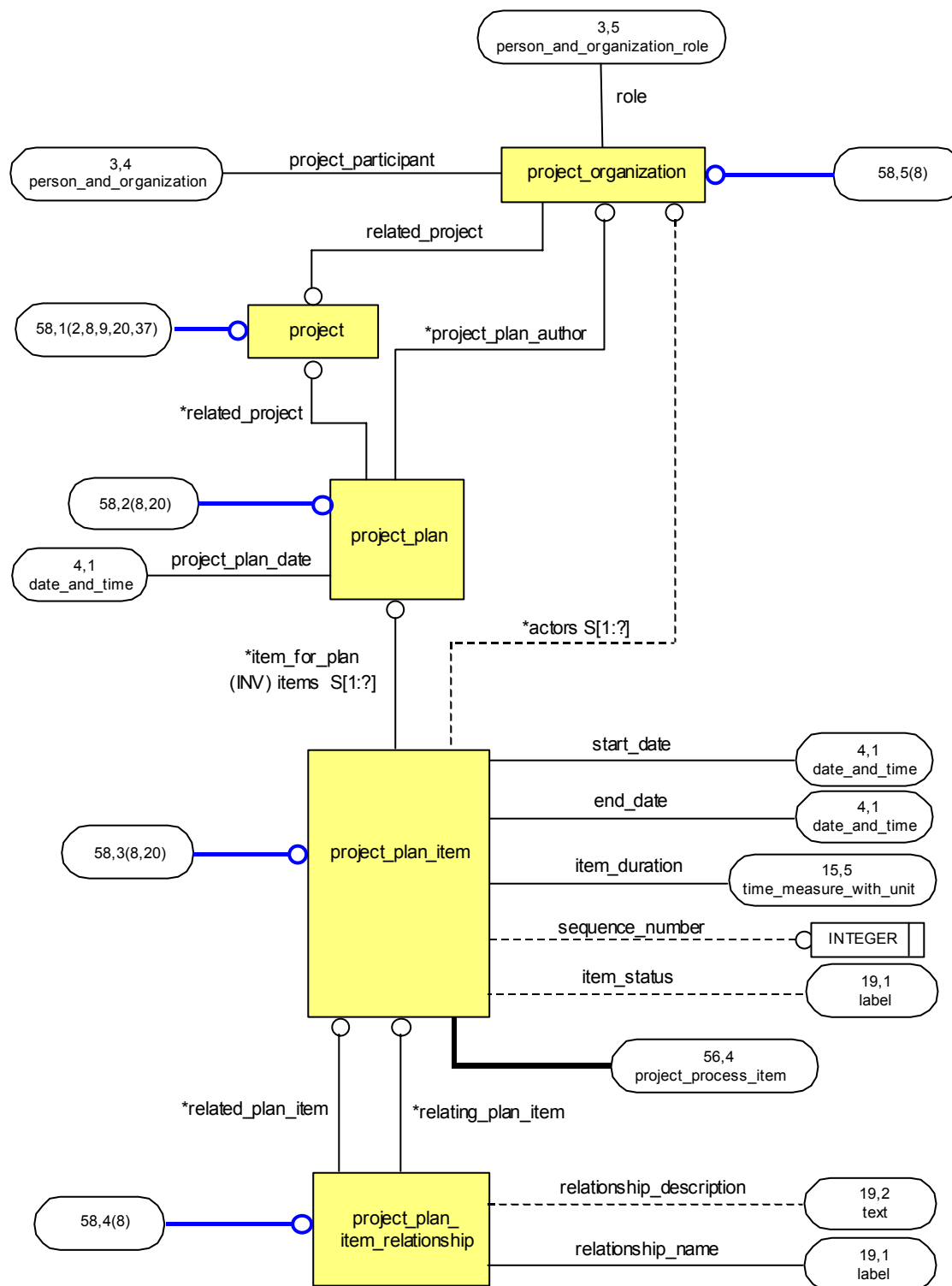


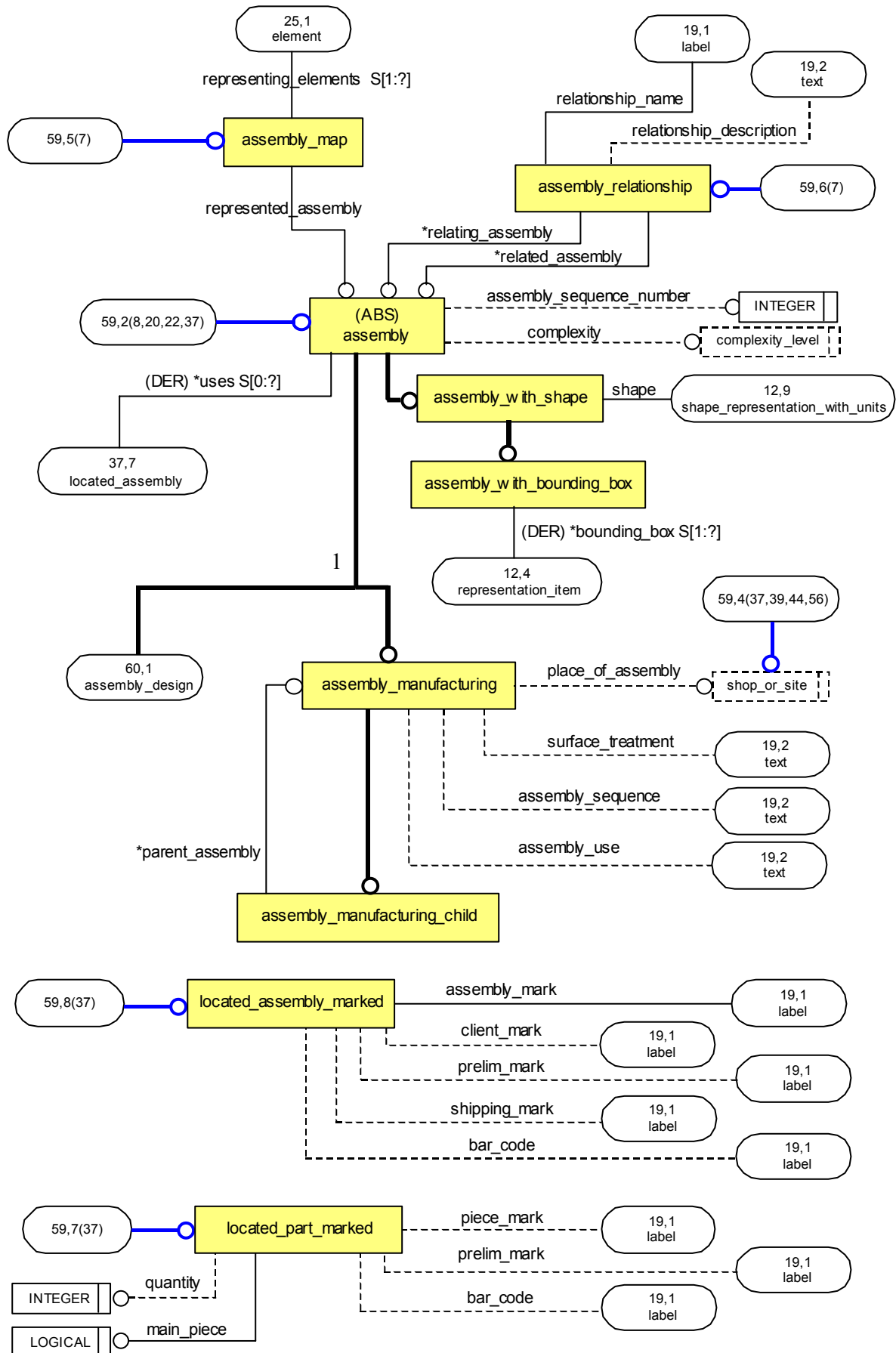
LPM/6 EXPRESS-G Diagram 55 of 82

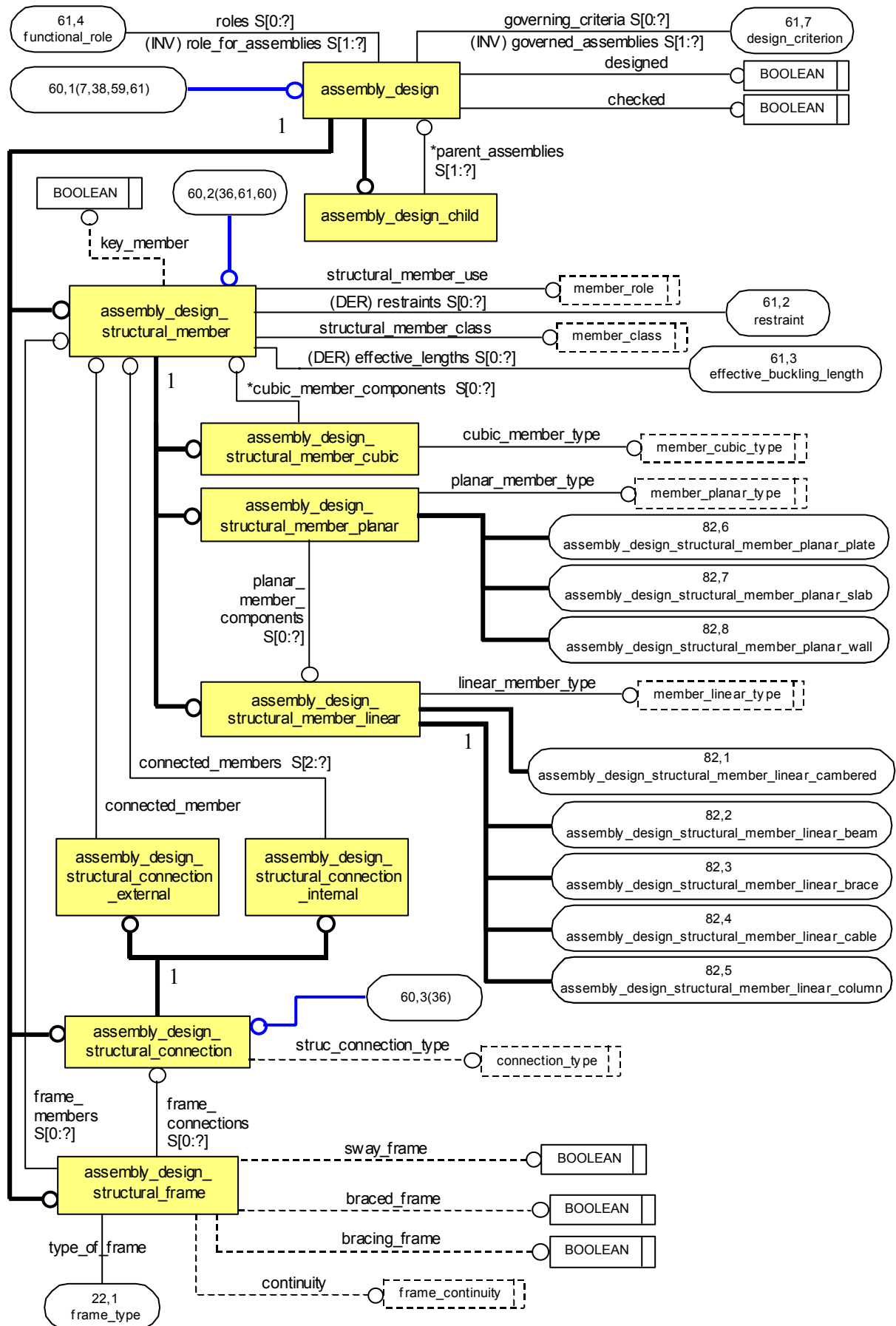


LPM/6 EXPRESS-G Diagram 56 of 82 (Modified for 2nd Edition)

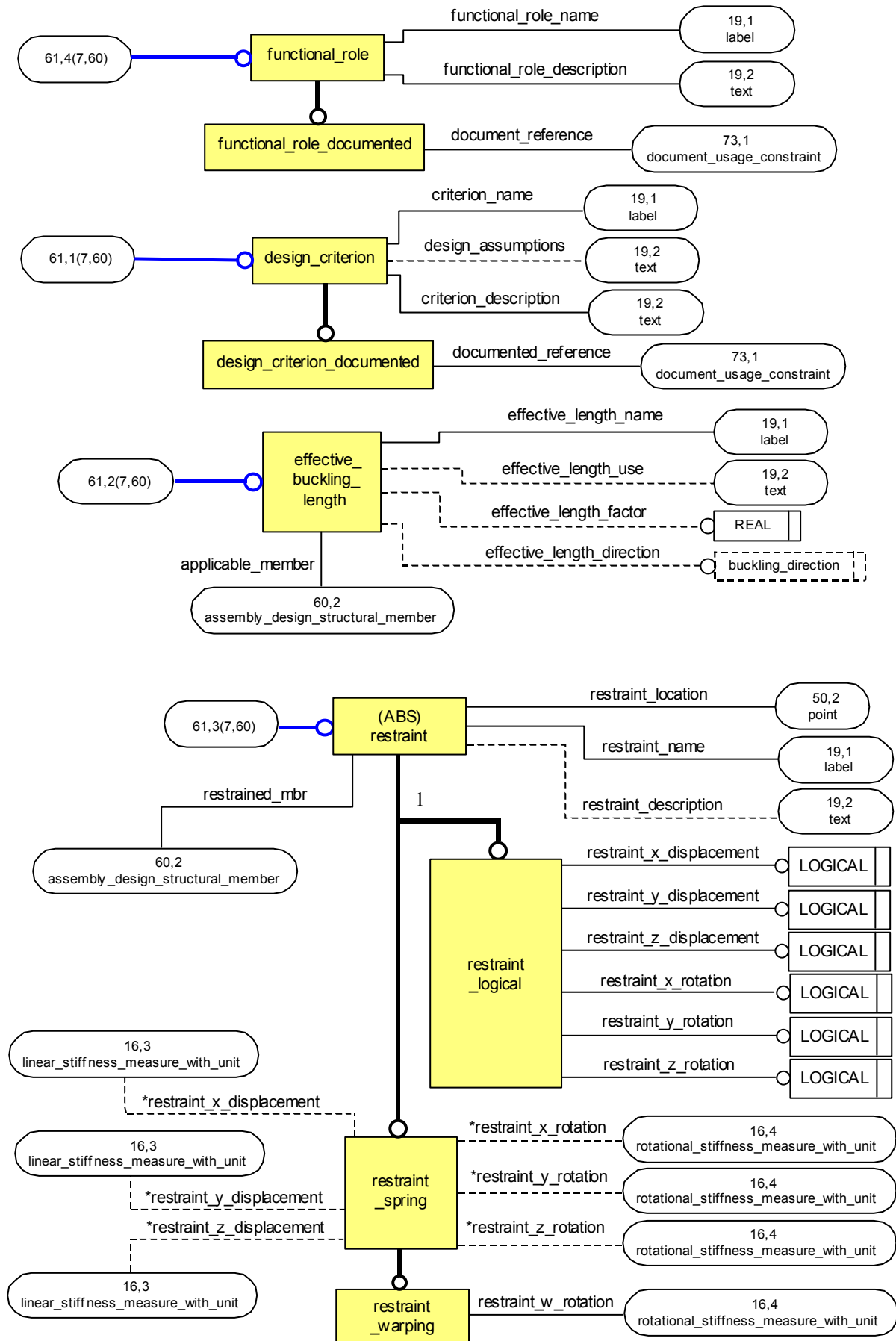
LPM/6 EXPRESS-G Diagram 57 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 58 of 82 (Modified for 2nd Edition)

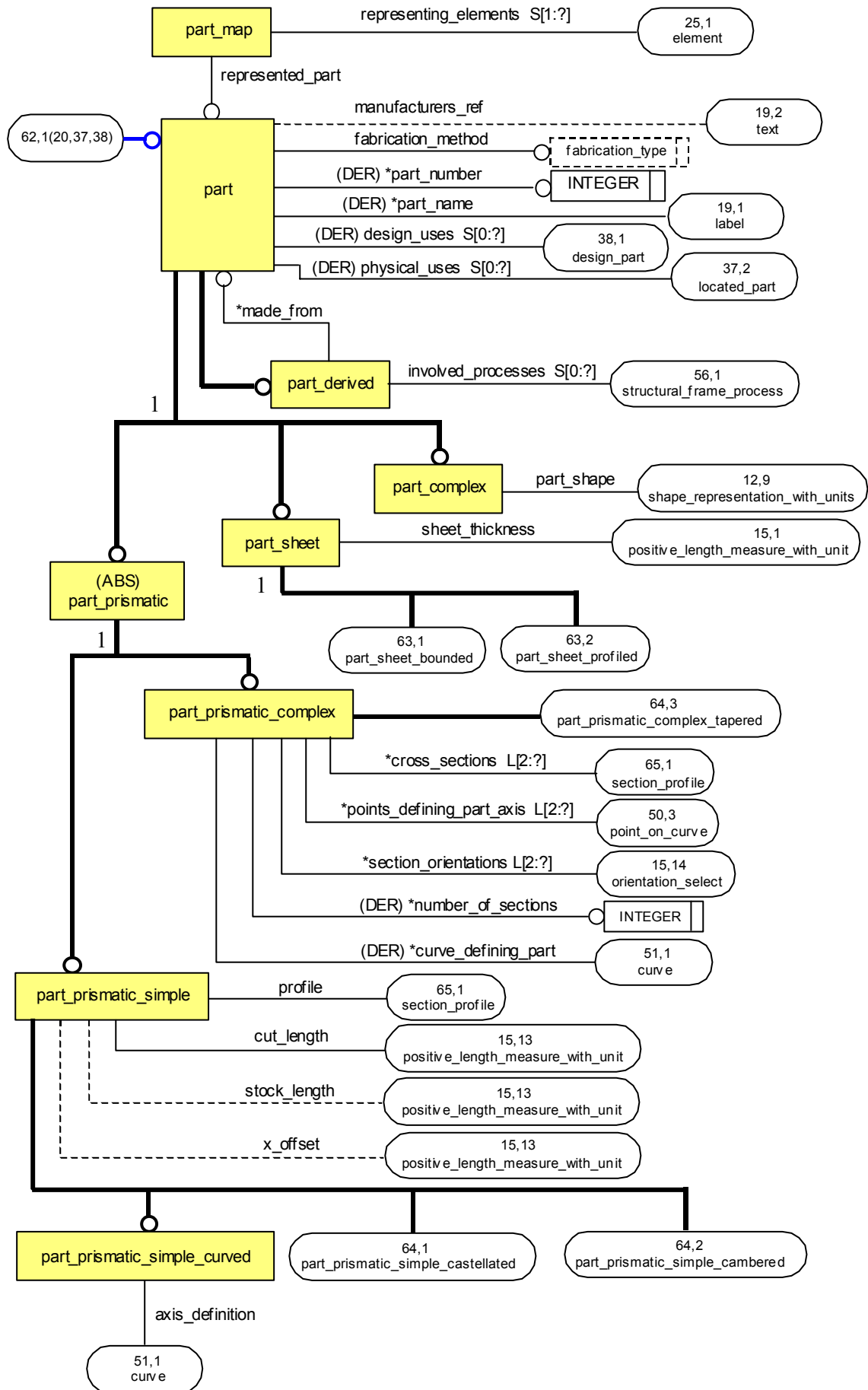
LPM/6 EXPRESS-G Diagram 59 of 82 (Modified for 2nd Edition)

LPM/6 EXPRESS-G Diagram 60 of 82 (Modified for 2nd Edition)

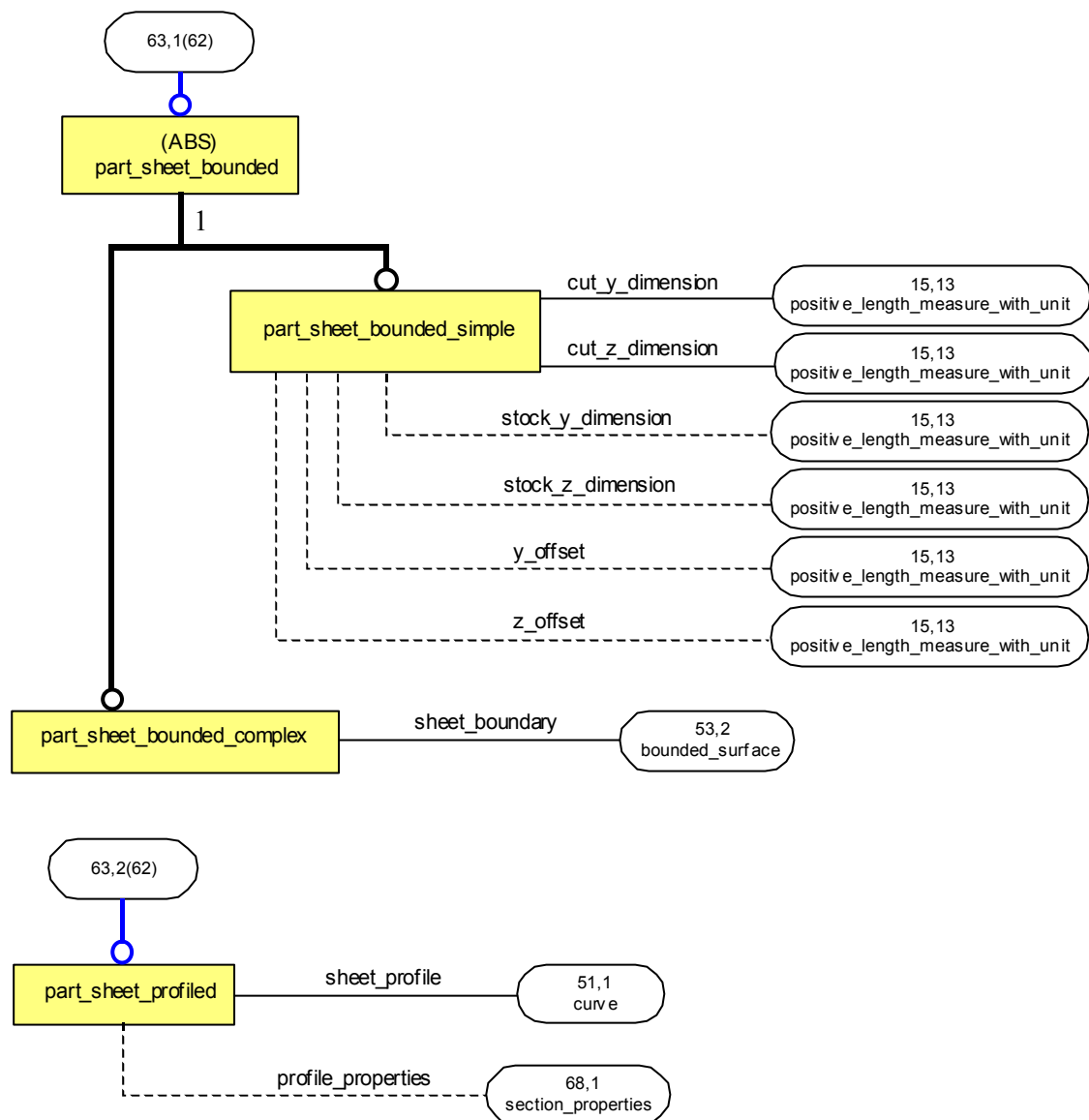
LPM/6 EXPRESS-G Diagram 61 of 82

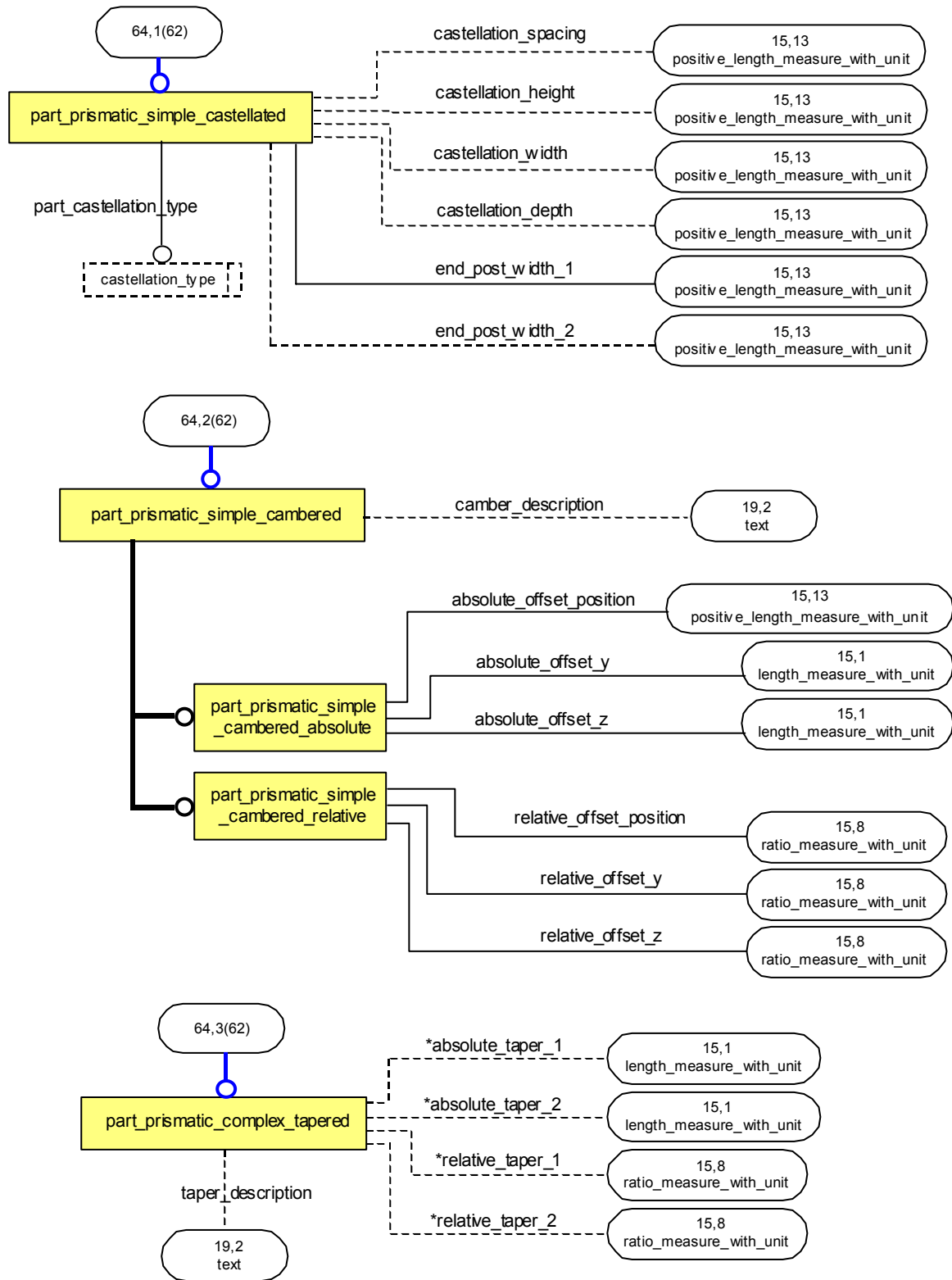


LPM/6 EXPRESS-G Diagram 62 of 82 (Modified for 2nd Edition)

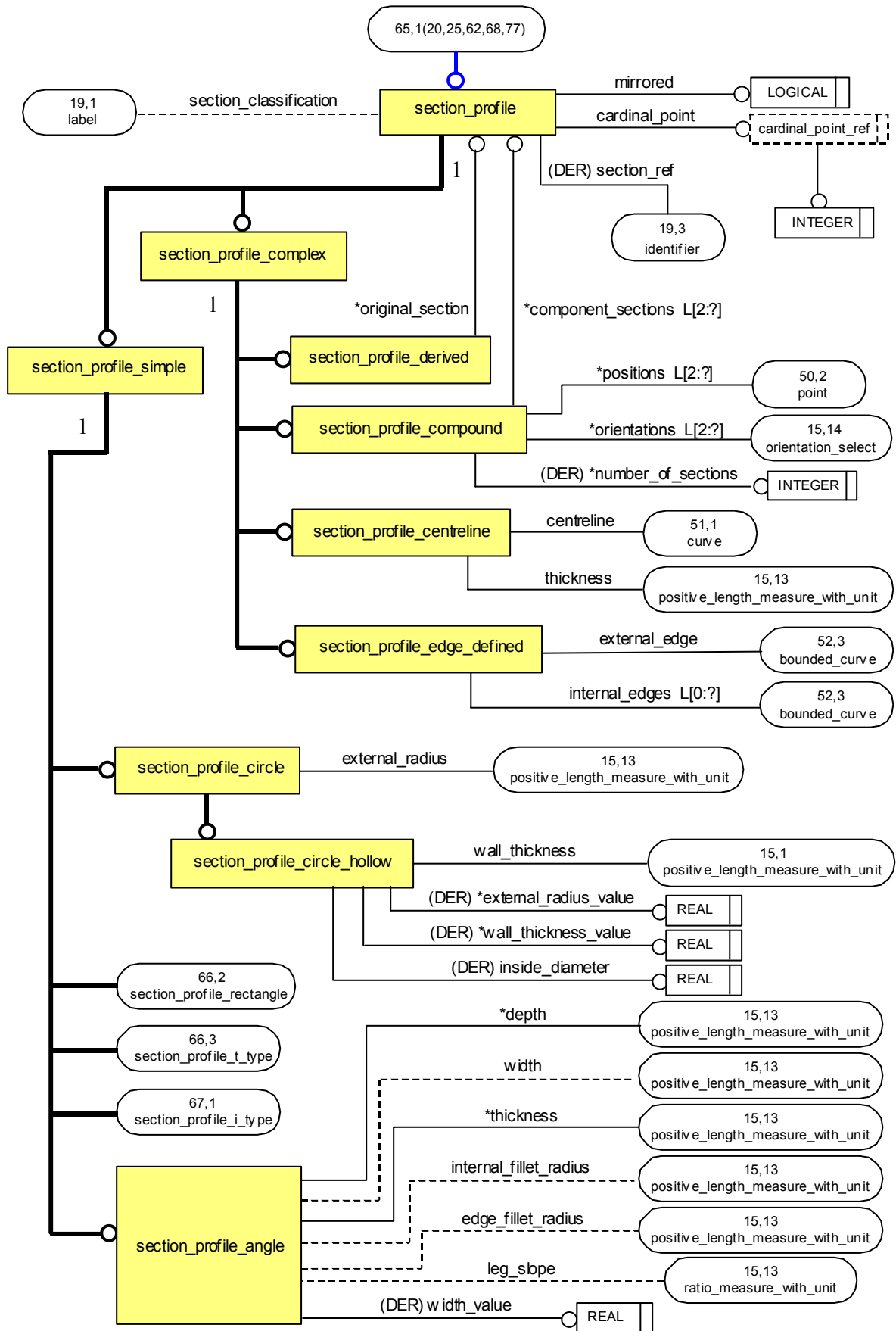


LPM/6 EXPRESS-G Diagram 63 of 82

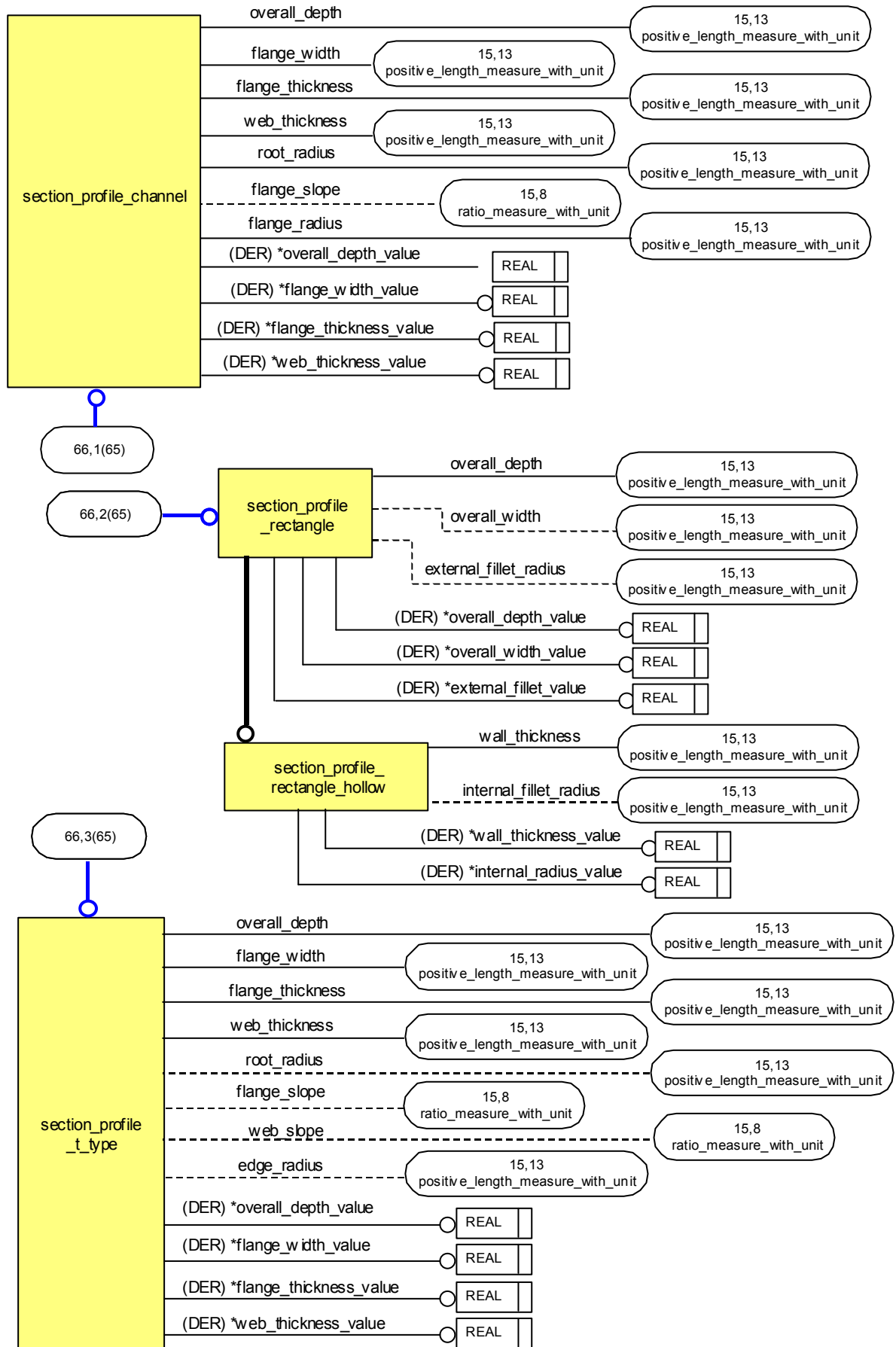


LPM/6 EXPRESS-G Diagram 64 of 82 (Modified for 2nd Edition)

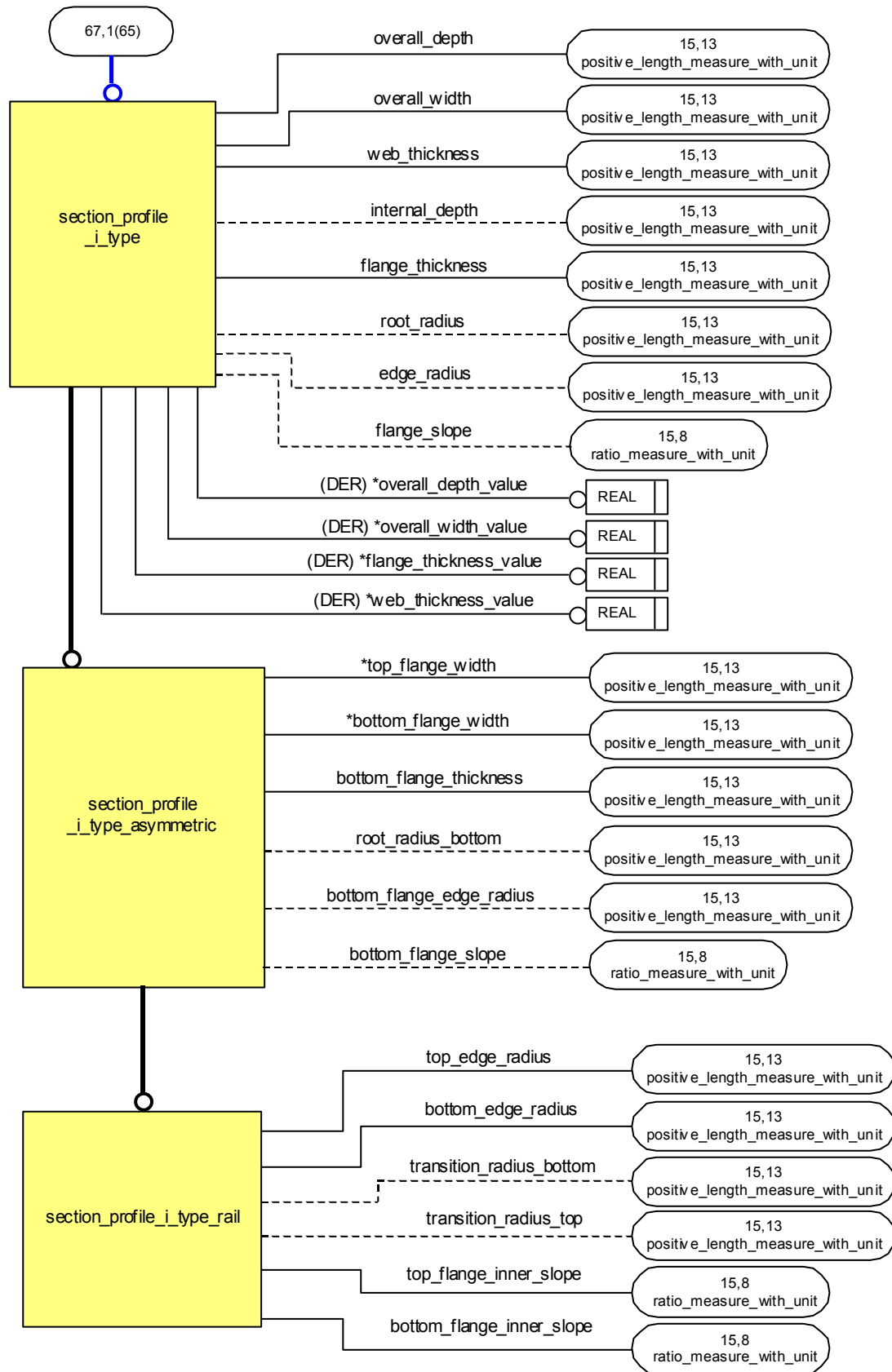
LPM/6 EXPRESS-G Diagram 65 of 82 (Modified for 2nd Edition)



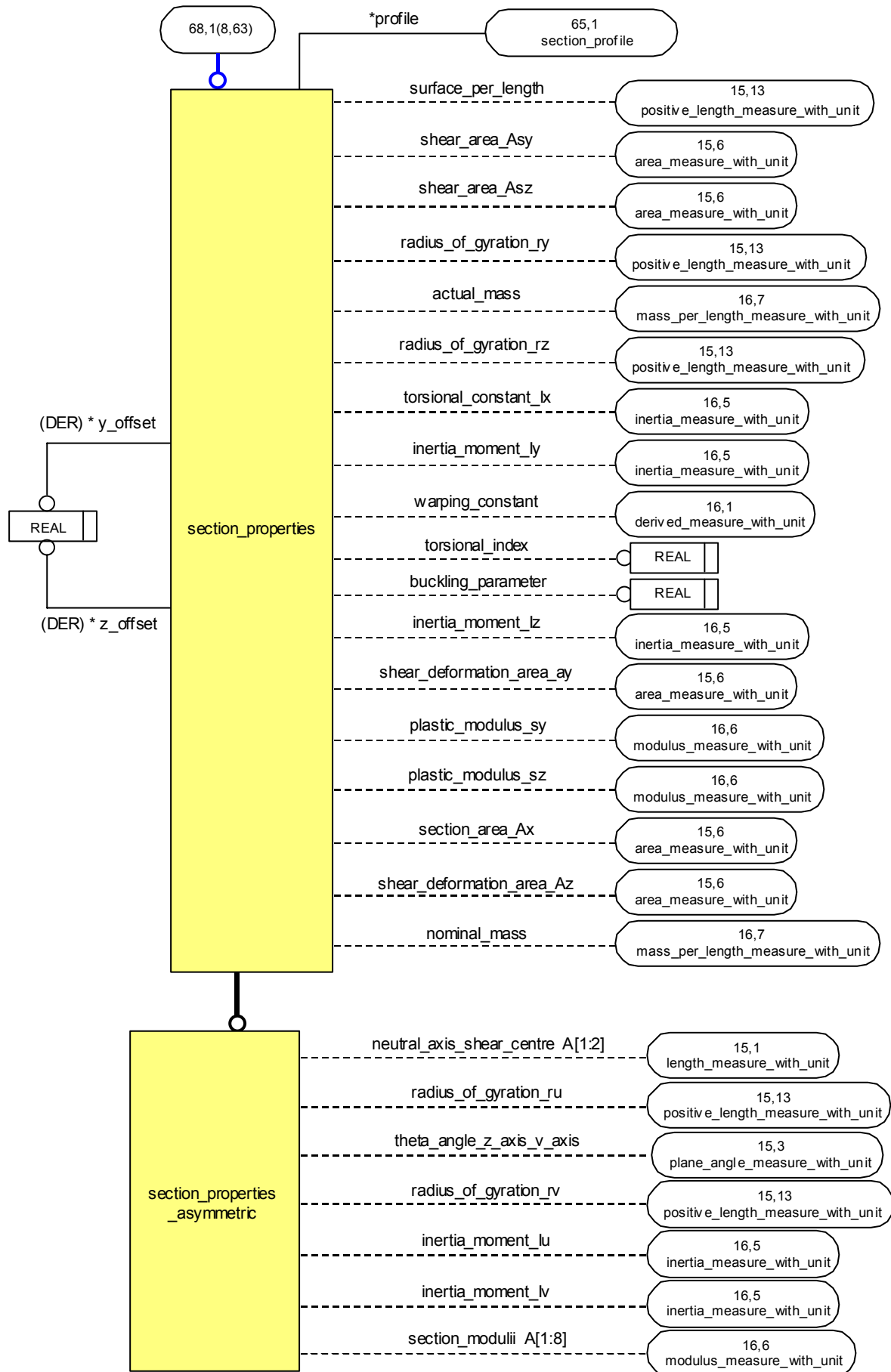
LPM/6 EXPRESS-G Diagram 66 of 82



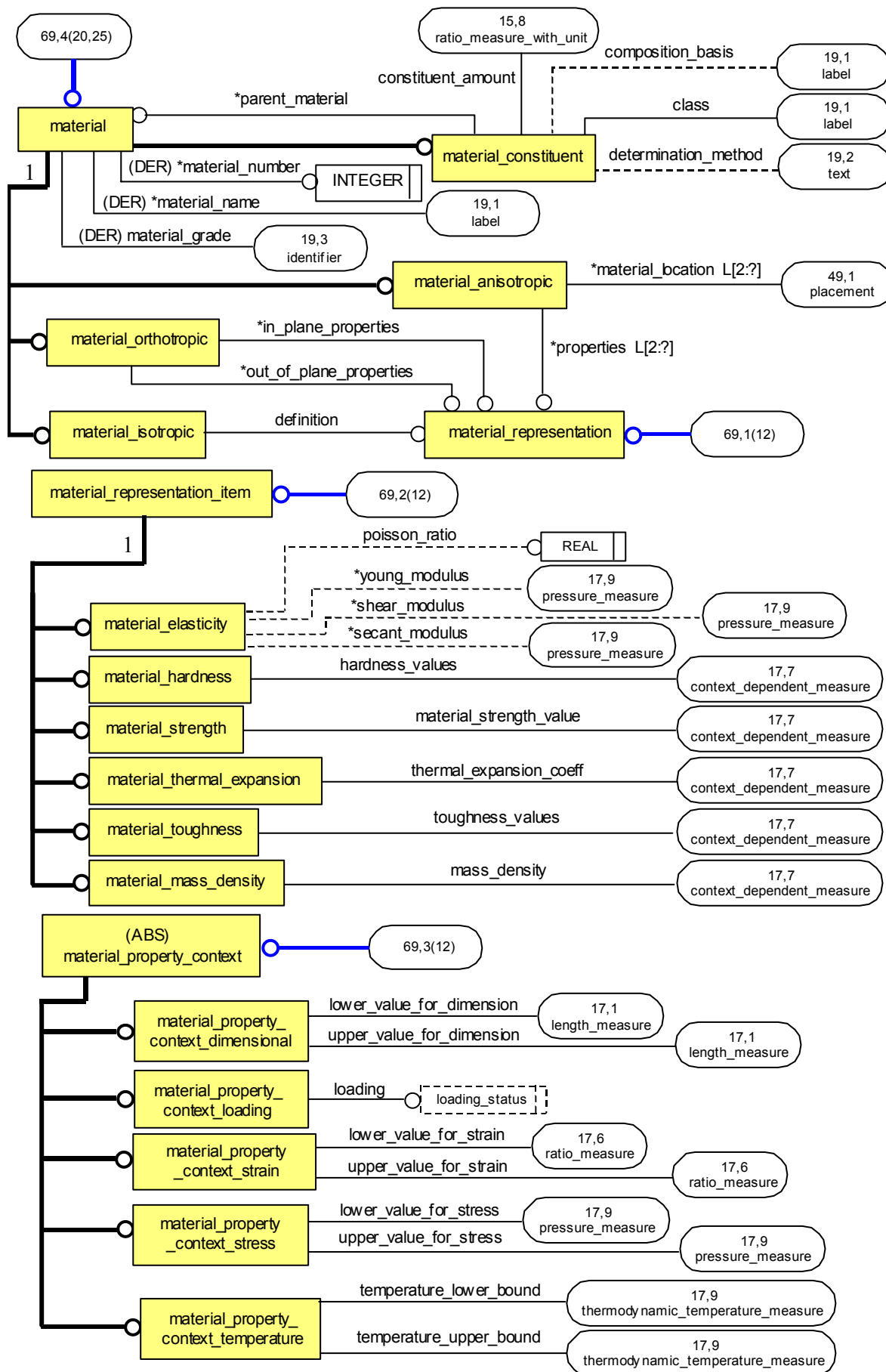
LPM/6 EXPRESS-G Diagram 67 of 82

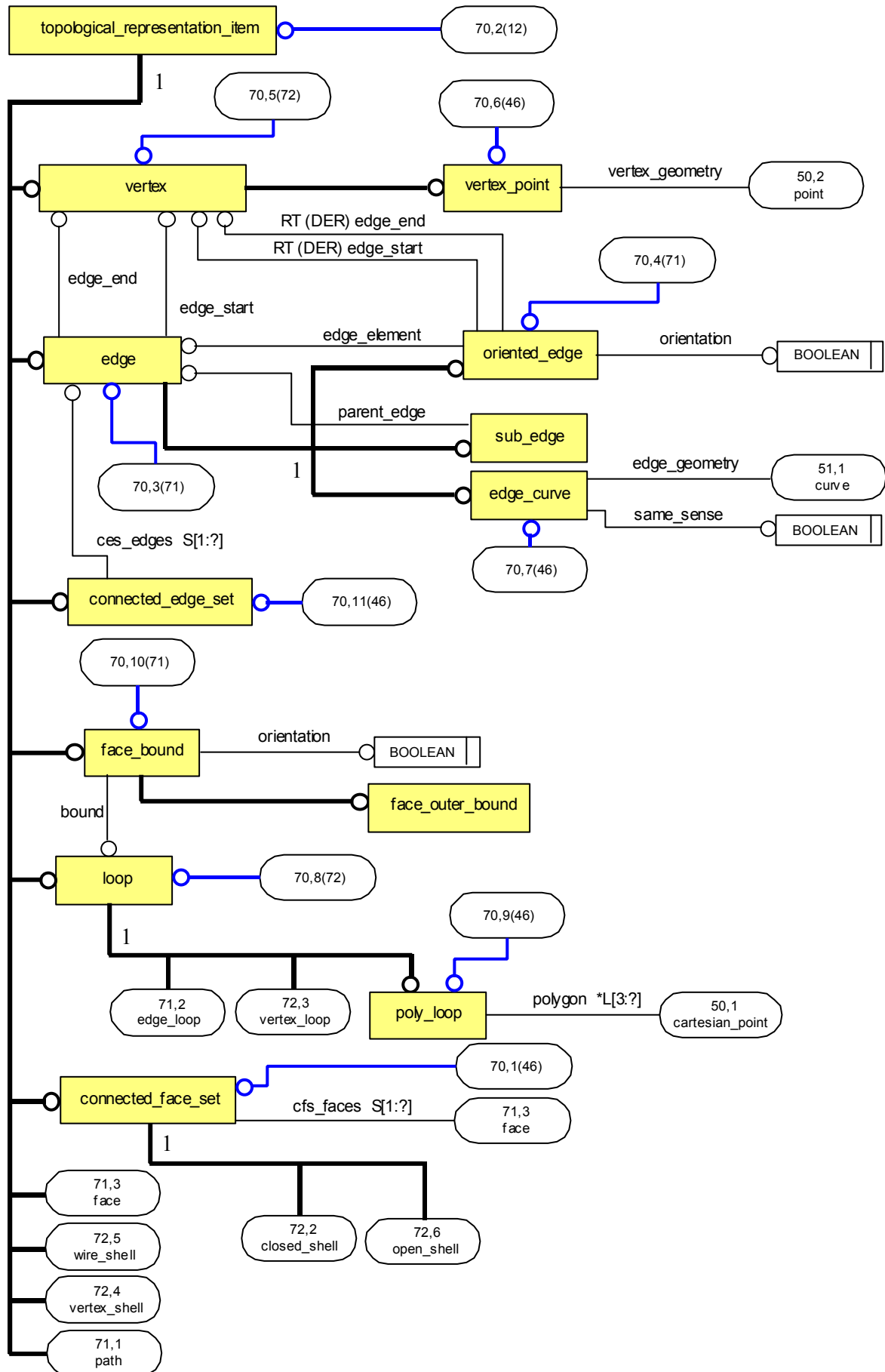


LPM/6 EXPRESS-G Diagram 68 of 82

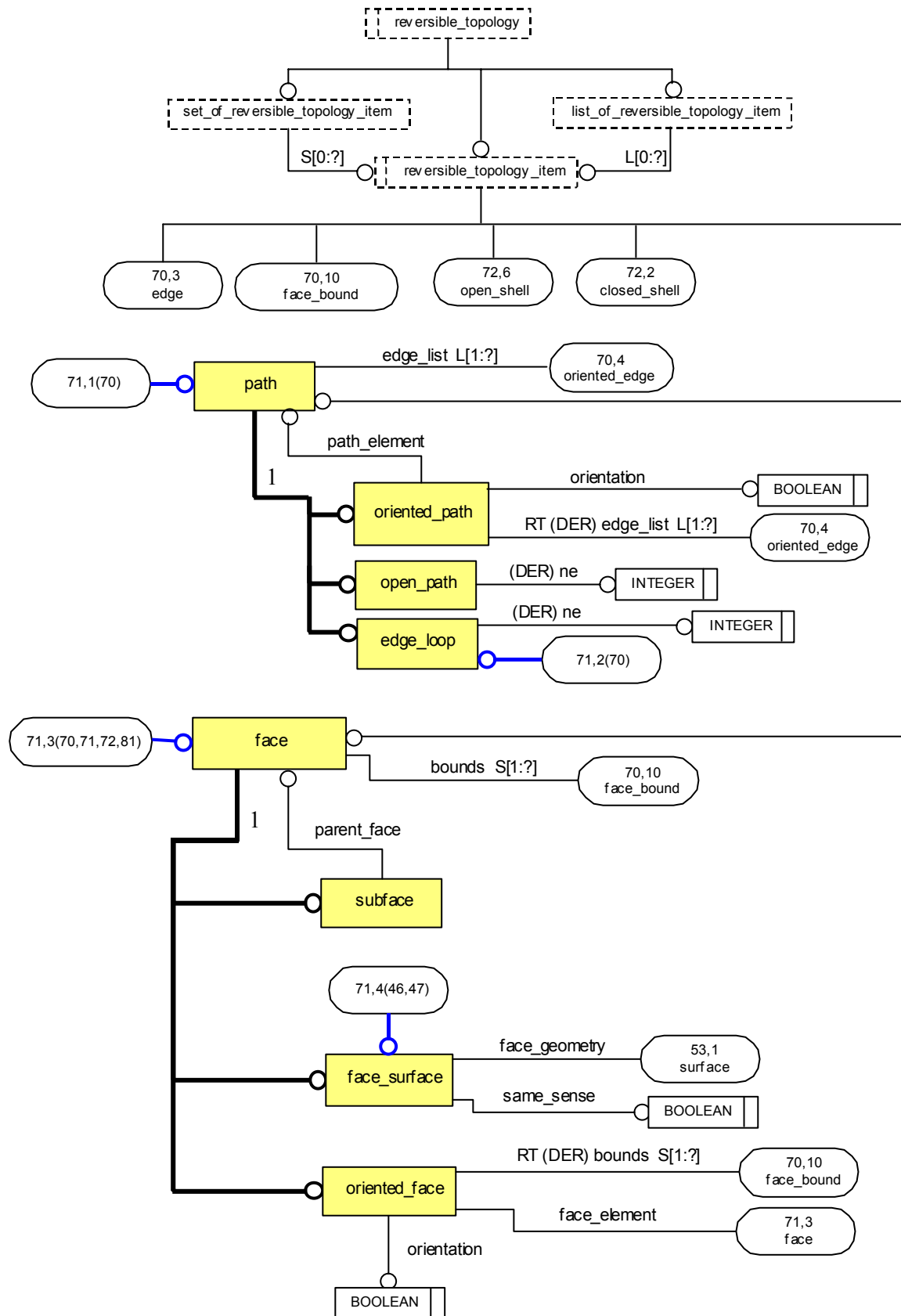


LPM/6 EXPRESS-G Diagram 69 of 82 (Modified for 2nd Edition)

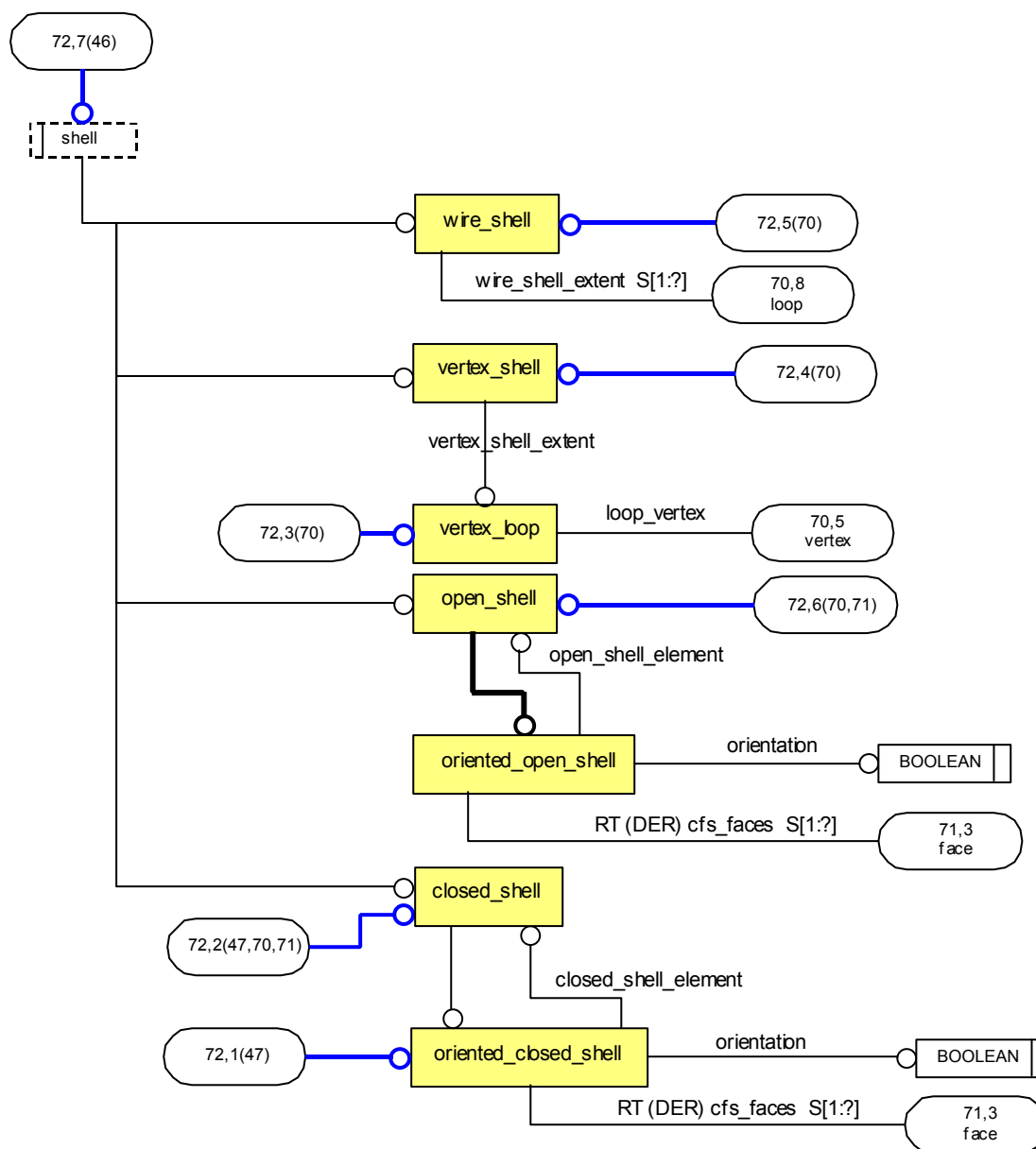


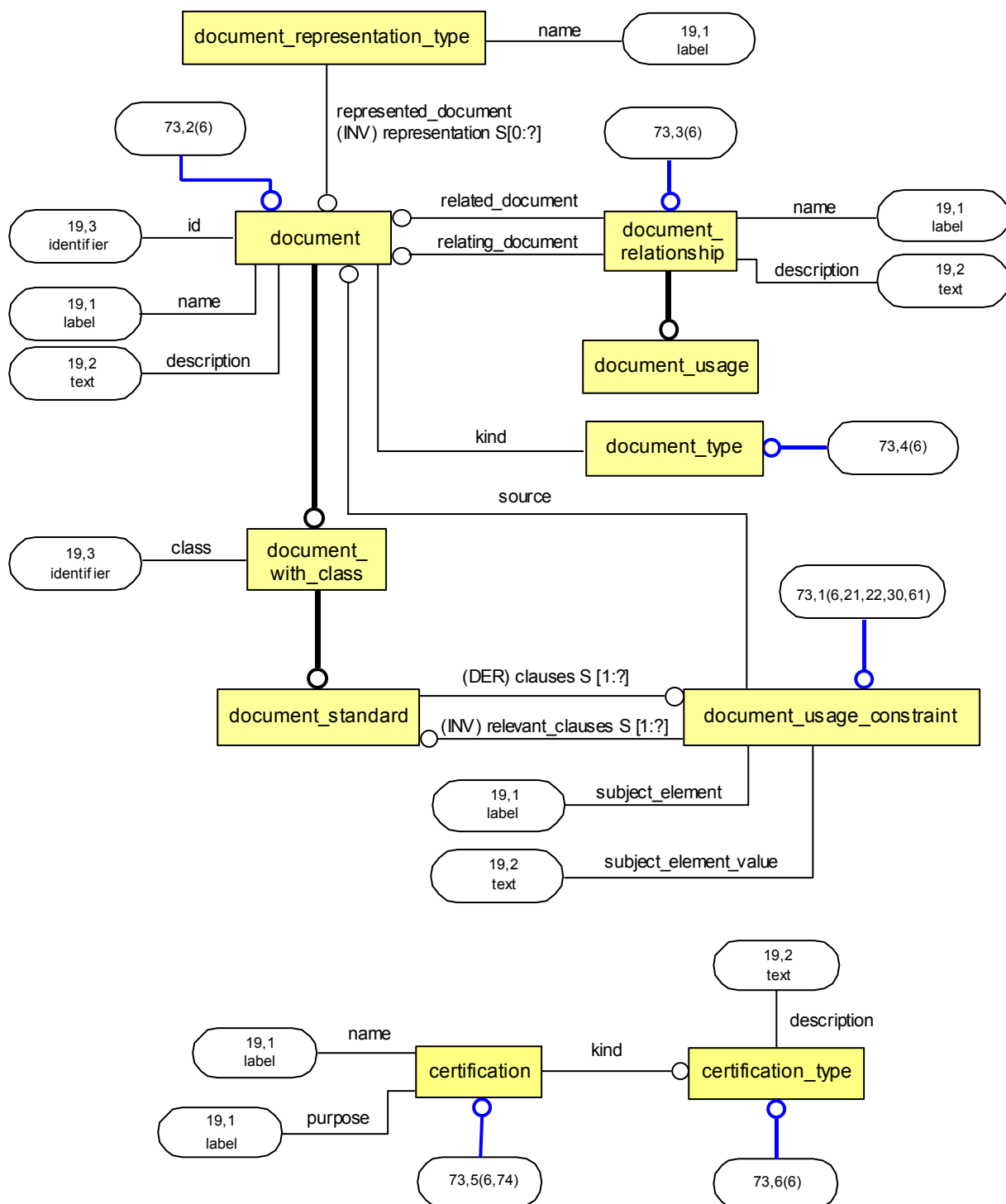
LPM/6 EXPRESS-G Diagram 70 of 82 (Modified for 2nd Edition)

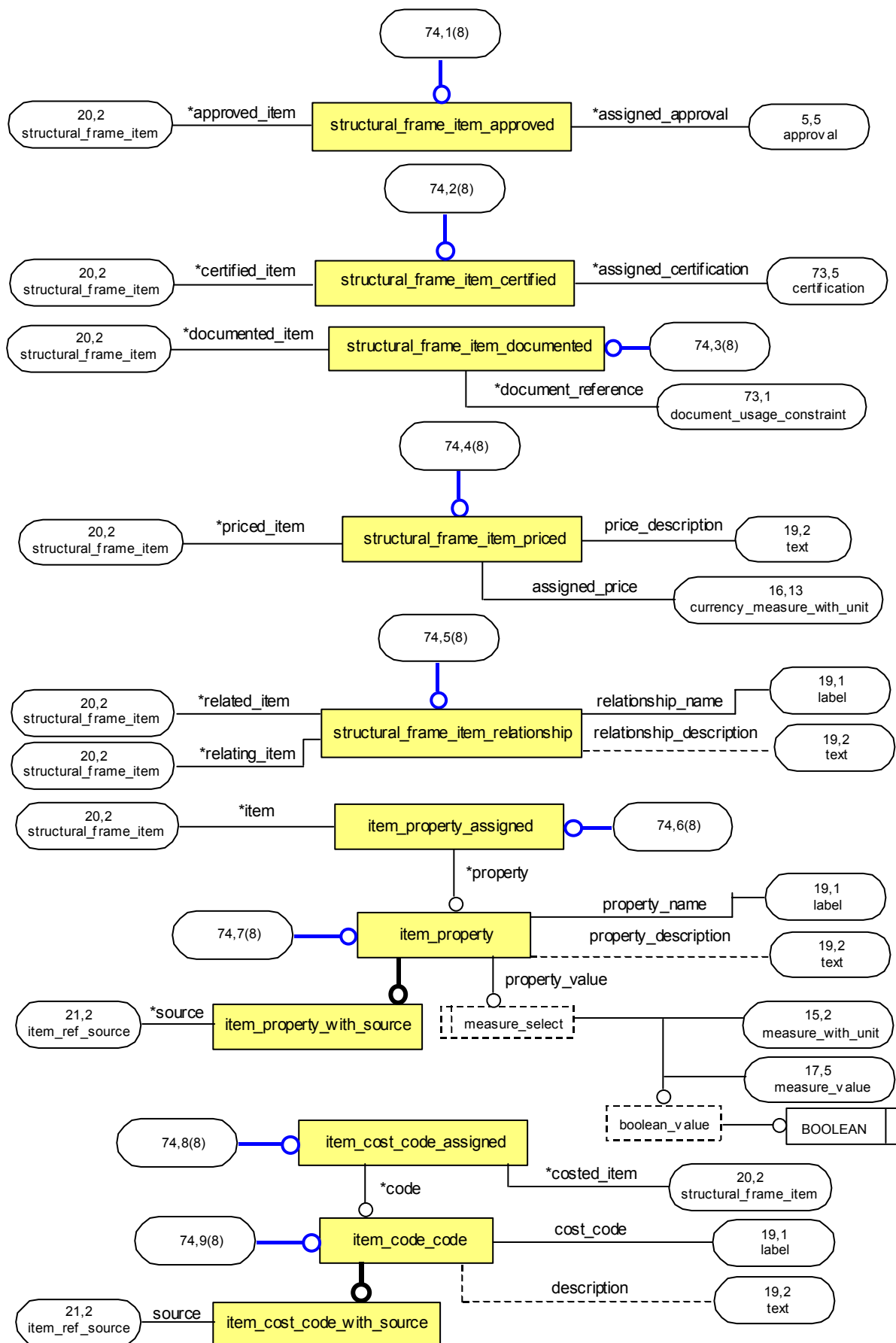
LPM/6 EXPRESS-G Diagram 71 of 82

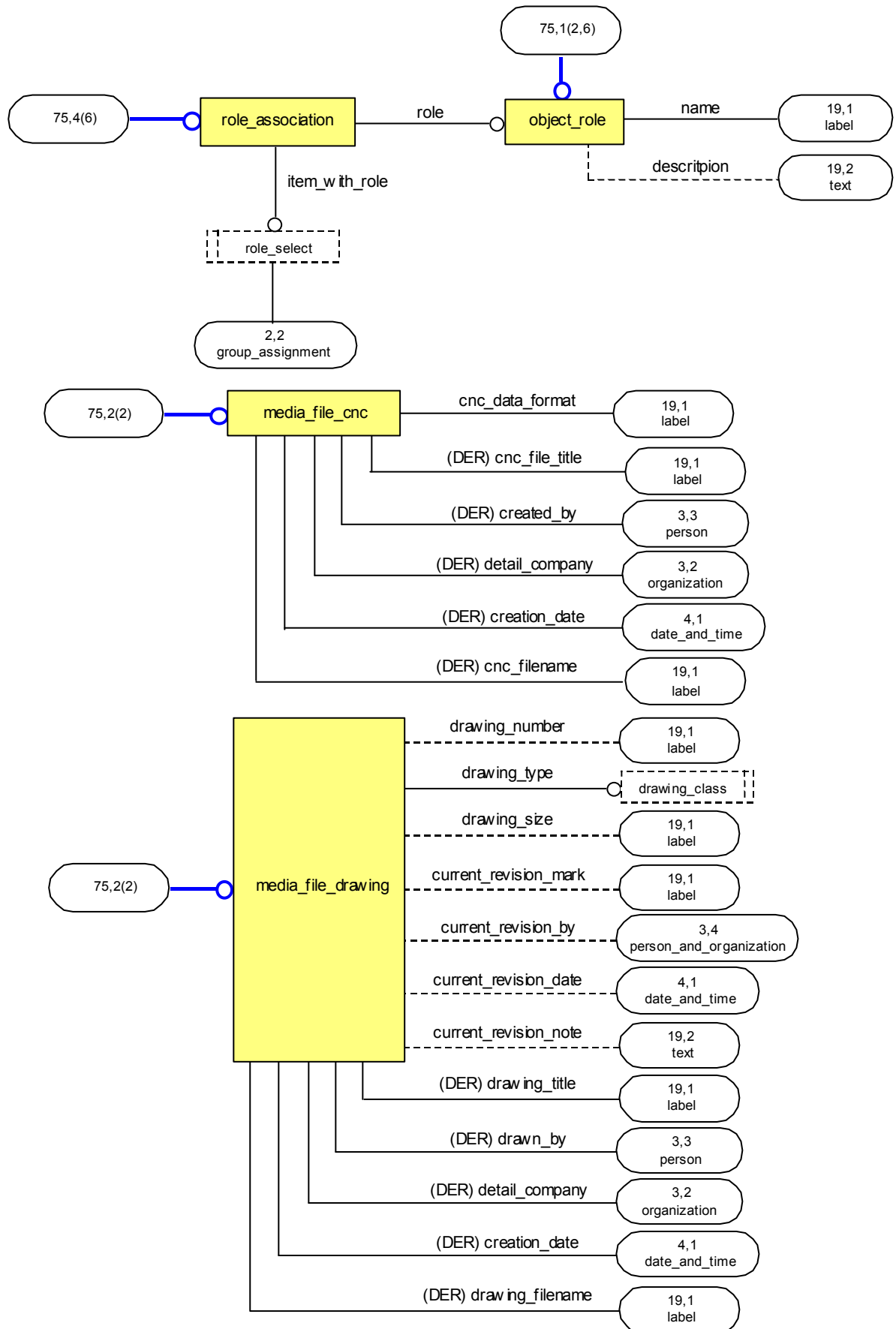


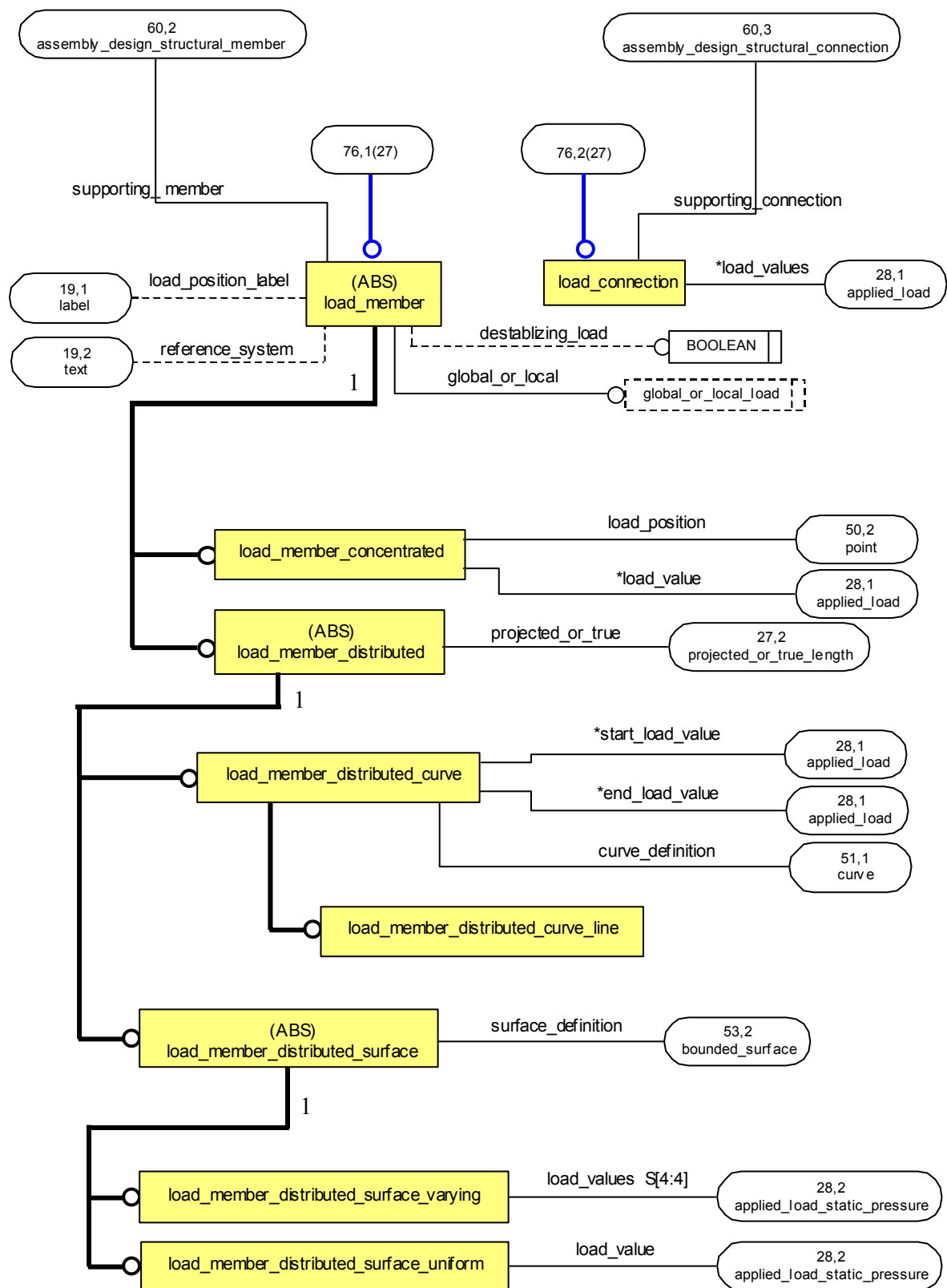
LPM/6 EXPRESS-G Diagram 72 of 82

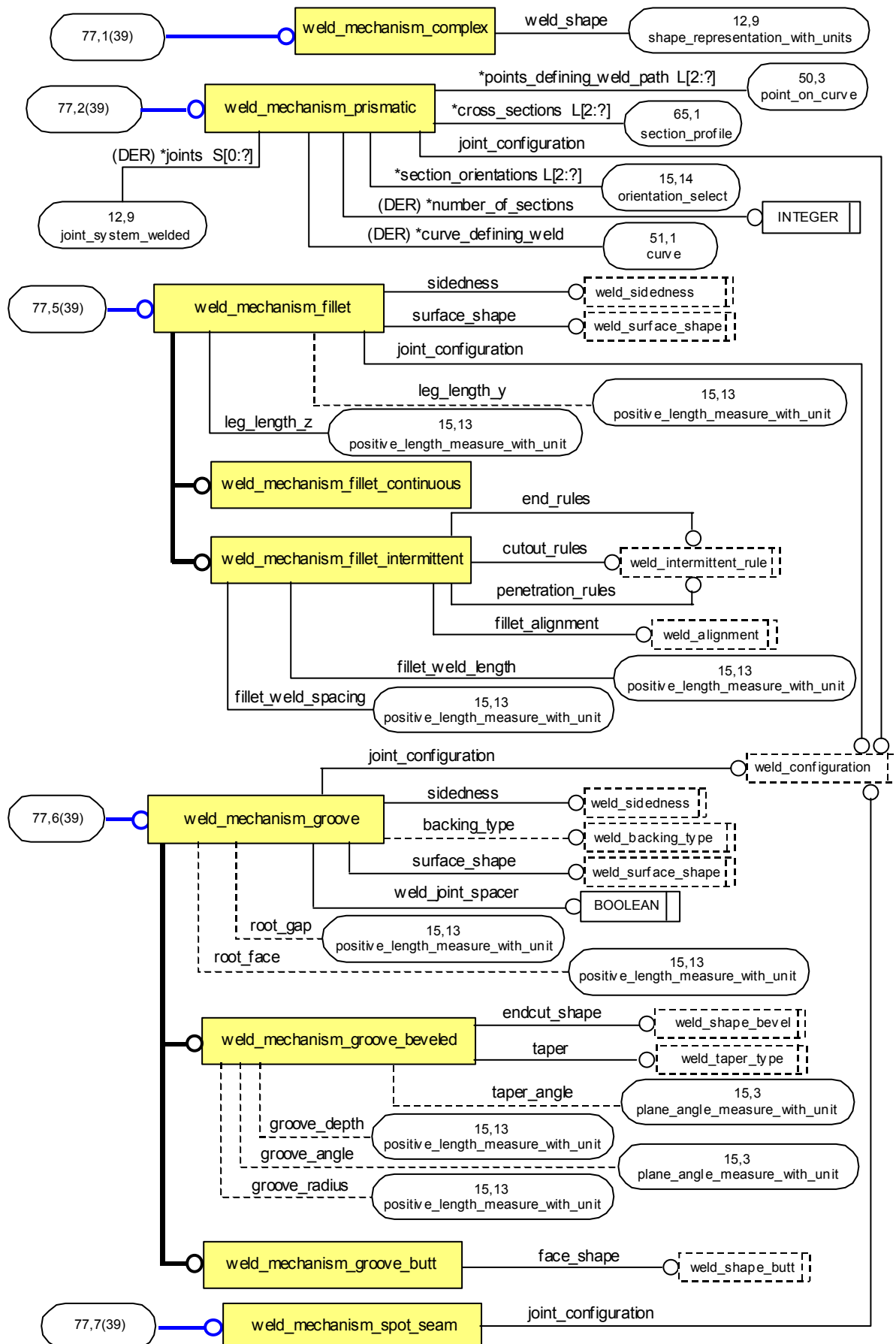


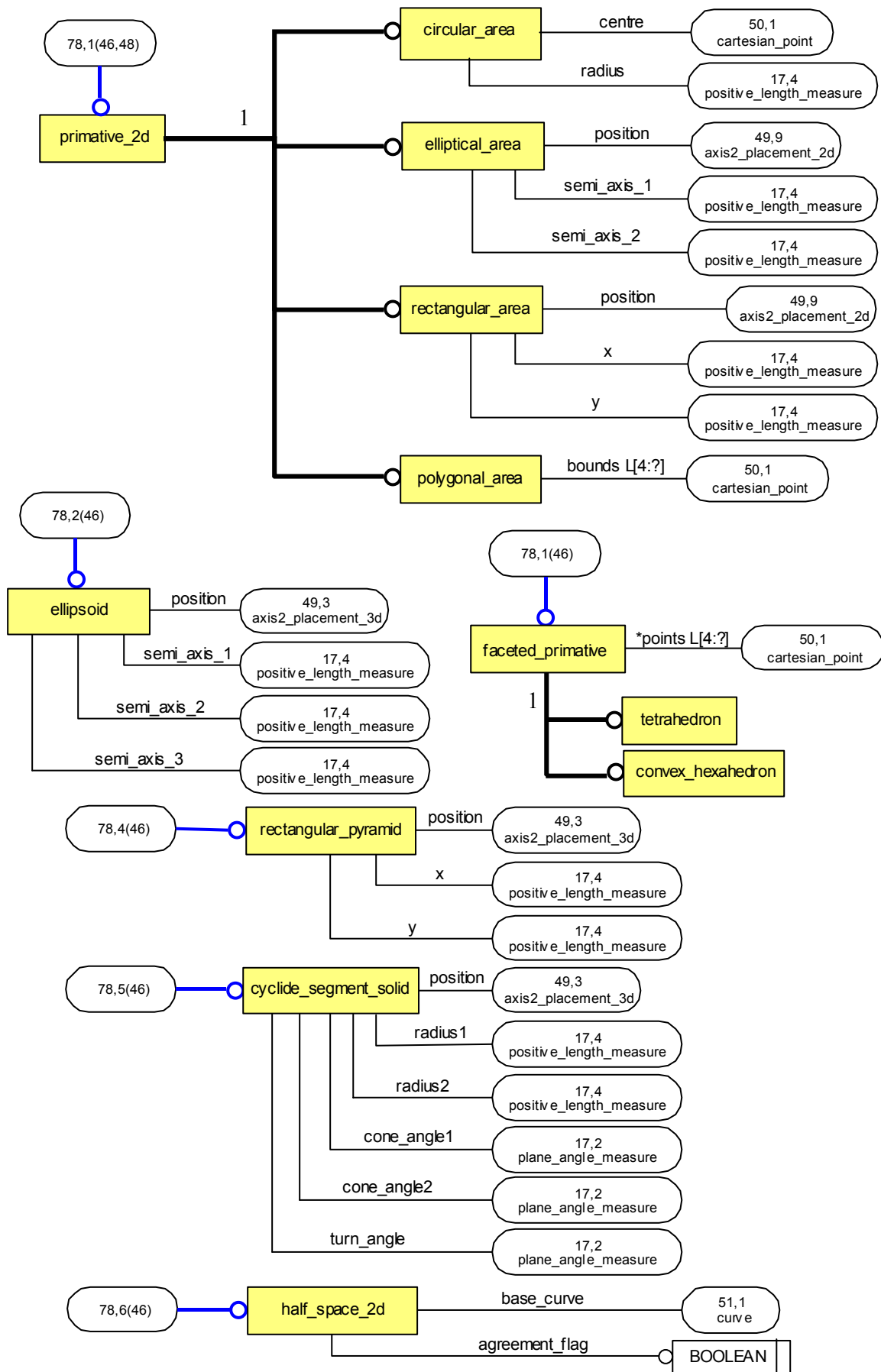
LPM/6 EXPRESS-G Diagram 73 of 82 (Modified for 2nd Edition)

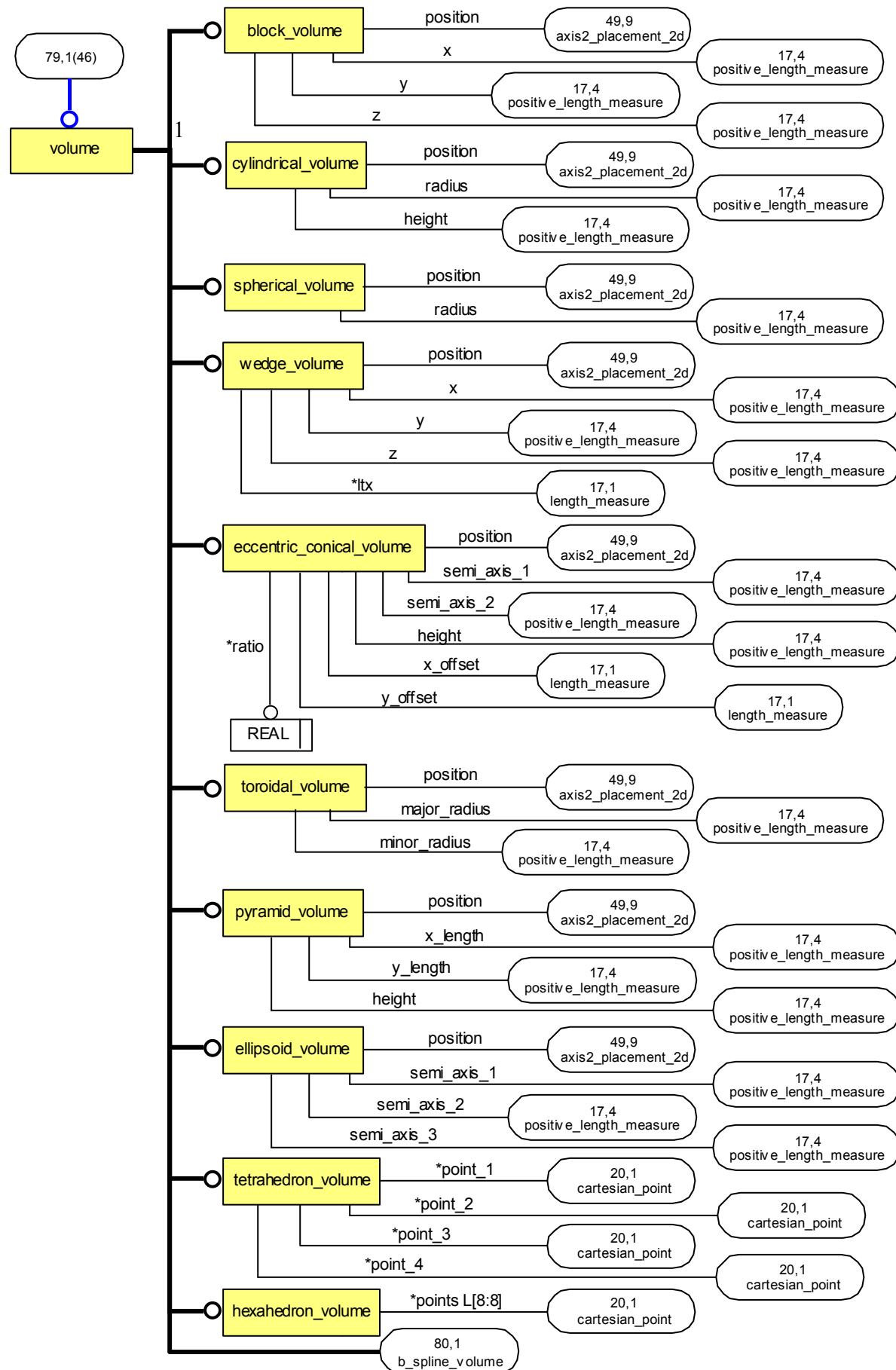
LPM/6 EXPRESS-G Diagram 74 of 82 (Modified for 2nd Edition)

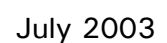
LPM/6 EXPRESS-G Diagram 75 of 82 (New for 2nd Edition)

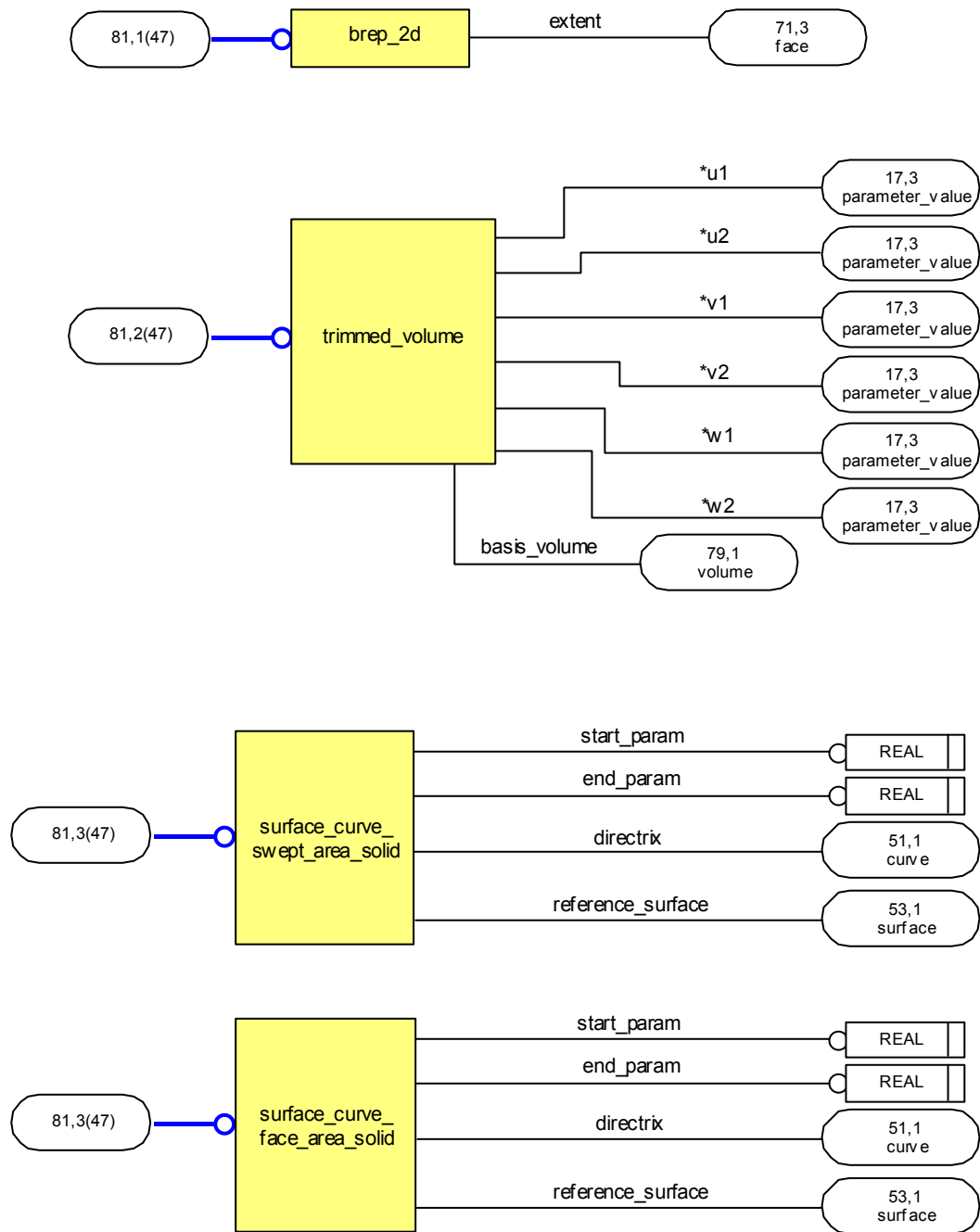
LPM/6 EXPRESS-G Diagram 76 of 82 (New for 2nd Edition)

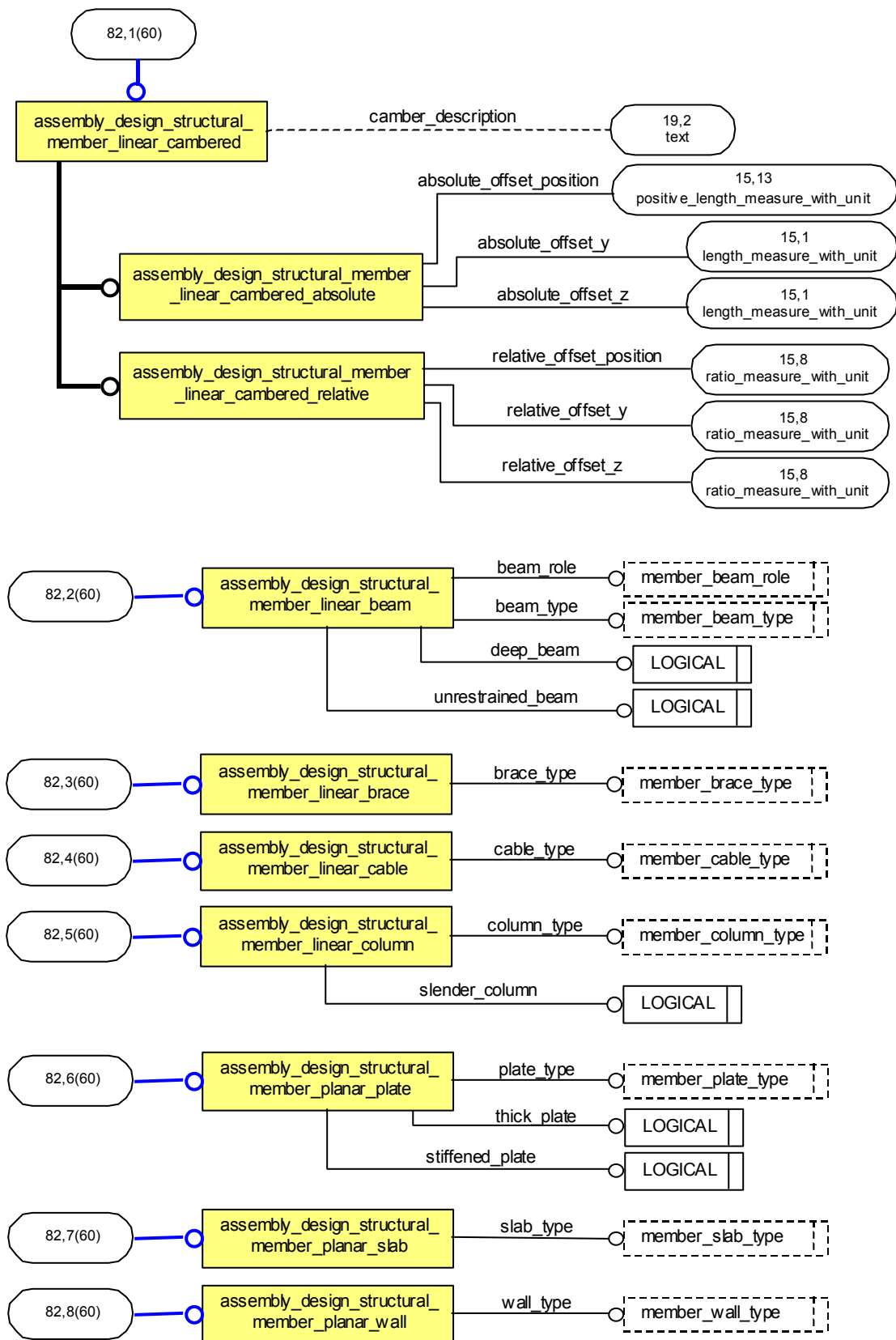
LPM/6 EXPRESS-G Diagram 77 of 82 (New for 2nd Edition)

LPM/6 EXPRESS-G Diagram 78 of 82 (New for 2nd Edition)

LPM/6 EXPRESS-G Diagram 79 of 82 (New for 2nd Edition)



LPM/6 EXPRESS-G Diagram 81 of 82 (New for 2nd Edition)

LPM/6 EXPRESS-G Diagram 82 of 82 (New for 2nd Edition)

APPENDIX C NEW CONSTRUCTS IN 2ND EDITION OF CIS/2

CONSTANT dummy_gri
 CONSTANT dummy_tri
 TYPE boolean_value
 TYPE brazing_type
 TYPE description_attribute_select
 TYPE drawing_class
 TYPE founded_item_select
 TYPE globally_unique_id
 TYPE id_attribute_select
 TYPE measure_select
 TYPE member_beam_role
 TYPE member_brace_type
 TYPE member_cable_type
 TYPE member_column_type
 TYPE member_plate_type
 TYPE member_slab_type
 TYPE member_wall_type
 TYPE name_attribute_select
 TYPE product_item_select
 TYPE role_select
 TYPE soldering_type
 TYPE weld_alignment
 TYPE weld_backing_type
 TYPE weld_configuration
 TYPE weld_intermittent_rule
 TYPE weld_shape_bevel
 TYPE weld_shape_butt
 TYPE weld_sidedness
 TYPE weld_surface_shape
 TYPE weld_taper_type
 TYPE welding_type_arc
 TYPE welding_type_beam
 TYPE welding_type_gas
 TYPE welding_type_other
 TYPE welding_type_pressure
 TYPE welding_type_resistance
 TYPE welding_type_stud
 ENTITY assembly_design_structural_member_linear_beam
 ENTITY assembly_design_structural_member_linear_brace
 ENTITY assembly_design_structural_member_linear_cable
 ENTITY assembly_design_structural_member_linear_campered
 ENTITY assembly_design_structural_member_linear_campered_absolute
 ENTITY assembly_design_structural_member_linear_campered_relative
 ENTITY assembly_design_structural_member_linear_column

ENTITY assembly_design_structural_member_planar_plate
 ENTITY assembly_design_structural_member_planar_slab
 ENTITY assembly_design_structural_member_planar_wall
 ENTITY assembly_with_bounding_box
 ENTITY b_spline_volume
 ENTITY b_spline_volume_with_knots
 ENTITY bezier_volume
 ENTITY block_volume
 ENTITY braze
 ENTITY brep_2d
 ENTITY circular_area
 ENTITY clothoid
 ENTITY convex_hexahedron
 ENTITY cyclide_segment_solid
 ENTITY cylindrical_volume
 ENTITY description_attribute
 ENTITY document_representation_type
 ENTITY eccentric_cone
 ENTITY eccentric_conical_volume
 ENTITY ellipsoid
 ENTITY ellipsoid_volume
 ENTITY elliptic_area
 ENTITY faceted_primitive
 ENTITY fastener_mechanism_with_position
 ENTITY feature_surface_point
 ENTITY feature_surface_point_mark
 ENTITY feature_volume_with_depth
 ENTITY feature_volume_with_limit
 ENTITY fixed_reference_swept_surface
 ENTITY founded_item
 ENTITY half_space_2d
 ENTITY hexahedron_volume
 ENTITY id_attribute
 ENTITY item_cost_code
 ENTITY item_cost_code_assigned
 ENTITY item_cost_code_with_source
 ENTITY item_property_with_source
 ENTITY joint_system_welded_with_shape
 ENTITY load_member
 ENTITY load_member_concentrated
 ENTITY load_member_distributed
 ENTITY load_member_distributed_curve
 ENTITY load_member_distributed_curve_line
 ENTITY load_member_distributed_surface
 ENTITY load_member_distributed_surface_uniform
 ENTITY load_member_distributed_surface_varying
 ENTITY located_assembly_marked
 ENTITY located_part_marked

ENTITY managed_data_item_with_history
 ENTITY media_content_drawing
 ENTITY media_file_cnc
 ENTITY media_file_drawing
 ENTITY name_attribute
 ENTITY object_role
 ENTITY oriented_surface
 ENTITY point_in_volume
 ENTITY polar_point
 ENTITY polygonal_area
 ENTITY primitive_2d
 ENTITY project_data_group
 ENTITY pyramid_volume
 ENTITY quasi_uniform_volume
 ENTITY rational_b_spline_volume
 ENTITY rectangular_area
 ENTITY rectangular_pyramid
 ENTITY role_association
 ENTITY solder
 ENTITY spherical_volume
 ENTITY subedge
 ENTITY surface_curve_swept_area_solid
 ENTITY surface_curve_swept_face_solid
 ENTITY surface_curve_swept_surface
 ENTITY tetrahedron
 ENTITY tetrahedron_volume
 ENTITY toroidal_volume
 ENTITY trimmed_volume
 ENTITY uniform_volume
 ENTITY volume
 ENTITY wedge_volume
 ENTITY weld_arc
 ENTITY weld_beam
 ENTITY weld_gas
 ENTITY weld_other
 ENTITY weld_pressure
 ENTITY weld_resistance
 ENTITY weld_stud
 ENTITY weld_mechanism_complex
 ENTITY weld_mechanism_fillet
 ENTITY weld_mechanism_fillet_continuous
 ENTITY weld_mechanism_fillet_intermittent
 ENTITY weld_mechanism_groove
 ENTITY weld_mechanism_groove_beveled
 ENTITY weld_mechanism_groove_butt
 ENTITY weld_mechanism_prismatic
 ENTITY weld_mechanism_spot_seam
 ENTITY zone_of_structure_sequence

ENTITY zone_of_structure_sequence_lot
FUNCTION above_plane
FUNCTION closed_shell_reversed
FUNCTION get_description_value
FUNCTION get_id_value
FUNCTION get_instance_id
FUNCTION get_item_cost_code
FUNCTION get_item_ref
FUNCTION get_name_value
FUNCTION get_role
FUNCTION make_array_of_array_of_array
FUNCTION open_shell_reversed
FUNCTION same_side
FUNCTION using_items

APPENDIX D MODIFIED CONSTRUCTS IN 2ND EDITION OF CIS/2

TYPE ahead_or_behind
TYPE assembly_component_select
TYPE boolean_operand
TYPE cardinal_point_ref
TYPE csg_primitive
TYPE element_volume_shape
TYPE member_linear_type
TYPE projected_or_true_length
TYPE select_generic_item
TYPE select_structural_item
TYPE weld_type
ENTITY action
ENTITY action_directive
ENTITY action_method
ENTITY address
ENTITY assembly
ENTITY assembly_design_structural_member
ENTITY assembly_design_structural_member_linear
ENTITY assembly_design_structural_member_planar
ENTITY axis1_placement
ENTITY axis2_placement_3d
ENTITY building
ENTITY cartesian_point
ENTITY composite_curve_segment
ENTITY coordinated_universal_time_offset
ENTITY curve
ENTITY curve_bounded_surface
ENTITY cylindrical_point
ENTITY derived_unit
ENTITY dispatch
ENTITY document
ENTITY document_relationship
ENTITY edge
ENTITY face_surface
ENTITY fastener_simple_bolt
ENTITY fastener_simple_nut
ENTITY fastener_simple_stud
ENTITY fastener_simple_washer
ENTITY feature
ENTITY feature_surface
ENTITY feature_volume
ENTITY geometric_representation_item
ENTITY group
ENTITY group_assignment

ENTITY group_relationship
 ENTITY item_property
 ENTITY item_ref_source_library
 ENTITY item_ref_source_proprietary
 ENTITY item_ref_source_standard
 ENTITY joint_system
 ENTITY joint_system_welded
 ENTITY joint_system_welded_linear
 ENTITY joint_system_welded_point
 ENTITY joint_system_welded_surface
 ENTITY load
 ENTITY load_case
 ENTITY load_element_distributed
 ENTITY loading_combination
 ENTITY located_assembly
 ENTITY located_assembly_child
 ENTITY managed_data_item
 ENTITY material
 ENTITY media_file
 ENTITY organization_relationship
 ENTITY oriented_path
 ENTITY part
 ENTITY part_prismatic_simple_cambered_relative
 ENTITY path
 ENTITY person_and_organization
 ENTITY person_and_organization_role
 ENTITY point
 ENTITY poly_loop
 ENTITY procure
 ENTITY project_process_item
 ENTITY rectangular_composite_surface
 ENTITY representation
 ENTITY revolved_area_solid
 ENTITY revolved_face_solid
 ENTITY section_profile
 ENTITY solid_model
 ENTITY solid_replica
 ENTITY spherical_point
 ENTITY step_file
 ENTITY structural_frame_item
 ENTITY structural_frame_process
 ENTITY surface_of_revolution
 ENTITY surface_replica
 ENTITY swept_area_solid
 ENTITY swept_face_solid
 ENTITY swept_surface
 ENTITY uncertainty_measure_with_unit
 ENTITY versioned_action_request

ENTITY weld
ENTITY weld_mechanism
FUNCTION acyclic_document_relationship
FUNCTION acyclic_group_relationship
FUNCTION acyclic_organization_relationship
FUNCTION bag_to_set
FUNCTION base_axis
FUNCTION boolean_choose
FUNCTION build_2axes
FUNCTION build_axes
FUNCTION build_transformed_set
FUNCTION cross_product
FUNCTION dimension_of
FUNCTION edge_reversed
FUNCTION face_bound_reversed
FUNCTION face_reversed
FUNCTION first_proj_axis
FUNCTION make_array_of_array
FUNCTION mixed_loop_type_set
FUNCTION normalise
FUNCTION orthogonal_complement
FUNCTION path_reversed
FUNCTION scalar_times_vector
FUNCTION second_proj_axis
FUNCTION shell_reversed

This page is blank

INDEX

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
above_plane	FUNCTION	11	857	N/A	N/A	New
action	ENTITY	4	707	5	903	Modified
action_directive	ENTITY	4	707	5	903	Modified
action_method	ENTITY	4	707	5	903	Modified
action_source_accidental	TYPE	173	677	26	924	Unchanged
action_source_permanent	TYPE	174	677	26	924	Unchanged
action_source_variable_long_term	TYPE	174	677	26	924	Unchanged
action_source_variable_short_term	TYPE	175	678	26	924	Unchanged
action_source_variable_transient	TYPE	175	678	26	924	Unchanged
acyclic_curve_replica	FUNCTION	11	858	N/A	N/A	Unchanged
acyclic_document_relationship	FUNCTION	10	858	N/A	N/A	Modified
acyclic_group_relationship	FUNCTION	10	858	N/A	N/A	Modified
acyclic_mapped_representation	FUNCTION	12	859	N/A	N/A	Unchanged
acyclic_organization_relationship	FUNCTION	10	861	N/A	N/A	Modified
acyclic_point_replica	FUNCTION	11	861	N/A	N/A	Unchanged
acyclic_set_replica	FUNCTION	11	861	N/A	N/A	Unchanged
acyclic_solid_replica	FUNCTION	11	862	N/A	N/A	Unchanged
acyclic_surface_replica	FUNCTION	11	862	N/A	N/A	Unchanged
address	ENTITY	5	708	3	901	Modified
ahead_or_behind	TYPE	2	678	4	902	Modified
analysis_method	ENTITY	108	708	22	920	Unchanged
analysis_method_documented	ENTITY	109	708	22	920	Unchanged
analysis_method_dynamic	ENTITY	109	709	22	920	Unchanged
analysis_method_pseudo_dynamic	ENTITY	110	709	22	920	Unchanged
analysis_method_static	ENTITY	110	709	22	920	Unchanged
analysis_model	ENTITY	111	709	22	920	Unchanged
analysis_model_2D	ENTITY	113	709	22	920	Unchanged
analysis_model_3D	ENTITY	114	709	22	920	Unchanged
analysis_model_child	ENTITY	115	710	22	920	Unchanged
analysis_model_located	ENTITY	116	710	22	920	Unchanged
analysis_model_mapping	ENTITY	117	710	22	920	Unchanged
analysis_model_relationship	ENTITY	117	710	22	920	Unchanged
analysis_result	ENTITY	224	710	31	929	Unchanged
analysis_result_element	ENTITY	225	711	31	929	Unchanged
analysis_result_element_curve	ENTITY	225	711	31	929	Unchanged
analysis_result_element_node	ENTITY	227	711	31	929	Unchanged
analysis_result_element_point	ENTITY	227	711	31	929	Unchanged
analysis_result_element_surface	ENTITY	228	711	31	929	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
analysis_result_element_surface_stresses	ENTITY	229	711	32	930	Unchanged
analysis_result_element_surface_tractions	ENTITY	230	712	32	930	Unchanged
analysis_result_element_volume	ENTITY	232	712	31	929	Unchanged
analysis_result_element_volume_stress_tensor	ENTITY	234	712	32	930	Unchanged
analysis_result_node	ENTITY	235	712	31	929	Unchanged
analysis_results_set	ENTITY	236	712	35	933	Unchanged
analysis_results_set_basic	ENTITY	237	713	35	933	Unchanged
analysis_results_set_combined	ENTITY	237	713	35	933	Unchanged
analysis_results_set_envelope	ENTITY	238	713	35	933	Unchanged
analysis_results_set_redistributed	ENTITY	238	713	35	933	Unchanged
applied_load	ENTITY	178	713	28	926	Unchanged
applied_load_dynamic	ENTITY	179	713	28	926	Unchanged
applied_load_dynamic_acceleration	ENTITY	181	714	29	927	Unchanged
applied_load_dynamic_velocity	ENTITY	182	714	29	927	Unchanged
applied_load_static	ENTITY	184	714	28	926	Unchanged
applied_load_static_displacement	ENTITY	185	715	28	926	Unchanged
applied_load_static_force	ENTITY	186	715	28	926	Unchanged
applied_load_static_pressure	ENTITY	188	715	28	926	Unchanged
approval	ENTITY	4	715	5	903	Unchanged
approval_status	ENTITY	4	716	5	903	Unchanged
area_measure	TYPE	2	678	17	915	Unchanged
area_measure_with_unit	ENTITY	5	716	15	913	Unchanged
area_unit	ENTITY	5	716	14	912	Unchanged
assemble	ENTITY	628	716	59	957	Unchanged
assembly	ENTITY	23	716	59	957	Modified
assembly_component_select	TYPE	618	678	56	954	Modified
assembly_design	ENTITY	24	717	60	958	Unchanged
assembly_design_child	ENTITY	26	717	60	958	Unchanged
assembly_design_structural_connection	ENTITY	27	717	60	958	Unchanged
assembly_design_structural_connection_external	ENTITY	28	717	60	958	Unchanged
assembly_design_structural_connection_internal	ENTITY	28	717	60	958	Unchanged
assembly_design_structural_frame	ENTITY	29	718	60	958	Unchanged
assembly_design_structural_member	ENTITY	31	718	60	958	Modified
assembly_design_structural_member_cubic	ENTITY	32	718	60	958	Unchanged
assembly_design_structural_member_linear	ENTITY	33	718	60	958	Modified
assembly_design_structural_member_linear_beam	ENTITY	34	719	82	980	New
assembly_design_structural_member_linear_brace	ENTITY	35	719	82	980	New
assembly_design_structural_member_linear_cable	ENTITY	36	719	82	980	New
assembly_design_structural_member_linear_campered	ENTITY	37	719	82	980	New
assembly_design_structural_member_linear_campered_absolute	ENTITY	37	719	82	980	New

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
assembly_design_structural_member_linear_campered_relative	ENTITY	38	720	82	980	New
assembly_design_structural_member_linear_column	ENTITY	40	720	82	980	New
assembly_design_structural_member_planar	ENTITY	41	720	60	958	Modified
assembly_design_structural_member_planar_plate	ENTITY	42	720	82	980	New
assembly_design_structural_member_planar_slab	ENTITY	43	720	82	980	New
assembly_design_structural_member_planar_wall	ENTITY	43	721	82	980	New
assembly_manufacturing	ENTITY	44	721	59	957	Unchanged
assembly_manufacturing_child	ENTITY	45	721	59	957	Unchanged
assembly_map	ENTITY	45	721	59	957	Unchanged
assembly_relationship	ENTITY	46	721	59	957	Unchanged
assembly_with_bounding_box	ENTITY	47	722	59	957	New
assembly_with_shape	ENTITY	48	721	59	957	Unchanged
associated_surface	FUNCTION	11	862	N/A	N/A	Unchanged
axis1_placement	ENTITY	7	722	49	947	Modified
axis2_placement	TYPE	3	678	49	947	Unchanged
axis2_placement_2d	ENTITY	7	722	49	947	Unchanged
axis2_placement_3d	ENTITY	7	722	49	947	Modified
b_spline_curve	ENTITY	7	723	55	953	Unchanged
b_spline_curve_form	TYPE	3	678	55	953	Unchanged
b_spline_curve_with_knots	ENTITY	7	723	55	953	Unchanged
b_spline_surface	ENTITY	7	723	54	952	Unchanged
b_spline_surface_form	TYPE	3	679	54	952	Unchanged
b_spline_surface_with_knots	ENTITY	7	724	54	952	Unchanged
b_spline_volume	ENTITY	7	724	80	978	New
b_spline_volume_with_knots	ENTITY	7	725	80	978	New
bag_to_set	FUNCTION	10	863	N/A	N/A	Modified
base_axis	FUNCTION	11	863	N/A	N/A	Modified
bend	ENTITY	630	726	56	954	Unchanged
bending_method	TYPE	619	679	56	954	Unchanged
bezier_curve	ENTITY	7	726	55	953	Unchanged
bezier_surface	ENTITY	7	726	54	952	Unchanged
bezier_volume	ENTITY	7	726	80	978	New
block	ENTITY	6	726	48	946	Unchanged
block_volume	ENTITY	7	726	79	977	New
boolean_choose	FUNCTION	11	864	N/A	N/A	Modified
boolean_operand	TYPE	3	679	48	946	Modified
boolean_operator	TYPE	3	679	48	946	Unchanged
boolean_result	ENTITY	6	726	48	946	Unchanged
boolean_value	TYPE	582	679	74	972	New
boundary_condition	ENTITY	119	727	23	921	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
boundary_condition_logical	ENTITY	120	727	23	921	Unchanged
boundary_condition_skewed	ENTITY	122	727	23	921	Unchanged
boundary_condition_spring_linear	ENTITY	123	727	23	921	Unchanged
boundary_condition_spring_non_linear	ENTITY	126	728	23	921	Unchanged
boundary_condition_warping	ENTITY	128	728	23	921	Unchanged
boundary_curve	ENTITY	7	728	52	950	Unchanged
bounded_curve	ENTITY	7	728	52	950	Unchanged
bounded_pcurve	ENTITY	7	728	52	950	Unchanged
bounded_surface	ENTITY	7	728	53	951	Unchanged
bounded_surface_curve	ENTITY	7	729	52	950	Unchanged
box_domain	ENTITY	6	729	48	946	Unchanged
boxed_half_space	ENTITY	6	729	48	946	Unchanged
braze	ENTITY	630	729	56	954	New
brazing_type	TYPE	619	680	56	954	New
brep_2d	ENTITY	6	729	81	979	New
brep_with_voids	ENTITY	6	730	47	945	Unchanged
buckling_direction	TYPE	14	680	61	959	Unchanged
build_2axes	FUNCTION	11	864	N/A	N/A	Modified
build_axes	FUNCTION	11	864	N/A	N/A	Modified
build_transformed_set	FUNCTION	11	865	N/A	N/A	Modified
building	ENTITY	64	730	9	907	Modified
building_complex	ENTITY	65	730	9	907	Unchanged
building_with_shape	ENTITY	67	730	9	907	Unchanged
calendar_date	ENTITY	4	730	4	902	Unchanged
cardinal_point_ref	TYPE	493	680	65	963	Modified
cartesian_point	ENTITY	7	731	50	948	Modified
cartesian_transformation_operator	ENTITY	7	731	49	947	Unchanged
cartesian_transformation_operator_2d	ENTITY	7	731	49	947	Unchanged
cartesian_transformation_operator_3d	ENTITY	7	731	49	947	Unchanged
castellation_type	TYPE	260	680	64	962	Unchanged
certification	ENTITY	4	731	73	971	Unchanged
certification_type	ENTITY	4	732	73	971	Unchanged
chemical_mechanism	ENTITY	343	732	39	937	Unchanged
chemical_mechanism_type	TYPE	330	680	39	937	Unchanged
circle	ENTITY	7	732	51	949	Unchanged
circular_area	ENTITY	6	732	78	976	New
cleaning_method	TYPE	619	680	57	955	Unchanged
closed_shell	ENTITY	9	732	72	970	Unchanged
closed_shell_reversed	FUNCTION	11	865	N/A	N/A	New
clothoid	ENTITY	7	732	51	949	New

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
coating	ENTITY	614	732	20	918	Unchanged
coating_method	TYPE	620	680	57	955	Unchanged
coating_purpose	TYPE	614	680	20	918	Unchanged
compatible_dimension	RULE	12	856	N/A	N/A	Unchanged
complexity_level	TYPE	15	681	59	957	Unchanged
composite_curve	ENTITY	7	732	52	950	Unchanged
composite_curve_on_surface	ENTITY	7	733	52	950	Unchanged
composite_curve_segment	ENTITY	7	733	52	950	Modified
conditional_reverse	FUNCTION	11	866	N/A	N/A	Unchanged
conic	ENTITY	7	733	51	949	Unchanged
conical_surface	ENTITY	7	733	53	951	Unchanged
connected_edge_set	ENTITY	9	734	70	968	Unchanged
connected_face_set	ENTITY	9	734	70	968	Unchanged
connection_type	TYPE	15	681	60	958	Unchanged
constraints_composite_curve_on_surface	FUNCTION	11	866	N/A	N/A	Unchanged
constraints_geometry_shell_based_surface_model	FUNCTION	11	866	N/A	N/A	Unchanged
constraints_geometry_shell_based_wireframe_model	FUNCTION	11	867	N/A	N/A	Unchanged
constraints_param_b_spline	FUNCTION	11	867	N/A	N/A	Unchanged
constraints_rectangular_composite_surface	FUNCTION	11	868	N/A	N/A	Unchanged
context_dependent_measure	TYPE	2	681	17	915	Unchanged
context_dependent_unit	ENTITY	5	734	14	912	Unchanged
contract	ENTITY	4	734	57	955	Unchanged
contract_type	ENTITY	4	734	57	955	Unchanged
conversion_based_unit	ENTITY	5	734	14	912	Unchanged
convex_hexahedron	ENTITY	6	734	78	976	New
coord_system	ENTITY	426	735	50	948	Unchanged
coord_system_cartesian_2d	ENTITY	428	735	50	948	Unchanged
coord_system_cartesian_3d	ENTITY	430	735	50	948	Unchanged
coord_system_child	ENTITY	435	736	50	948	Unchanged
coord_system_cylindrical	ENTITY	437	736	50	948	Unchanged
coord_system_spherical	ENTITY	438	736	50	948	Unchanged
coordinated_universal_time_offset	ENTITY	4	736	4	902	Modified
count_measure	TYPE	2	681	17	915	Unchanged
cross_product	FUNCTION	11	869	N/A	N/A	Modified
csg_primitive	TYPE	3	681	48	946	Modified
csg_select	TYPE	3	681	47	945	Unchanged
csg_solid	ENTITY	6	736	47	945	Unchanged
currency_measure_with_unit	ENTITY	549	736	16	914	Unchanged
currency_rate_with_unit	ENTITY	550	736	16	914	Unchanged
currency_unit	ENTITY	550	737	16	914	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
curve	ENTITY	7	737	51	949	Modified
curve_bounded_surface	ENTITY	7	737	53	951	Modified
curve_on_surface	TYPE	3	682	52	950	Unchanged
curve_replica	ENTITY	7	737	51	949	Unchanged
curve_weights_positive	FUNCTION	11	870	N/A	N/A	Unchanged
cut	ENTITY	631	738	44	942	Unchanged
cutting_type	TYPE	620	682	42	940	Unchanged
cyclide_segment_solid	ENTITY	6	738	78	976	New
cylindrical_point	ENTITY	7	738	50	948	Modified
cylindrical_surface	ENTITY	7	738	53	951	Unchanged
cylindrical_volume	ENTITY	7	738	79	977	New
data_status_type	TYPE	654	682	6	904	Unchanged
date	ENTITY	4	739	4	902	Unchanged
date_and_time	ENTITY	4	739	4	902	Unchanged
day_in_month_number	TYPE	2	682	4	902	Unchanged
definitional_representation	ENTITY	9	739	12	910	Unchanged
degenerate_pcurve	ENTITY	7	739	50	948	Unchanged
degenerate_toroidal_surface	ENTITY	7	739	53	951	Unchanged
degrees_rotation	TYPE	62	682	11	909	Unchanged
derive_dimensional_exponents	FUNCTION	10	870	N/A	N/A	Unchanged
derived_measure	TYPE	544	682	17	915	Unchanged
derived_measure_with_unit	ENTITY	551	739	16	914	Unchanged
derived_unit	ENTITY	5	740	14	912	Modified
derived_unit_element	ENTITY	5	740	14	912	Unchanged
description_attribute	ENTITY	4	740	19	917	New
description_attribute_select	TYPE	2	683	19	917	New
descriptive_measure	TYPE	2	683	17	915	Unchanged
design_criterion	ENTITY	49	740	61	959	Unchanged
design_criterion_documented	ENTITY	50	741	61	959	Unchanged
design_joint_system	ENTITY	50	741	38	936	Unchanged
design_part	ENTITY	52	741	38	936	Unchanged
design_result	ENTITY	239	741	36	934	Unchanged
design_result_connection	ENTITY	240	741	36	934	Unchanged
design_result_joint_system	ENTITY	241	742	36	934	Unchanged
design_result_mapped	ENTITY	242	742	36	934	Unchanged
design_result_member	ENTITY	242	742	36	934	Unchanged
design_result_part	ENTITY	243	742	36	934	Unchanged
design_result_resolved	ENTITY	244	742	36	934	Unchanged
dimension_count	TYPE	3	683	12	910	Unchanged
dimension_of	FUNCTION	11	871	N/A	N/A	Modified

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
dimensional_exponents	ENTITY	5	742	14	912	Unchanged
dimensions_for_si_unit	FUNCTION	10	872	N/A	N/A	Unchanged
direct_or_indirect_action	TYPE	176	683	26	924	Unchanged
directed_action	ENTITY	4	742	5	903	Unchanged
direction	ENTITY	7	743	49	947	Unchanged
dispatch	ENTITY	632	743	56	954	Modified
document	ENTITY	4	743	73	971	Modified
document_relationship	ENTITY	4	743	73	971	Modified
document_representation_type	ENTITY	4	743	73	971	New
document_standard	ENTITY	583	743	73	971	Unchanged
document_type	ENTITY	4	744	73	971	Unchanged
document_usage	ENTITY	584	744	73	971	Unchanged
document_usage_constraint	ENTITY	4	744	73	971	Unchanged
document_with_class	ENTITY	4	744	73	971	Unchanged
dot_product	FUNCTION	11	873	N/A	N/A	Unchanged
drawing_class	TYPE	464	683	75	973	New
dummy_gri	CONSTANT	12	677	N/A	N/A	New
dummy_tri	CONSTANT	12	677	N/A	N/A	New
dynamic_analysis_type	TYPE	104	683	22	920	Unchanged
eccentric_cone	ENTITY	6	744	80	978	New
eccentric_conical_volume	ENTITY	7	745	79	977	New
edge	ENTITY	9	745	70	968	Modified
edge_based_wireframe_model	ENTITY	6	745	46	944	Unchanged
edge_curve	ENTITY	9	745	70	968	Unchanged
edge_loop	ENTITY	9	745	71	969	Unchanged
edge_reversed	FUNCTION	11	874	N/A	N/A	Modified
effective_buckling_length	ENTITY	53	746	61	959	Unchanged
elastic_or_plastic_resistance	TYPE	223	684	36	934	Unchanged
element	ENTITY	129	746	25	923	Unchanged
element_curve	ENTITY	130	746	25	923	Unchanged
element_curve_complex	ENTITY	132	746	25	923	Unchanged
element_curve_simple	ENTITY	137	747	25	923	Unchanged
element_eccentricity	ENTITY	143	747	24	922	Unchanged
element_mapping	ENTITY	147	747	38	936	Unchanged
element_node_connectivity	ENTITY	147	747	24	922	Unchanged
element_point	ENTITY	149	748	25	923	Unchanged
element_point_grounded_damper	ENTITY	151	748	25	923	Unchanged
element_point_grounded_spring	ENTITY	152	748	25	923	Unchanged
element_point_stationary_mass	ENTITY	153	748	25	923	Unchanged
element_surface	ENTITY	154	748	25	923	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
element_surface_complex	ENTITY	156	749	25	923	Unchanged
element_surface_plane	ENTITY	156	749	25	923	Unchanged
element_surface_profiled	ENTITY	157	749	25	923	Unchanged
element_surface_shape	TYPE	105	684	25	923	Unchanged
element_surface_simple	ENTITY	158	749	25	923	Unchanged
element_volume	ENTITY	159	749	25	923	Unchanged
element_volume_complex	ENTITY	161	749	25	923	Unchanged
element_volume_shape	TYPE	106	684	25	923	Modified
element_volume_simple	ENTITY	162	749	25	923	Unchanged
element_with_material	ENTITY	162	750	25	923	Unchanged
elementary_surface	ENTITY	7	750	53	951	Unchanged
ellipse	ENTITY	7	750	51	949	Unchanged
ellipsoid	ENTITY	6	750	78	976	New
ellipsoid_volume	ENTITY	7	750	79	977	New
elliptic_area	ENTITY	6	750	78	976	New
evaluated_degenerate_pcurve	ENTITY	7	751	50	948	Unchanged
executed_action	ENTITY	4	751	5	903	Unchanged
extruded_area_solid	ENTITY	6	751	47	945	Unchanged
extruded_face_solid	ENTITY	6	751	47	945	Unchanged
fabrication_type	TYPE	261	684	62	960	Unchanged
face	ENTITY	9	751	71	969	Unchanged
face_based_surface_model	ENTITY	6	751	46	944	Unchanged
face_bound	ENTITY	9	752	70	968	Unchanged
face_bound_reversed	FUNCTION	11	874	N/A	N/A	Modified
face_outer_bound	ENTITY	9	752	70	968	Unchanged
face_reversed	FUNCTION	11	874	N/A	N/A	Modified
face_surface	ENTITY	9	752	71	969	Modified
faceted_brep	ENTITY	6	752	47	945	Unchanged
faceted_primitive	ENTITY	6	752	78	976	New
fastener	ENTITY	344	752	40	938	Unchanged
fastener_complex	ENTITY	345	752	40	938	Unchanged
fastener_mechanism	ENTITY	346	752	39	937	Unchanged
fastener_mechanism_with_position	ENTITY	347	753	39	937	New
fastener_simple	ENTITY	348	753	40	938	Unchanged
fastener_simple_bolt	ENTITY	349	753	41	939	Modified
fastener_simple_bolt_circular_head	ENTITY	351	753	41	939	Unchanged
fastener_simple_bolt_hexagonal_head	ENTITY	351	754	41	939	Unchanged
fastener_simple_bolt_square_head	ENTITY	353	754	41	939	Unchanged
fastener_simple_countersunk	ENTITY	354	754	40	938	Unchanged
fastener_simple_curved	ENTITY	355	754	40	938	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status
						1 st – 2 nd Edition
fastener_simple_nail	ENTITY	356	754	40	938	Unchanged
fastener_simple_nut	ENTITY	357	754	41	939	Modified
fastener_simple_nut_circular	ENTITY	358	755	41	939	Unchanged
fastener_simple_nut_closed	ENTITY	358	755	41	939	Unchanged
fastener_simple_nut_hexagonal	ENTITY	359	755	41	939	Unchanged
fastener_simple_nut_square	ENTITY	361	755	41	939	Unchanged
fastener_simple_pin	ENTITY	361	755	40	938	Unchanged
fastener_simple_screw	ENTITY	362	755	42	940	Unchanged
fastener_simple_screw_machine	ENTITY	363	756	42	940	Unchanged
fastener_simple_screw_self_drilling	ENTITY	364	756	42	940	Unchanged
fastener_simple_screw_self_tapping	ENTITY	365	756	42	940	Unchanged
fastener_simple_screw_tapered	ENTITY	367	756	42	940	Unchanged
fastener_simple_shear_connector	ENTITY	367	756	40	938	Unchanged
fastener_simple_stud	ENTITY	368	757	40	938	Modified
fastener_simple_washer	ENTITY	370	757	41	939	Modified
fastener_simple_washer_load_indicating	ENTITY	372	757	41	939	Unchanged
fastener_simple_washer_tapered	ENTITY	373	757	41	939	Unchanged
feature	ENTITY	289	758	43	941	Modified
feature_cutting_plane	ENTITY	290	758	43	941	Unchanged
feature_edge_chamfer	ENTITY	291	758	43	941	Unchanged
feature_edge_chamfer_fillet	ENTITY	292	758	43	941	Unchanged
feature_edge_chamfer_rounding	ENTITY	293	758	43	941	Unchanged
feature_edge_chamfer_straight	ENTITY	294	758	43	941	Unchanged
feature_surface	ENTITY	296	759	43	941	Modified
feature_surface_complex	ENTITY	296	759	43	941	Unchanged
feature_surface_name_tag	ENTITY	297	759	43	941	Unchanged
feature_surface_point	ENTITY	298	759	43	941	New
feature_surface_point_mark	ENTITY	299	759	43	941	New
feature_surface_simple	ENTITY	299	759	43	941	Unchanged
feature_surface_treatment	ENTITY	300	759	43	941	Unchanged
feature_surface_with_layout	ENTITY	301	759	43	941	Unchanged
feature_thread	ENTITY	301	760	43	941	Unchanged
feature_volume	ENTITY	303	760	44	942	Modified
feature_volume_complex	ENTITY	304	760	44	942	Unchanged
feature_volume_curved	ENTITY	305	760	44	942	Unchanged
feature_volume_curved_line	ENTITY	305	760	44	942	Unchanged
feature_volume_hole	ENTITY	307	760	44	942	Unchanged
feature_volume_hole_circular	ENTITY	308	761	44	942	Unchanged
feature_volume_hole_circular_threaded	ENTITY	309	761	44	942	Unchanged
feature_volume_hole_rectangular	ENTITY	310	761	44	942	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
feature_volume_hole_slotted	ENTITY	312	761	44	942	Unchanged
feature_volume_hole_slotted_curved	ENTITY	313	761	44	942	Unchanged
feature_volume_prismatic	ENTITY	315	762	44	942	Unchanged
feature_volume_prismatic_chamfer	ENTITY	316	762	45	943	Unchanged
feature_volume_prismatic_flange_chamfer	ENTITY	318	762	45	943	Unchanged
feature_volume_prismatic_flange_notch	ENTITY	320	762	45	943	Unchanged
feature_volume_prismatic_notch	ENTITY	321	762	45	943	Unchanged
feature_volume_prismatic_skewed_end	ENTITY	323	762	45	943	Unchanged
feature_volume_with_depth	ENTITY	327	763	44	942	New
feature_volume_with_layout	ENTITY	327	763	44	942	Unchanged
feature_volume_with_limit	ENTITY	328	763	44	942	New
feature_volume_with_process	ENTITY	329	763	44	942	Unchanged
first_proj_axis	FUNCTION	11	875	N/A	N/A	Modified
fixed_reference_swept_surface	ENTITY	7	763	53	951	New
flavour	ENTITY	586	763	2	900	Unchanged
force_measure	TYPE	544	684	17	915	Unchanged
force_measure_with_unit	ENTITY	552	763	15	913	Unchanged
force_per_length_measure	TYPE	545	684	17	915	Unchanged
force_per_length_measure_with_unit	ENTITY	553	764	17	915	Unchanged
force_per_length_unit	ENTITY	554	764	14	912	Unchanged
force_unit	ENTITY	555	764	14	912	Unchanged
founded_item	ENTITY	9	764	52	950	New
founded_item_select	TYPE	3	684	52	950	New
frame_continuity	TYPE	16	684	60	958	Unchanged
frame_type	TYPE	16	685	22	920	Unchanged
frequency_measure	TYPE	545	685	17	915	Unchanged
frequency_measure_with_unit	ENTITY	556	764	15	913	Unchanged
frequency_unit	ENTITY	557	765	14	912	Unchanged
functional_role	ENTITY	55	765	61	959	Unchanged
functional_role_documented	ENTITY	55	765	61	959	Unchanged
functionally_defined_transformation	ENTITY	9	765	13	911	Unchanged
geographical_location	ENTITY	67	765	11	909	Unchanged
geometric_curve_set	ENTITY	6	765	46	944	Unchanged
geometric_representation_context	ENTITY	7	766	12	910	Unchanged
geometric_representation_item	ENTITY	7	766	46	944	Modified
geometric_set	ENTITY	6	767	46	944	Unchanged
geometric_set_replica	ENTITY	6	767	46	944	Unchanged
geometric_set_select	TYPE	3	685	50	948	Unchanged
get_basis_surface	FUNCTION	11	876	N/A	N/A	Unchanged
get_description_value	FUNCTION	10	876	N/A	N/A	New

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
get_id_value	FUNCTION	10	877	N/A	N/A	New
get_instance_id	FUNCTION	609	877	N/A	N/A	New
get_item_cost_code	FUNCTION	610	878	N/A	N/A	New
get_item_ref	FUNCTION	611	878	N/A	N/A	New
get_name_value	FUNCTION	10	879	N/A	N/A	New
get_role	FUNCTION	10	879	N/A	N/A	New
global_location	ENTITY	68	767	11	909	Unchanged
global_or_local_load	TYPE	176	685	27	925	Unchanged
global_or_local_resistance	TYPE	223	685	36	934	Unchanged
global_uncertainty_assigned_context	ENTITY	9	767	12	910	Unchanged
global_unit_assigned_context	ENTITY	5	767	12	910	Unchanged
globally_unique_id	TYPE	655	685	1	899	New
grid	ENTITY	69	767	10	908	Unchanged
grid_intersection	ENTITY	71	768	10	908	Unchanged
grid_intersection_resolved	ENTITY	72	768	10	908	Unchanged
grid_level	ENTITY	73	768	10	908	Unchanged
grid_of_building	ENTITY	74	768	10	908	Unchanged
grid_of_site	ENTITY	74	768	10	908	Unchanged
grid_of_structure	ENTITY	75	769	10	908	Unchanged
grid_offset	ENTITY	75	769	10	908	Unchanged
grid_orthogonal	ENTITY	76	769	10	908	Unchanged
grid_radial	ENTITY	76	769	10	908	Unchanged
grid_skewed	ENTITY	77	769	10	908	Unchanged
gridline	ENTITY	78	769	10	908	Unchanged
group	ENTITY	4	769	2	900	Modified
group_assignment	ENTITY	5	770	2	900	Modified
group_assignment_actioned	ENTITY	470	770	2	900	Unchanged
group_assignment_approved	ENTITY	471	770	2	900	Unchanged
group_of_analysis_data	ENTITY	471	770	2	900	Unchanged
group_of_design_data	ENTITY	472	771	2	900	Unchanged
group_of_generic_data	ENTITY	473	771	2	900	Unchanged
group_of_physical_data	ENTITY	474	771	2	900	Unchanged
group_of_project_definition_data	ENTITY	475	771	2	900	Unchanged
group_of_structural_data	ENTITY	476	771	2	900	Unchanged
group_relationship	ENTITY	4	771	2	900	Modified
group_usage	ENTITY	477	771	2	900	Unchanged
half_space_2d	ENTITY	6	772	78	976	New
half_space_solid	ENTITY	6	772	48	946	Unchanged
hexahedron_volume	ENTITY	8	772	79	977	New
hour_in_day	TYPE	2	685	4	902	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
hyperbola	ENTITY	8	772	51	949	Unchanged
id_attribute	ENTITY	4	773	19	917	New
id_attribute_select	TYPE	2	686	19	917	New
identifier	TYPE	3	686	19	917	Unchanged
inertia_measure	TYPE	545	686	17	915	Unchanged
inertia_measure_with_unit	ENTITY	558	773	16	914	Unchanged
inertia_unit	ENTITY	559	773	14	912	Unchanged
intersection_curve	ENTITY	8	773	52	950	Unchanged
item_cost_code	ENTITY	587	773	74	972	New
item_cost_code_assigned	ENTITY	587	773	74	972	New
item_cost_code_with_source	ENTITY	588	774	74	972	New
item_defined_transformation	ENTITY	9	774	13	911	Unchanged
item_in_context	FUNCTION	12	879	N/A	N/A	Unchanged
item_property	ENTITY	589	774	74	972	Modified
item_property_assigned	ENTITY	590	774	74	972	Unchanged
item_property_with_source	ENTITY	591	774	74	972	New
item_ref_source	ENTITY	591	774	21	919	Unchanged
item_ref_source_documented	ENTITY	592	775	21	919	Unchanged
item_ref_source_library	ENTITY	593	775	21	919	Modified
item_ref_source_proprietary	ENTITY	594	775	21	919	Modified
item_ref_source_standard	ENTITY	595	775	21	919	Modified
item_reference	ENTITY	597	775	21	919	Unchanged
item_reference_assigned	ENTITY	598	776	21	919	Unchanged
item_reference_library	ENTITY	598	776	21	919	Unchanged
item_reference_proprietary	ENTITY	599	776	21	919	Unchanged
item_reference_standard	ENTITY	600	776	21	919	Unchanged
joint_system	ENTITY	374	776	39	937	Modified
joint_system_amorphous	ENTITY	375	777	39	937	Unchanged
joint_system_chemical	ENTITY	376	777	39	937	Unchanged
joint_system_complex	ENTITY	377	777	39	937	Unchanged
joint_system_mechanical	ENTITY	378	777	39	937	Unchanged
joint_system_welded	ENTITY	380	777	39	937	Modified
joint_system_welded_linear	ENTITY	380	778	39	937	Modified
joint_system_welded_point	ENTITY	382	778	39	937	Modified
joint_system_welded_surface	ENTITY	383	778	39	937	Modified
joint_system_welded_with_shape	ENTITY	384	778	39	937	New
knot_type	TYPE	3	686	54	952	Unchanged
label	TYPE	3	686	19	917	Unchanged
leap_year	FUNCTION	10	880	N/A	N/A	Unchanged
left_or_right	TYPE	287	686	45	943	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G		Status
				Diagram	Page	1 st – 2 nd Edition
length_measure	TYPE	2	686	17	915	Unchanged
length_measure_with_unit	ENTITY	5	778	15	913	Unchanged
length_unit	ENTITY	5	779	14	912	Unchanged
line	ENTITY	8	779	51	949	Unchanged
linear_acceleration_measure	TYPE	546	686	17	915	Unchanged
linear_acceleration_measure_with_unit	ENTITY	560	779	16	914	Unchanged
linear_acceleration_unit	ENTITY	561	779	14	912	Unchanged
linear_stiffness_measure	TYPE	546	686	17	915	Unchanged
linear_stiffness_measure_with_unit	ENTITY	562	779	16	914	Unchanged
linear_stiffness_unit	ENTITY	563	780	14	912	Unchanged
linear_velocity_measure	TYPE	546	687	17	915	Unchanged
linear_velocity_measure_with_unit	ENTITY	564	780	16	914	Unchanged
linear_velocity_unit	ENTITY	565	780	14	912	Unchanged
list_face_loops	FUNCTION	11	880	N/A	N/A	Unchanged
list_loop_edges	FUNCTION	11	881	N/A	N/A	Unchanged
list_of_reversible_topology_item	TYPE	3	687	71	969	Unchanged
list_of_topology_reversed	FUNCTION	12	881	N/A	N/A	Unchanged
list_to_array	FUNCTION	11	881	N/A	N/A	Unchanged
list_to_set	FUNCTION	12	882	N/A	N/A	Unchanged
load	ENTITY	189	780	27	925	Modified
load_case	ENTITY	190	781	30	928	Modified
load_case_documented	ENTITY	192	781	30	928	Unchanged
load_combination_occurrence	ENTITY	192	782	30	928	Unchanged
load_connection	ENTITY	193	782	76	974	Unchanged
load_element	ENTITY	194	782	27	925	Unchanged
load_element_concentrated	ENTITY	196	782	27	925	Unchanged
load_element_distributed	ENTITY	197	782	27	925	Modified
load_element_distributed_curve	ENTITY	198	783	27	925	Unchanged
load_element_distributed_curve_line	ENTITY	199	783	27	925	Unchanged
load_element_distributed_surface	ENTITY	200	783	27	925	Unchanged
load_element_distributed_surface_uniform	ENTITY	201	783	27	925	Unchanged
load_element_distributed_surface_varying	ENTITY	201	783	27	925	Unchanged
load_element_thermal	ENTITY	202	784	27	925	Unchanged
load_member	ENTITY	203	784	76	974	New
load_member_concentrated	ENTITY	204	784	76	974	New
load_member_distributed	ENTITY	205	784	76	974	New
load_member_distributed_curve	ENTITY	206	784	76	974	New
load_member_distributed_curve_line	ENTITY	208	785	76	974	New
load_member_distributed_surface	ENTITY	208	785	76	974	New
load_member_distributed_surface_uniform	ENTITY	209	785	76	974	New

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
load_member_distributed_surface_varying	ENTITY	210	785	76	974	New
load_node	ENTITY	211	785	27	925	Unchanged
loaded_product	ENTITY	212	786	20	918	Unchanged
loading_combination	ENTITY	213	786	30	928	Modified
loading_status	TYPE	401	687	69	967	Unchanged
local_time	ENTITY	4	786	4	902	Unchanged
located_assembly	ENTITY	440	786	37	935	Modified
located_assembly_child	ENTITY	442	787	37	935	Modified
located_assembly_marked	ENTITY	444	787	59	957	New
located_feature	ENTITY	445	787	37	935	Unchanged
located_feature_for_assembly	ENTITY	446	788	37	935	Unchanged
located_feature_for_design_part	ENTITY	446	788	37	935	Unchanged
located_feature_for_located_assembly	ENTITY	447	788	37	935	Unchanged
located_feature_for_located_part	ENTITY	449	788	37	935	Unchanged
located_feature_for_part	ENTITY	450	788	37	935	Unchanged
located_feature_joint_dependent	ENTITY	451	789	37	935	Unchanged
located_item	ENTITY	452	789	37	935	Unchanged
located_joint_system	ENTITY	453	789	37	935	Unchanged
located_part	ENTITY	455	789	37	935	Unchanged
located_part_joint	ENTITY	458	790	37	935	Unchanged
located_part_marked	ENTITY	459	790	59	957	New
located_site	ENTITY	460	790	37	935	Unchanged
located_structure	ENTITY	461	790	37	935	Unchanged
loop	ENTITY	9	791	70	968	Unchanged
make_array_of_array	FUNCTION	11	882	N/A	N/A	Modified
make_array_of_array_of_array	FUNCTION	11	883	N/A	N/A	New
managed_application_installation	ENTITY	657	791	1	899	Unchanged
managed_data_creation	ENTITY	659	791	1	899	Unchanged
managed_data_deleted	ENTITY	660	791	6	904	Unchanged
managed_data_export	ENTITY	661	792	1	899	Unchanged
managed_data_group	ENTITY	662	792	2	900	Unchanged
managed_data_import	ENTITY	663	792	1	899	Unchanged
managed_data_item	ENTITY	664	792	1	899	Modified
managed_data_item_with_history	ENTITY	666	792	1	899	New
managed_data_modification	ENTITY	669	793	1	899	Unchanged
managed_data_transaction	ENTITY	670	793	1	899	Unchanged
manifold_solid_brep	ENTITY	6	794	47	945	Unchanged
map_location	ENTITY	80	794	11	909	Unchanged
mapped_item	ENTITY	9	794	12	910	Unchanged
mass_measure	TYPE	2	687	17	915	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
mass_measure_with_unit	ENTITY	5	794	15	913	Unchanged
mass_per_length_measure	TYPE	547	687	17	915	Unchanged
mass_per_length_measure_with_unit	ENTITY	566	794	16	914	Unchanged
mass_per_length_unit	ENTITY	567	795	14	912	Unchanged
mass_unit	ENTITY	5	795	14	912	Unchanged
material	ENTITY	402	795	69	967	Modified
material_anisotropic	ENTITY	404	796	69	967	Unchanged
material_constituent	ENTITY	405	796	69	967	Unchanged
material_elasticity	ENTITY	406	796	69	967	Unchanged
material_hardness	ENTITY	409	796	69	967	Unchanged
material_isotropic	ENTITY	409	796	69	967	Unchanged
material_mass_density	ENTITY	410	796	69	967	Unchanged
material_orthotropic	ENTITY	411	796	69	967	Unchanged
material_property_context	ENTITY	412	797	69	967	Unchanged
material_property_context_dimensional	ENTITY	413	797	69	967	Unchanged
material_property_context_loading	ENTITY	414	797	69	967	Unchanged
material_property_context_strain	ENTITY	414	797	69	967	Unchanged
material_property_context_stress	ENTITY	415	797	69	967	Unchanged
material_property_context_temperature	ENTITY	416	797	69	967	Unchanged
material_representation	ENTITY	417	797	69	967	Unchanged
material_representation_item	ENTITY	418	798	69	967	Unchanged
material_strength	ENTITY	419	798	69	967	Unchanged
material_thermal_expansion	ENTITY	420	798	69	967	Unchanged
material_toughness	ENTITY	420	798	69	967	Unchanged
maximum_or_minimum	TYPE	223	687	35	933	Unchanged
measure_qualification	ENTITY	10	798	18	916	Unchanged
measure_select	TYPE	582	687	74	972	New
measure_value	TYPE	2	687	17	915	Unchanged
measure_with_unit	ENTITY	5	799	15	913	Unchanged
media_content	ENTITY	478	799	2	900	Unchanged
media_content_drawing	ENTITY	479	799	2	900	New
media_file	ENTITY	479	799	2	900	Modified
media_file_cnc	ENTITY	481	800	75	973	New
media_file_drawing	ENTITY	482	800	75	973	New
member_beam_role	TYPE	16	688	82	980	New
member_beam_type	TYPE	17	688	82	980	Unchanged
member_brace_type	TYPE	17	688	82	980	New
member_cable_type	TYPE	18	689	82	980	New
member_class	TYPE	18	689	60	958	Unchanged
member_column_type	TYPE	18	689	82	980	New

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G		Status
				Diagram	Page	1 st – 2 nd Edition
member_cubic_type	TYPE	19	689	60	958	Unchanged
member_linear_type	TYPE	19	690	60	958	Modified
member_planar_type	TYPE	20	690	60	958	Unchanged
member_plate_type	TYPE	20	690	82	980	New
member_role	TYPE	21	690	60	958	Unchanged
member_slab_type	TYPE	21	691	82	980	New
member_wall_type	TYPE	21	691	82	980	New
minute_in_hour	TYPE	2	691	4	902	Unchanged
minutes_rotation	TYPE	62	691	11	909	Unchanged
mixed_loop_type_set	FUNCTION	12	883	N/A	N/A	Modified
modulus_measure	TYPE	547	691	17	915	Unchanged
modulus_measure_with_unit	ENTITY	568	800	16	914	Unchanged
modulus_unit	ENTITY	569	801	14	912	Unchanged
moment_measure	TYPE	548	691	17	915	Unchanged
moment_measure_with_unit	ENTITY	570	801	16	914	Unchanged
moment_unit	ENTITY	571	801	14	912	Unchanged
month_in_year_number	TYPE	2	691	4	902	Unchanged
move	ENTITY	633	801	56	954	Unchanged
name_attribute	ENTITY	4	801	19	917	New
name_attribute_select	TYPE	2	692	19	917	New
named_unit	ENTITY	5	801	14	912	Unchanged
node	ENTITY	163	802	23	921	Unchanged
node_dependency	ENTITY	165	802	23	921	Unchanged
normalise	FUNCTION	11	884	N/A	N/A	Modified
numeric_measure	TYPE	2	692	17	915	Unchanged
object_role	ENTITY	4	802	75	973	New
offset_curve_2d	ENTITY	8	802	51	949	Unchanged
offset_curve_3d	ENTITY	8	803	51	949	Unchanged
offset_surface	ENTITY	8	803	53	951	Unchanged
open_path	ENTITY	9	803	71	969	Unchanged
open_shell	ENTITY	9	803	72	970	Unchanged
open_shell_reversed	FUNCTION	12	885	N/A	N/A	New
organization	ENTITY	5	803	3	901	Unchanged
organization_relationship	ENTITY	5	804	3	901	Modified
organization_relationship_contractual	ENTITY	81	804	57	955	Unchanged
organizational_address	ENTITY	5	803	3	901	Unchanged
orientation_select	TYPE	106	692	15	913	Unchanged
oriented_closed_shell	ENTITY	9	804	72	970	Unchanged
oriented_edge	ENTITY	9	804	70	968	Unchanged
oriented_face	ENTITY	9	805	71	969	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
oriented_open_shell	ENTITY	9	805	72	970	Unchanged
oriented_path	ENTITY	9	805	71	969	Modified
oriented_surface	ENTITY	8	805	53	951	New
orthogonal_complement	FUNCTION	11	885	N/A	N/A	Modified
outer_boundary_curve	ENTITY	8	806	52	950	Unchanged
parabola	ENTITY	8	806	51	949	Unchanged
parameter_value	TYPE	2	692	17	915	Unchanged
parametric_representation_context	ENTITY	9	806	12	910	Unchanged
part	ENTITY	262	806	62	960	Modified
part_complex	ENTITY	264	806	62	960	Unchanged
part_derived	ENTITY	264	806	62	960	Unchanged
part_map	ENTITY	265	807	62	960	Unchanged
part_prismatic	ENTITY	266	807	62	960	Unchanged
part_prismatic_complex	ENTITY	267	807	62	960	Unchanged
part_prismatic_complex_tapered	ENTITY	270	807	64	962	Unchanged
part_prismatic_simple	ENTITY	274	807	62	960	Unchanged
part_prismatic_simple_campered	ENTITY	275	808	64	962	Unchanged
part_prismatic_simple_campered_absolute	ENTITY	276	808	64	962	Unchanged
part_prismatic_simple_campered_relative	ENTITY	277	808	64	962	Modified
part_prismatic_simple_castellated	ENTITY	278	808	64	962	Unchanged
part_prismatic_simple_curved	ENTITY	280	808	62	960	Unchanged
part_select	TYPE	106	692	38	936	Unchanged
part_sheet	ENTITY	281	808	62	960	Unchanged
part_sheet_bounded	ENTITY	282	809	63	961	Unchanged
part_sheet_bounded_complex	ENTITY	283	809	63	961	Unchanged
part_sheet_bounded_simple	ENTITY	283	809	63	961	Unchanged
part_sheet_profiled	ENTITY	284	809	63	961	Unchanged
path	ENTITY	9	809	71	969	Modified
path_head_to_tail	FUNCTION	12	886	N/A	N/A	Unchanged
path_reversed	FUNCTION	12	886	N/A	N/A	Modified
pcurve	ENTITY	8	809	51	949	Unchanged
pcurve_or_surface	TYPE	3	692	52	950	Unchanged
person	ENTITY	5	810	3	901	Unchanged
person_and_organization	ENTITY	5	810	3	901	Modified
person_and_organization_role	ENTITY	5	810	3	901	Modified
personal_address	ENTITY	5	810	3	901	Unchanged
physical_action	ENTITY	214	811	26	924	Unchanged
physical_action_accidental	ENTITY	216	811	26	924	Unchanged
physical_action_permanent	ENTITY	217	811	26	924	Unchanged
physical_action_seismic	ENTITY	217	811	26	924	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
physical_action_variable	ENTITY	218	811	26	924	Unchanged
physical_action_variable_long_term	ENTITY	219	811	26	924	Unchanged
physical_action_variable_short_term	ENTITY	219	811	26	924	Unchanged
physical_action_variable_transient	ENTITY	220	812	26	924	Unchanged
placement	ENTITY	8	812	49	947	Unchanged
plane	ENTITY	8	812	53	951	Unchanged
plane_angle_measure	TYPE	2	692	17	915	Unchanged
plane_angle_measure_with_unit	ENTITY	5	812	15	913	Unchanged
plane_angle_unit	ENTITY	5	812	14	912	Unchanged
plane_stress_or_strain	TYPE	107	692	25	923	Unchanged
point	ENTITY	8	812	50	948	Modified
point_in_volume	ENTITY	8	813	50	948	New
point_on_curve	ENTITY	8	813	50	948	Unchanged
point_on_surface	ENTITY	8	813	50	948	Unchanged
point_replica	ENTITY	8	813	50	948	Unchanged
polar_point	ENTITY	8	813	50	948	New
poly_loop	ENTITY	9	814	70	968	Modified
polygonal_area	ENTITY	6	814	78	976	New
polyline	ENTITY	8	814	52	950	Unchanged
positive_length_measure	TYPE	2	692	17	915	Unchanged
positive_length_measure_with_unit	ENTITY	572	814	15	913	Unchanged
positive_plane_angle_measure	TYPE	2	693	17	915	Unchanged
positive_ratio_measure	TYPE	2	693	17	915	Unchanged
precision_qualifier	ENTITY	10	814	18	916	Unchanged
preferred_surface_curve_representation	TYPE	3	693	52	950	Unchanged
pressure_measure	TYPE	548	693	17	915	Unchanged
pressure_measure_with_unit	ENTITY	573	814	15	913	Unchanged
pressure_unit	ENTITY	574	814	14	912	Unchanged
primitive_2d	ENTITY	6	815	78	976	New
procure	ENTITY	634	815	57	955	Modified
product_item_select	TYPE	620	693	56	954	New
project	ENTITY	83	815	58	956	Unchanged
project_data_group	ENTITY	483	815	2	900	New
project_organization	ENTITY	83	815	58	956	Unchanged
project_plan	ENTITY	85	816	58	956	Unchanged
project_plan_item	ENTITY	86	816	58	956	Unchanged
project_plan_item_relationship	ENTITY	88	816	10	908	Unchanged
project_process_item	ENTITY	636	816	56	954	Modified
project_select	TYPE	425	693	37	935	Unchanged
projected_or_true_length	TYPE	176	693	27	925	Modified

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
pyramid_volume	ENTITY	8	817	79	977	New
qualitative_uncertainty	ENTITY	10	817	18	916	Unchanged
quasi_uniform_curve	ENTITY	8	817	55	953	Unchanged
quasi_uniform_surface	ENTITY	8	817	54	952	Unchanged
quasi_uniform_volume	ENTITY	8	817	80	978	New
ratio_measure	TYPE	2	693	17	915	Unchanged
ratio_measure_with_unit	ENTITY	5	817	15	913	Unchanged
ratio_unit	ENTITY	5	817	14	912	Unchanged
rational_b_spline_curve	ENTITY	8	818	55	953	Unchanged
rational_b_spline_surface	ENTITY	8	818	54	952	Unchanged
rational_b_spline_volume	ENTITY	8	818	80	978	New
reaction	ENTITY	245	819	33	931	Unchanged
reaction_acceleration	ENTITY	246	819	34	932	Unchanged
reaction_displacement	ENTITY	247	819	33	931	Unchanged
reaction_dynamic	ENTITY	249	820	34	932	Unchanged
reaction_equilibrium	ENTITY	250	820	33	931	Unchanged
reaction_force	ENTITY	252	820	33	931	Unchanged
reaction_velocity	ENTITY	253	820	34	932	Unchanged
rectangular_area	ENTITY	6	821	78	976	New
rectangular_composite_surface	ENTITY	8	821	54	952	Modified
rectangular_pyramid	ENTITY	6	821	78	976	New
rectangular_trimmed_surface	ENTITY	8	821	54	952	Unchanged
release	ENTITY	165	822	24	922	Unchanged
release_logical	ENTITY	167	822	24	922	Unchanged
release_spring_linear	ENTITY	168	822	24	922	Unchanged
release_spring_non_linear	ENTITY	171	823	24	922	Unchanged
release_warping	ENTITY	172	823	24	922	Unchanged
reparametrised_composite_curve_segment	ENTITY	8	823	52	950	Unchanged
representation	ENTITY	9	823	12	910	Modified
representation_context	ENTITY	9	824	12	910	Unchanged
representation_item	ENTITY	9	824	12	910	Unchanged
representation_map	ENTITY	10	824	12	910	Unchanged
representation_relationship	ENTITY	10	824	12	910	Unchanged
representation_relationship_with_transformation	ENTITY	10	824	12	910	Unchanged
resistance	ENTITY	255	825	36	934	Unchanged
resistance_axial	ENTITY	256	825	36	934	Unchanged
resistance_bending	ENTITY	257	825	36	934	Unchanged
resistance_shear	ENTITY	258	825	36	934	Unchanged
restraint	ENTITY	56	825	61	959	Unchanged
restraint_logical	ENTITY	57	826	61	959	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G		Status	
				Diagram	Page	1 st – 2 nd	Edition
restraint_spring	ENTITY	59	826	61	959	Unchanged	
restraint_warping	ENTITY	60	826	61	959	Unchanged	
reversible_topology	TYPE	3	693	71	969	Unchanged	
reversible_topology_item	TYPE	3	694	71	969	Unchanged	
revolved_area_solid	ENTITY	6	826	47	945	Modified	
revolved_face_solid	ENTITY	6	827	47	945	Modified	
right_angular_wedge	ENTITY	6	827	48	946	Unchanged	
right_circular_cone	ENTITY	6	827	48	946	Unchanged	
right_circular_cylinder	ENTITY	6	827	48	946	Unchanged	
role_association	ENTITY	4	828	75	973	New	
role_select	TYPE	2	694	75	973	New	
rotational_acceleration_measure	TYPE	548	694	17	915	Unchanged	
rotational_acceleration_measure_with_unit	ENTITY	575	828	16	914	Unchanged	
rotational_acceleration_unit	ENTITY	575	828	14	912	Unchanged	
rotational_stiffness_measure	TYPE	548	694	17	915	Unchanged	
rotational_stiffness_measure_with_unit	ENTITY	577	828	16	914	Unchanged	
rotational_stiffness_unit	ENTITY	577	828	14	912	Unchanged	
rotational_velocity_measure	TYPE	549	694	17	915	Unchanged	
rotational_velocity_measure_with_unit	ENTITY	579	829	16	914	Unchanged	
rotational_velocity_unit	ENTITY	580	829	14	912	Unchanged	
same_side	FUNCTION	11	887	N/A	N/A	New	
scalar_times_vector	FUNCTION	11	887	N/A	N/A	Modified	
seam_curve	ENTITY	8	829	52	950	Unchanged	
second_in_minute	TYPE	2	694	4	902	Unchanged	
second_proj_axis	FUNCTION	11	888	N/A	N/A	Modified	
seconds_rotation	TYPE	63	694	11	909	Unchanged	
section_profile	ENTITY	494	829	65	963	Modified	
section_profile_angle	ENTITY	496	830	65	963	Unchanged	
section_profile_centrelines	ENTITY	499	830	65	963	Unchanged	
section_profile_channel	ENTITY	500	830	65	963	Unchanged	
section_profile_circle	ENTITY	503	830	65	963	Unchanged	
section_profile_circle_hollow	ENTITY	504	831	65	963	Unchanged	
section_profile_complex	ENTITY	505	831	65	963	Unchanged	
section_profile_compound	ENTITY	506	831	65	963	Unchanged	
section_profile_derived	ENTITY	510	831	65	963	Unchanged	
section_profile_edge_defined	ENTITY	511	831	65	963	Unchanged	
section_profile_i_type	ENTITY	512	832	67	965	Unchanged	
section_profile_i_type_asymmetric	ENTITY	516	832	67	965	Unchanged	
section_profile_i_type_rail	ENTITY	518	832	67	965	Unchanged	
section_profile_rectangle	ENTITY	520	832	66	964	Unchanged	

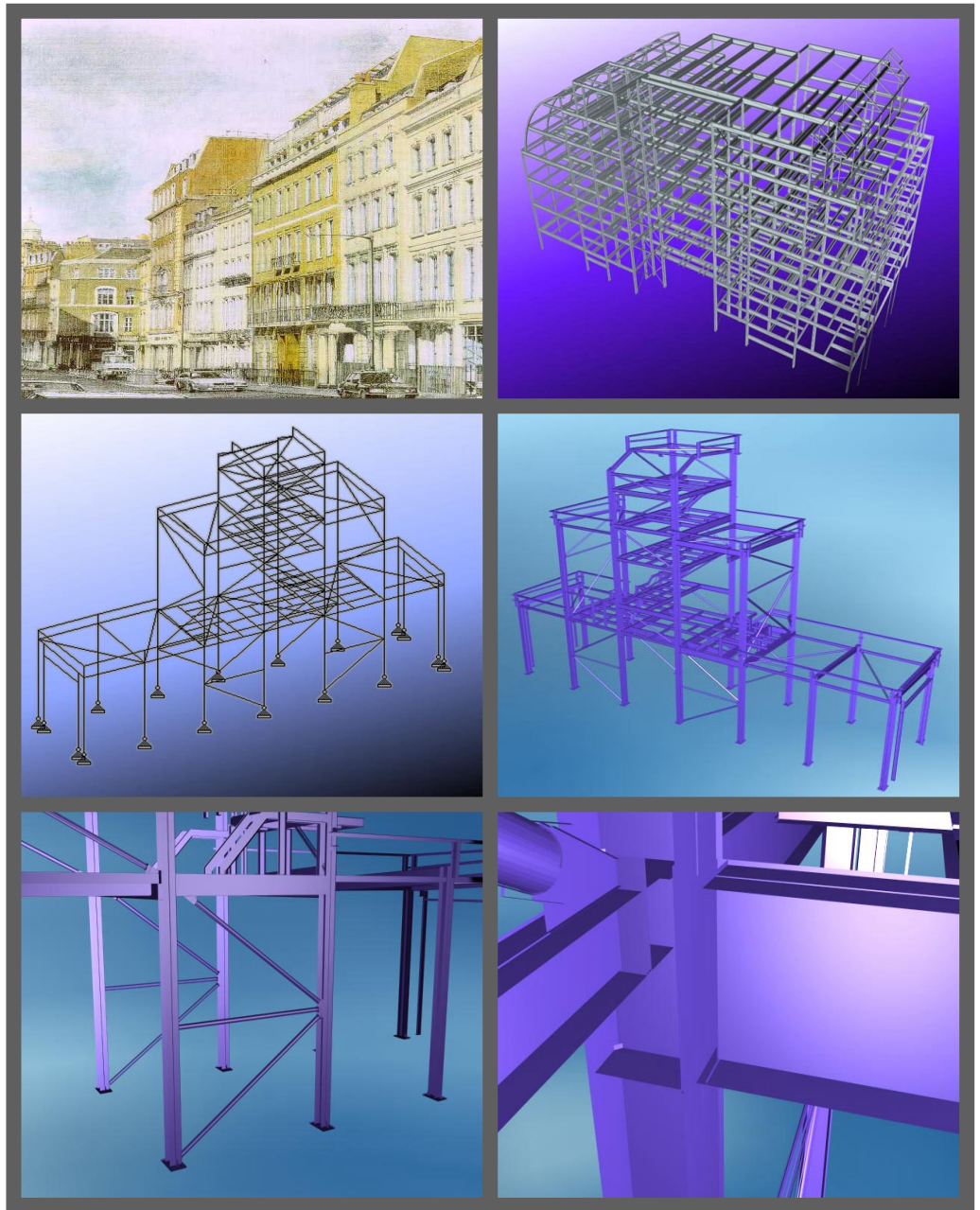
Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G		Status	
				Diagram	Page	1 st – 2 nd Edition	
section_profile_rectangle_hollow	ENTITY	522	833	66	964	Unchanged	
section_profile_simple	ENTITY	524	833	65	963	Unchanged	
section_profile_t_type	ENTITY	525	833	66	964	Unchanged	
section_properties	ENTITY	528	834	68	966	Unchanged	
section_properties_asymmetric	ENTITY	534	834	68	966	Unchanged	
select_analysis_item	TYPE	464	694	7	905	Unchanged	
select_analysis_model_item	TYPE	464	695	7	905	Unchanged	
select_data_item	TYPE	655	695	6	904	Unchanged	
select_data_source	TYPE	656	695	1	899	Unchanged	
select_design_item	TYPE	465	695	7	905	Unchanged	
select_generic_item	TYPE	465	696	6	904	Modified	
select_loading_item	TYPE	467	697	7	905	Unchanged	
select_physical_item	TYPE	467	697	7	905	Unchanged	
select_project_definition_item	TYPE	468	697	8	906	Unchanged	
select_response_item	TYPE	468	697	7	905	Unchanged	
select_structural_item	TYPE	469	698	8	906	Modified	
set_of_reversible_topology_item	TYPE	3	698	71	969	Unchanged	
set_of_topology_reversed	FUNCTION	12	888	N/A	N/A	Unchanged	
setting_out_point	ENTITY	89	835	11	909	Unchanged	
shape_representation	ENTITY	5	835	12	910	Unchanged	
shape_representation_with_units	ENTITY	537	835	12	910	Unchanged	
shell	TYPE	3	698	72	970	Unchanged	
shell_based_surface_model	ENTITY	6	835	46	944	Unchanged	
shell_based_wireframe_model	ENTITY	6	835	46	944	Unchanged	
shell_reversed	FUNCTION	12	889	N/A	N/A	Modified	
shop_or_site	TYPE	22	698	59	957	Unchanged	
si_prefix	TYPE	2	699	14	912	Unchanged	
si_unit	ENTITY	5	835	14	912	Unchanged	
si_unit_name	TYPE	2	699	14	912	Unchanged	
site	ENTITY	90	836	9	907	Unchanged	
site_select	TYPE	425	700	37	935	Unchanged	
site_with_shape	ENTITY	91	836	9	907	Unchanged	
solder	ENTITY	637	836	56	954	New	
soldering_type	TYPE	621	700	56	954	New	
solid_angle_measure	TYPE	2	700	17	915	Unchanged	
solid_angle_measure_with_unit	ENTITY	5	836	15	913	Unchanged	
solid_angle_unit	ENTITY	5	836	14	912	Unchanged	
solid_model	ENTITY	6	836	47	945	Modified	
solid_replica	ENTITY	6	837	47	945	Modified	
spatial_variation	TYPE	177	700	26	924	Unchanged	

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status
						1 st – 2 nd Edition
sphere	ENTITY	6	837	48	946	Unchanged
spherical_point	ENTITY	8	837	50	948	Modified
spherical_surface	ENTITY	8	837	53	951	Unchanged
spherical_volume	ENTITY	8	837	79	977	New
standard_uncertainty	ENTITY	10	838	18	916	Unchanged
start_or_end_face	TYPE	287	700	44	942	Unchanged
static_analysis_type	TYPE	107	700	22	920	Unchanged
static_or_dynamic	TYPE	177	701	26	924	Unchanged
step_file	ENTITY	484	838	2	900	Modified
structural_frame_item	ENTITY	601	838	20	918	Modified
structural_frame_item_approved	ENTITY	603	839	74	972	Unchanged
structural_frame_item_certified	ENTITY	604	839	74	972	Unchanged
structural_frame_item_documented	ENTITY	605	839	74	972	Unchanged
structural_frame_item_priced	ENTITY	606	839	74	972	Unchanged
structural_frame_item_relationship	ENTITY	607	839	74	972	Unchanged
structural_frame_process	ENTITY	638	839	56	954	Modified
structural_frame_product	ENTITY	615	840	20	918	Unchanged
structural_frame_product_with_material	ENTITY	617	840	20	918	Unchanged
structure	ENTITY	92	840	9	907	Unchanged
structure_select	TYPE	426	701	37	935	Unchanged
subedge	ENTITY	9	840	70	968	New
subface	ENTITY	9	840	71	969	Unchanged
surface	ENTITY	8	841	53	951	Unchanged
surface_curve	ENTITY	8	841	52	950	Unchanged
surface_curve_swept_area_solid	ENTITY	6	841	81	979	New
surface_curve_swept_face_solid	ENTITY	6	841	81	979	New
surface_curve_swept_surface	ENTITY	8	842	53	951	New
surface_model	TYPE	3	701	46	944	Unchanged
surface_of_linear_extrusion	ENTITY	8	842	53	951	Unchanged
surface_of_revolution	ENTITY	8	842	53	951	Modified
surface_patch	ENTITY	8	842	54	952	Unchanged
surface_replica	ENTITY	8	843	53	951	Modified
surface_treatment	ENTITY	639	843	57	955	Unchanged
surface_treatment_clean	ENTITY	639	843	57	955	Unchanged
surface_treatment_coat	ENTITY	640	843	57	955	Unchanged
surface_treatment_grind	ENTITY	641	843	57	955	Unchanged
surface_treatment_hard_stamp	ENTITY	642	843	57	955	Unchanged
surface_treatment_thermal	ENTITY	643	844	57	955	Unchanged
surface_treatment_thermal_timed	ENTITY	644	844	57	955	Unchanged
surface_weights_positive	FUNCTION	11	889	N/A	N/A	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G		Status
				Diagram	Page	1 st – 2 nd Edition
swept_area_solid	ENTITY	6	844	47	945	Modified
swept_face_solid	ENTITY	6	844	47	945	Modified
swept_surface	ENTITY	8	844	53	951	Modified
tetrahedron	ENTITY	6	845	78	976	New
tetrahedron_volume	ENTITY	8	845	79	977	New
text	TYPE	3	701	19	917	Unchanged
thermodynamic_temperature_measure	TYPE	2	701	17	915	Unchanged
thermodynamic_temperature_measure_with_unit	ENTITY	5	845	15	913	Unchanged
thermodynamic_temperature_unit	ENTITY	5	845	14	912	Unchanged
time_measure	TYPE	2	701	17	915	Unchanged
time_measure_with_unit	ENTITY	5	846	15	913	Unchanged
time_unit	ENTITY	5	846	14	912	Unchanged
top_or_bottom	TYPE	287	701	44	942	Unchanged
topological_representation_item	ENTITY	9	846	70	968	Unchanged
topology_reversed	FUNCTION	12	889	N/A	N/A	Unchanged
toroidal_surface	ENTITY	8	846	53	951	Unchanged
toroidal_volume	ENTITY	8	846	79	977	New
torus	ENTITY	6	847	48	946	Unchanged
transformation	TYPE	3	701	13	911	Unchanged
transition_code	TYPE	3	702	54	952	Unchanged
trimmed_curve	ENTITY	8	847	52	950	Unchanged
trimmed_volume	ENTITY	6	847	81	979	New
trimming_preference	TYPE	3	702	52	950	Unchanged
trimming_select	TYPE	3	702	52	950	Unchanged
truncated_pyramid	ENTITY	10	847	48	946	Unchanged
type_qualifier	ENTITY	10	848	18	916	Unchanged
uncertainty_measure_with_unit	ENTITY	10	848	15	913	Modified
uncertainty_qualifier	ENTITY	10	848	18	916	Unchanged
uniform_curve	ENTITY	8	848	55	953	Unchanged
uniform_surface	ENTITY	9	848	54	952	Unchanged
uniform_volume	ENTITY	9	848	80	978	New
unique_data_item	FUNCTION	673	892	N/A	N/A	Unchanged
unit	TYPE	2	702	14	912	Unchanged
using_items	FUNCTION	12	890	N/A	N/A	New
using_representations	FUNCTION	12	891	N/A	N/A	Unchanged
valid_calendar_date	FUNCTION	10	892	N/A	N/A	Unchanged
valid_measure_value	FUNCTION	12	893	N/A	N/A	Unchanged
valid_time	FUNCTION	10	893	N/A	N/A	Unchanged
valid_units	FUNCTION	10	893	N/A	N/A	Unchanged
value_qualifier	TYPE	3	702	18	916	Unchanged

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G Diagram	Page	Status 1 st – 2 nd Edition
vector	ENTITY	9	848	49	947	Unchanged
vector_difference	FUNCTION	11	895	N/A	N/A	Unchanged
vector_or_direction	TYPE	3	702	49	947	Unchanged
versioned_action_request	ENTITY	4	848	5	903	Modified
vertex	ENTITY	9	849	70	968	Unchanged
vertex_loop	ENTITY	9	849	72	970	Unchanged
vertex_point	ENTITY	9	849	70	968	Unchanged
vertex_shell	ENTITY	9	849	72	970	Unchanged
volume	ENTITY	9	849	79	977	New
volume_measure	TYPE	3	702	17	915	Unchanged
volume_measure_with_unit	ENTITY	5	849	15	913	Unchanged
volume_unit	ENTITY	5	850	14	912	Unchanged
volume_weights_positive	FUNCTION	11	896	N/A	N/A	Unchanged
wedge_volume	ENTITY	9	850	79	977	New
weld	ENTITY	645	850	56	954	Modified
weld_alignment	TYPE	331	702	77	975	New
weld_arc	ENTITY	646	850	56	954	New
weld_backing_type	TYPE	332	702	77	975	New
weld_beam	ENTITY	647	851	56	954	New
weld_configuration	TYPE	332	703	77	975	New
weld_gas	ENTITY	648	851	56	954	New
weld_intermittent_rule	TYPE	333	703	77	975	New
weld_mechanism	ENTITY	385	852	39	937	Modified
weld_mechanism_complex	ENTITY	387	852	77	975	New
weld_mechanism_fillet	ENTITY	390	852	77	975	New
weld_mechanism_fillet_continuous	ENTITY	391	853	77	975	New
weld_mechanism_fillet_intermittent	ENTITY	391	853	77	975	New
weld_mechanism_groove	ENTITY	393	853	77	975	New
weld_mechanism_groove_beveled	ENTITY	396	853	77	975	New
weld_mechanism_groove_butt	ENTITY	397	854	77	975	New
weld_mechanism_prismatic	ENTITY	397	854	77	975	New
weld_mechanism_spot_seam	ENTITY	400	854	77	975	New
weld_other	ENTITY	648	851	56	954	New
weld_penetration	TYPE	334	703	39	937	Unchanged
weld_pressure	ENTITY	649	851	56	954	New
weld_resistance	ENTITY	650	851	56	954	New
weld_shape_bevel	TYPE	338	703	77	975	New
weld_shape_butt	TYPE	339	704	77	975	New
weld_sidedness	TYPE	339	704	77	975	New
weld_stud	ENTITY	650	852	56	954	New

Construct Name	Construct Type	Short Form	Long Form	EXPRESS-G		Status	
				Diagram	Page	1 st – 2 nd	Edition
weld_surface_shape	TYPE	340	704	77	975	New	
weld_taper_type	TYPE	340	704	77	975	New	
weld_type	TYPE	341	704	39	937	Modified	
welding_type	TYPE	621	705	56	954	Unchanged	
welding_type_arc	TYPE	623	705	56	954	New	
welding_type_beam	TYPE	623	705	56	954	New	
welding_type_gas	TYPE	624	705	56	954	New	
welding_type_other	TYPE	624	706	56	954	New	
welding_type_pressure	TYPE	625	706	56	954	New	
welding_type_resistance	TYPE	627	706	56	954	New	
welding_type_stud	TYPE	628	706	56	954	New	
wire_shell	ENTITY	9	854	72	970	Unchanged	
wireframe_model	TYPE	3	707	46	944	Unchanged	
year_number	TYPE	2	707	4	902	Unchanged	
zone	ENTITY	93	855	9	907	Unchanged	
zone_bounded	ENTITY	94	855	9	907	Unchanged	
zone_of_building	ENTITY	96	855	9	907	Unchanged	
zone_of_building_storey	ENTITY	96	855	9	907	Unchanged	
zone_of_project	ENTITY	97	855	9	907	Unchanged	
zone_of_site	ENTITY	98	855	9	907	Unchanged	
zone_of_structure	ENTITY	98	856	9	907	Unchanged	
zone_of_structure_sequence	ENTITY	99	856	9	907	New	
zone_of_structure_sequence_lot	ENTITY	100	856	9	907	New	



The Steel Construction Institute

CIMsteel Integration Standards Release 2: Second Edition

Volume 5 Conformance Requirements

CIS/2.1



The Steel Construction Institute

The Steel Construction Institute is an independent, technical, member-based organization, dedicated to the development and promotion of the effective use of steel in construction. Founded in 1986, the SCI now has over 600 corporate members in 37 countries. The SCI's research and development activities cover many aspects of steel construction including multi-storey construction, industrial buildings, light gauge steel framing systems, stainless steel, fire engineering, bridge and civil engineering, offshore and hazard engineering, structural analysis systems, environmental engineering and information technology. The results of these projects are fed back to SCI members via a comprehensive package of benefits. Membership is open to all organizations and individuals that are concerned with the use of steel in construction. Members include designers, contractors, suppliers, fabricators, academics and government departments in the United Kingdom, elsewhere in Europe and in countries around the world. The SCI is financed by subscriptions from its members, revenue from research contracts and consultancy services, publication sales and course fees.

A Membership Information Pack is available free on request from:

The Membership and Development Manager, The Steel Construction Institute, Silwood Park, Ascot, Berkshire, SL5 7QN, UK.

Telephone: +44 (0)1344 623345, Fax: +44 (0)1344 622944.



School of Civil Engineering

The School of Civil Engineering in the University of Leeds is the largest in the United Kingdom, and has excellent contacts with the construction industry and professional institutions. Its staff members have a wide range of experience in the practice of engineering, planning, architecture, design, construction and management. The School is committed to Quality Assurance of its teaching. Contacts with industry include an Advisory Committee of senior engineers and architects in practice, an industrial tutor scheme for undergraduates involvement of practising specialists in student's projects, and lectures by eminent engineers, architects and landscape architects. Industry also participates by giving sponsorships, prizes, providing vacation work and welcoming its graduates on completion of their degree. The emphasis on multi-discipline work, design projects, communication skills and teamwork seems to make Leeds graduates particularly useful.

The Computer Aided Engineering (CAE) group is a multi-disciplinary research group concerned with the application of Information Technologies to the overall engineering process. The group gained an international reputation through its innovative work applying the concepts of product modelling to achieve a more efficient information flow in the construction sector.

For details of current research or a prospectus, please contact:

The School Secretary, School of Civil Engineering, University of Leeds, Leeds, LS2 9JT, UK. Telephone: +44 (0)113 343 2248, Fax: +44 (0)113 343 2265.

CIMsteel Integration Standards

Release 2: Second Edition

Volume 5 –

Conformance Requirements

A J Crowley BEng, PhD

A M Smith BSc, MSc

A S Watson BTech, PhD, CEng, MICE

Published by:

The Steel Construction Institute
Silwood Park
Ascot
Berkshire
SL5 7QN

Tel: 01344 623345
Fax: 01344 622944
URL: <http://www.steel-sci.org/>

In association with:

Computer Aided Engineering Group
School of Civil Engineering
The University of Leeds
Leeds
LS2 9JT

Tel: 0113 343 2282
Fax: 0113 343 2265
URL: <http://www.leeds.ac.uk/civil/>

This publication is derived from the deliverables of the CIMsteel Project

© 2000, 2003 The University of Leeds

© 2000, 2003 The Steel Construction Institute

The University of Leeds and the Steel Construction Institute wish to acknowledge the valuable contribution from other CIMsteel Collaborators, and from other parties, in the generation of this material.

Apart from any fair dealing for the purposes of research or private study or criticism or review, as permitted under the Copyright Designs and Patents Act, 1988, this publication may not be reproduced, stored or transmitted, in any form or by any means, without the prior permission in writing of the publishers, or in the case of reprographic reproduction only in accordance with the terms of the licences issued by the UK Copyright Licensing Agency, or in accordance with the terms of licences issued by the appropriate Reproduction Rights Organisation outside the UK.

Enquiries concerning reproduction outside the terms stated here should be sent to the publishers, The Steel Construction Institute, at the address given on the title page.

Although care has been taken to ensure, to the best of our knowledge, that all data and information contained herein are accurate to the extent that they relate to either matters of fact or accepted practice or matters of opinion at the time of publication, The Steel Construction Institute, The University of Leeds, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data and/or information or any loss or damage arising from or related to their use.

Publications supplied to the Members of the Institute at a discount are not for resale by them.

Publication Number: SCI-P-269


ISBN 1 85942 103 2 (1st Edition)

British Library Cataloguing-in-Publication Data. (1st Edition)

A catalogue record for this book is available from the British Library. (1st Edition)

Front cover images have been reproduced by kind permission of Whitby Bird & Partners, Rowen Structures Ltd, Taywood Engineering Ltd, and Philip Quantrill (Structural Engineers) Ltd.

This publication is supported by a companion web site at <http://www.cis2.org/>

 [®] is a Registered Trademark

FOREWORD

2nd Edition

This publication specifies what software vendors are required to do to make their applications ‘CIS/2-compatible’. It is the fifth of a series of SCI publications that document the Second Edition of the Second release of the CIMsteel Integration Standards (CIS/2.1). It tabulates the Conformance Classes and details the requirements of the formal Conformance Testing procedures, which test whether an application conforms to the CIS/2 specifications. Formal Conformance Testing is based on the procedures defined by STEP and is intended to provide software vendors with an independent ‘seal of approval’ and give the engineering end-users greater confidence when purchasing tested applications.

This Second Edition should be seen as a ‘point release’, rather than a whole ‘new’ release of the CIMsteel Integration Standards. Of greatest impact will be the relaxation of the Conformance Requirements for DMC translators. This new simplified approach to data management should allow many more CIS/2 vendors to add significant value to their software products at minimal effort.

In the 3 years since the first edition of CIS/2 was published, CIS/2 users have raised 103 issues. Most of these have been trivial, but the 25 ‘Major Technical’ issues warranted the development of a second edition of CIS/2. Further, second editions of the STEP Generic Resources^[13, 14, 15, 16] used in LPM/5 were published in 2000; the revised constructs contained therein are incorporated into LPM/6.

The development of the second edition was coordinated by the CIS/2 International Technical Committee (ITC); the body responsible for improving and maintaining CIS/2 as an open standard. Chaired by Chuck Eastman, the ITC comprises The Steel Construction Institute and Leeds University (the joint custodians of CIS/2), together with The American Institute of Steel Construction (AISC), Georgia Institute of Technology and the National Institute of Standards and Technology (NIST). The successful deployment of CIS/2 technology over the past years has been due to the efforts of software vendors who have developed commercially viable CIS/2 translators. Consequently, several of these companies are represented on the ITC, including: Bentley, CSC (UK) Ltd, CSI, Design Data, Fabtrol, Intergraph, RAM International, and Tekla.

The development of the second edition was sponsored by the American Institute of Steel Construction (AISC).

Previous Editions

The first release of the CIMsteel Integration Standards (CIS/1) was one of the deliverables of the Pan-European Eureka EU130 CIMsteel Project. Over a ten-year period, this extensive research and development project involved seventy organizations in ten countries; representing a wide cross section of the constructional steelwork industry, including designers, fabricators, software vendors, universities, research and trade organizations. In addition to the contributions of the individual collaborators, the CIMsteel project also received financial support from:

- the Forschungsförderungsfonds für die gewerbliche Wirtschaft (FFF) in Austria
- the Erhvervsfremme Styrelsen, Industrie-ministeriet in Denmark
- the Ministère de l'Industrie, Département Communication et Commerce Extérieur in France
- the Istituto Mobiliare Italiano in Italy
- the Senter Den Haag in The Netherlands
- the Department of Trade and Industry in the United Kingdom

The principal authors of this publication are Andrew Crowley of the SCI and Alastair Watson of the University of Leeds. The development of this publication was funded by the SCI, the University of Leeds and Corus Group plc (formerly British Steel plc). The authors would like to thank their (former) colleagues in the Computer Aided Engineering (CAE) Group within the School of Civil Engineering at the University of Leeds for their work on the CIMsteel Project; in particular Steven Bennett, Dimitris Christodoulakis, Paul Hirst, Nashett El-Kaddah, George Kamparis, Gareth Knowles, Peter Riley, Alan Smith, and Mike Ward. The authors would also like to acknowledge the valuable outputs of the CIMsteel 'Overall Design and Analysis Working Group'. This group was led by Richard Greiner of the Technical University of Graz, Austria, and included representatives from Leeds and Nottingham Universities, QSE, CSC, SCI, Ove Arup, Taywood from the UK, Ramboll (Denmark), TDV (Austria), CTICM (France), Sidercad (Italy) and FCSA (Finland).

The CIS are the result of considerable efforts by many persons representing CIMsteel collaborators, associate collaborators and other interested parties. The principal developers of the CIS were Leeds University & SCI (UK), CTICM & LGCH (France), TNO (Netherlands), Ramboll (Denmark), Italsiel & Sidercad (Italy). Inputs from beyond Europe, particularly from the USA and Japan are acknowledged. The issues raised by the international review team of AP230 have been invaluable in the development of CIS/2.

AISC endorsement of CIS/2

CIS/2 has been endorsed by the American Institute of Steel Construction (AISC) as the standard for the electronic exchange of structural steel project information for the North American structural steel design and construction industry^[7].

During 1998, the Electronic Data Interchange (EDI) Review Team of the AISC evaluated data transfer standards with a view to adopting one. On December 7th 1998, their recommendation of CIS/2 was approved by the AISC Board of Directors as part of the AISC Business Plan for Standardizing the Electronic Exchange of Structural Steel Project Information. Phase I of the AISC Business Plan (now completed) included the public endorsement of CIS/2 as the standard for the electronic exchange of structural steel project information for the entire U.S. structural steel design and construction industry, as well as the recommendation that it be adopted as an international standard. Phase II of the AISC Business Plan includes several activities that will promote and support CIS/2 and its implementation.

AISC, headquartered in Chicago, is a not-for-profit organization established in 1921 to serve the structural steel industry in the United States. The organization's mission is to promote the use of structural steel through research activities, market development, education, codes and specifications, technical assistance, quality certification and standardization. AISC maintains the specification for the design of structural steel framing in the U.S. It has a long tradition of more than 75 years of providing assurance and service to the steel construction industry by providing reliable information.

For further details, please contact:

Director of Information Technology,
American Institute of Steel Construction,
One East Wacker Drive, Suite 3100, Chicago, IL 60601-2001, USA.
Telephone: +1 312 670 5413, Fax: +1 312 670 5403

CONTENTS

	Page No.
FOREWORD	III
2 nd Edition	iii
Previous Editions	iv
AISC endorsement of CIS/2	v
CONTENTS	VI
SUMMARY	VIII
1 INTRODUCTION	1
2 CONFORMANCE CLASSES	3
2.1 What is a Conformance Class?	3
2.2 What is the purpose of a CC?	3
2.3 Why does CIS/2 use CCs?	4
2.4 Defining CCs	4
2.5 Implementation Considerations	5
2.6 Conformance Class Definitions	6
3 CONFORMANCE REQUIREMENTS	8
3.1 Types of CIS/2 implementations	8
3.2 Requirements for physical file data sharing	9
3.3 Requirements for in-memory data sharing	12
3.4 Requirements for DBMS data sharing	13
3.5 Requirements for Basic CIS Translators	14
3.6 Requirements for DMC Translators	20
3.7 Requirements for IDI Translators	24
3.8 Requirements for PMR-Enabled Translators	25
3.9 Requirements for PMRs	27
3.10 Details of the 'Capability Files'	30
3.11 Details of the Log Files	34
4 CONFORMANCE TESTING	50
4.1 Introduction	50
4.2 Types of Conformance Testing	51
4.3 Conformance Testing Statements	54
5 INDUSTRIAL REALIZATION	60
5.1 International considerations	60
5.2 Conventions	60

5.3	Units of Measure	61
5.4	Product Items	69
6	TEST CASES	73
7	REFERENCES	74
APPENDIX A	CONFORMANCE CLASS LISTING	76
A.1	Generic Conformance Classes	76
A.2	DMC Conformance Classes	77
A.3	Specific Conformance Classes	77

SUMMARY

This publication is the fifth of a series of SCI publications that make up the CIS/2 documentation. It specifies what software vendors are required to do when developing a CIS/2-conformant system and how they can make their applications ‘CIS/2-conformant’.

Other documents in this series include:

- *Volume 1: Overview*^[2]
- *Volume 2: Implementation Guide*^[3]
- *Volume 3: The Information Requirements*^[4]
- *Volume 4: The Logical Product Model (LPM/6)*^[5]
- *Volume 6: Worked Examples*^[6]

Taken as a whole, the CIS/2 documentation specifies what information may be transferred between software applications, and how that information must be structured in a repository or data exchange file. This publication includes a number of essential components that are used to test an implementation of CIS/2 against a software vendor’s claims of conformance to the CIS/2 specifications. It explains the ‘Conformance Classes’, which form the testable subsets of the implementation schema. It then explains the detailed ‘Conformance Requirements’ at each level and form of implementation, taking into consideration the increasing degrees of complexity and the requirements for operational documentation, error messages, and log files.

Later Sections provide a description of the formal ‘Conformance Testing’ procedures that demonstrate whether a specific implementation conforms to the CIS/2 specifications. These Sections discuss how a CIS translator should be submitted for conformance testing and how it will be tested for compatibility with CIS/2 and particular Conformance Classes. It also describes how problems with implementation are dealt with, and what degrees of ‘CIS Conformance’ can result.

Because the CIS has to address the needs of the steel construction industry world-wide, lists of ‘standard identifiers’ for various product items (defined in national standards) are included for implementation in CIS-conformant computer aided-engineering systems. This publication includes a textual description of how CIS/2 deals with national conventions and variations in the referencing of standard and manufacturers’ product items and ‘unit systems’.

1 INTRODUCTION

The preceding four volumes of the CIS/2 documentation have specified what information may be transferred between software applications, and how that information must be structured in a repository or data exchange file. This fifth volume specifies how an implementation of CIS/2 may be tested against the software vendor's claims of conformance to the CIS/2 specifications.

Typically, software vendors will be implementing CIS/2 to write export or import translators for an existing piece of application software that was specifically written for the specialist structural engineering market. In such a case, the whole 'system' must be considered. Thus, as illustrated in Figure 1.1, a 'CIS/2-conformant system' may include the combination of a piece of specialist application software together with its 'front end interface', its 'back-end operating system' and any 'internal' databases. CIS/2-conformant systems will also include purpose-built database management systems written around the CIS/2 specifications such as a Product Model Repository (PMR).

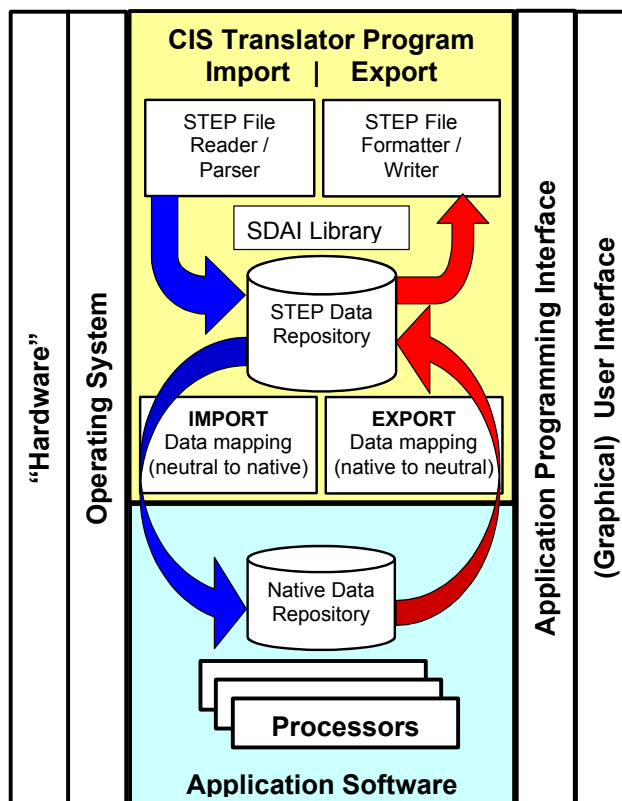


Figure 1.1 *Components of a Typical 'CIS/2-conformant system'*

Conformance of a 'CIS/2 system' shall be expressed as conformance to a combination of one or more Conformance Classes (CCs), including all types, entities, attributes, functions and rules contained in the CCs. A 'Conformance Class' is a testable subset of the implementation schema. A CIS/2-conformant system may also exhibit conformance to particular 'Flavours' (i.e. lists of standard or proprietary product references) and unit systems.

A CIS/2-conformant system is required to meet the specified 'Conformance Requirements' for the appropriate level and form of (STEP) implementation, taking into

consideration the degrees of complexity allowed by CIS/2. These ‘Conformance Requirements’ are detailed in Section 3, and include requirements for operational documentation, error messages, and log files.

‘Conformance Testing’ (described in Section 4) is the evaluation of an implementation for all the required characteristics to determine whether a ‘CIS/2 system’ conforms to the CIS/2 specifications, including any implementation-specific requirements defined in the ‘Conformance Requirements’. Conformance Testing will evaluate whether the CIS/2 system actually supports those aspects of the Logical Product Model (the entities, types, attributes, functions, and rules) that the software vendor claims it does.

Formal Conformance Testing is based on the procedures defined by STEP and is intended to provide software vendors with an independent ‘seal of approval’ and give the engineering end-users greater confidence when purchasing tested applications. The more formal and independent the testing regime is, the more confidence end-users will have in the application’s capabilities. The CIS/2 framework and methodology for Conformance Testing reflects the importance of testing to the success of any implementation of the CIS. Experience in other domains has shown that such a strategy is an essential prerequisite to repeatability and consistency of testing, and therefore to mutual recognition of test results across regional and national boundaries.

CIS/2 provides a suite of data exchange standards that are acceptable, and applicable, across the world. There are, however, significant national and regional differences in practice, terminology and custom. Three primary areas of international variation have been identified as:

- Conventions (such as axis systems)
- Units of measure
- Standard and Proprietary Product items

As described in Section 5, CIS/2 addresses international considerations in each of these three areas, thus providing a flexible and appropriate approach to international deployment. Within this context, CIS/2 provides two unit systems (SI and US Imperial) and employs international ‘Standard Identifiers’ for standard products such as hot rolled section profiles (the ‘Flavours’).

2 CONFORMANCE CLASSES

2.1 What is a Conformance Class?

A Conformance Class (CC) is a valid subset of the LPM/6 EXPRESS schema. A CC is specified as a short form EXPRESS schema using STEP schema interfacing. The CC schemas contain no entity or type declarations, merely the USE FROM and the REFERENCE FROM declarations. All the entities referred to in a CC schema are declared in the 'structural_frame_schema'; i.e. the LPM/6 long form schema documented in Volume 4^[5]. If instances of these entities were created and their attributes correctly populated, the set of data produced would be coherent and valid, and if it were exported from the STEP data repository, a valid STEP Part 21 file should result. At the same time, information conforming to a CC is compatible with the complete LPM/6 schema.

2.2 What is the purpose of a CC?

The primary purpose of a CC is to test a particular CIS implementation for conformance to the CIS/2 standard. For each CC, or combination of CCs, a number of test cases can be generated and used during formal Conformance Testing.

The LPM/6 long form EXPRESS schema is an extremely large and complex 'Product Model'. Moreover, LPM/6 is not a simple hierarchy of 'classified types' but a network of interrelated 'entity objects' that may be populated in many different ways to reflect their information content. Although it is theoretically possible to define CCs for each and every permutation of the possible populations of the LPM, it would not be practical to do so, as it would result in the creation of several hundred thousand CCs. The creation of 'modular' Conformance Classes was seen to be the more realistic approach. These modular CCs are then put together as required for implementation and testing.

Put simply, LPM/6 is far larger in scope and has a much greater range of functionality than any existing engineering software application. Thus, it is highly unlikely that any application vendor would implement the LPM/6 EXPRESS schema in its entirety. Furthermore, data is generally exchanged in relatively small packets. Given the life cycle of a steel structure, large exchanges of 'complete' models will be rare. Therefore, LPM/6 has been broken down into smaller manageable pieces. These pieces – the CCs – may be combined like building blocks in many different ways to define a specific collection of information.

CCs may also be used (individually or in combination) to formally describe the scope of a CIS/2-compatible application, and its translator. Obviously, the scope of the application will be greater than that of an individual CC, but the information that the end users will wish to share will fit into one or more combinations of CCs. The vendor of the engineering software application seeking CIS/2-conformance is at liberty to choose how the CCs are combined.

Although CCs may be used to formally define the scope of an exchange file, most engineering end-users will not need to know the details of CCs. All they need to be assured of is that when dealing with applications that have overlapping sets of CCs, some useful data exchange is possible.

2.3 Why does CIS/2 use CCs?

CIS/1 used the concept of Data Exchange Protocols (DEPs) and defined four DEPs to cover broad areas of analysis, member design, connection design and detailing. Unfortunately, these DEPs proved to be too broad in scope and too flexible on implementation to allow the enforcement of rigorous testing. CIS/2 uses the approach to implementation and testing that is used by STEP; i.e. Conformance Classes (CCs). This is one of the significant differences between CIS/1 and CIS/2, a difference which has had a major influence on the size and style of the data model for implementation (LPM/6). Since CCs are defined and tested at an entity level (rather than at the attribute level of a DEP), a CIS/2-compatible system is required to support all the attributes of all the entities within a CC.

2.4 Defining CCs

Almost half of LPM/6 has come directly from the STEP Generic Resources. These EXPRESS constructs (such as ‘date’ and ‘time’) have been included in LPM/6 to support the EXPRESS constructs created to represent structural engineering information. Although it is possible that CIS/2-conformant applications will create files containing only instances of these entities, this will not be the main concern of Conformance Testing. These STEP entities represent generic information. The end users of CIS/2-conformant systems are more interested in the specialist information that is captured by the ‘CIMsteel’ entities that are documented in the short form of the *Logical Product Model*^[5].

Therefore, CIS/2 defines CCs only for selected short form entities, with each CC based on one particular entity. These CCs are designed to be as minimal as practical, and include only those entities needed to support the selected ‘base entity’. In this way, they act as ‘building blocks’ such that larger CCs can be defined based on other smaller CCs plus selected entities. In many cases a ‘building block’ CC is defined based on a particular ‘high-level’ supertype entity (e.g. ‘element’) and other CCs are then based on this CC plus more specialized subtype entities (e.g. element_curve_simple). It should be noted that the CCs are trying to capture the majority of the likely implementations, rather than every possible implementation of the LPM.

Figure 2.1 and Figure 2.2 illustrate (in simplified EXPRESS-G) examples of how Conformance Classes are defined for LPM/6. A CC may be defined simply in terms of a list of entities. Thus, the first and second example CCs may be defined as:

- CC1 = entities A111, A11, A1, B1, B11, C1.
- CC2 = entities B1, B12, D1, E1.

Under the CIS approach, only six entities are included in the first example CC as shown in Figure 2.1. It can be seen that the CC is relatively small, but when implemented in conjunction with the second example base CC (shown in Figure 2.2) two ‘branches’ of the two supertype-subtype ‘trees’ are used to share specific and detailed information.

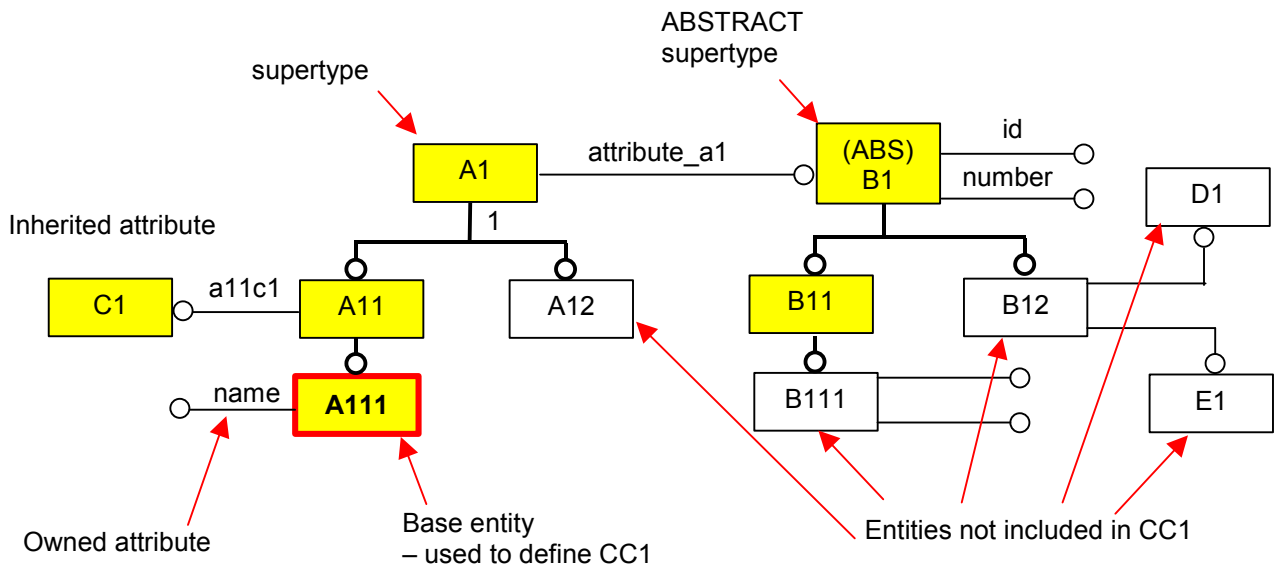


Figure 2.1 *Defining a Conformance Class for LPM/6*

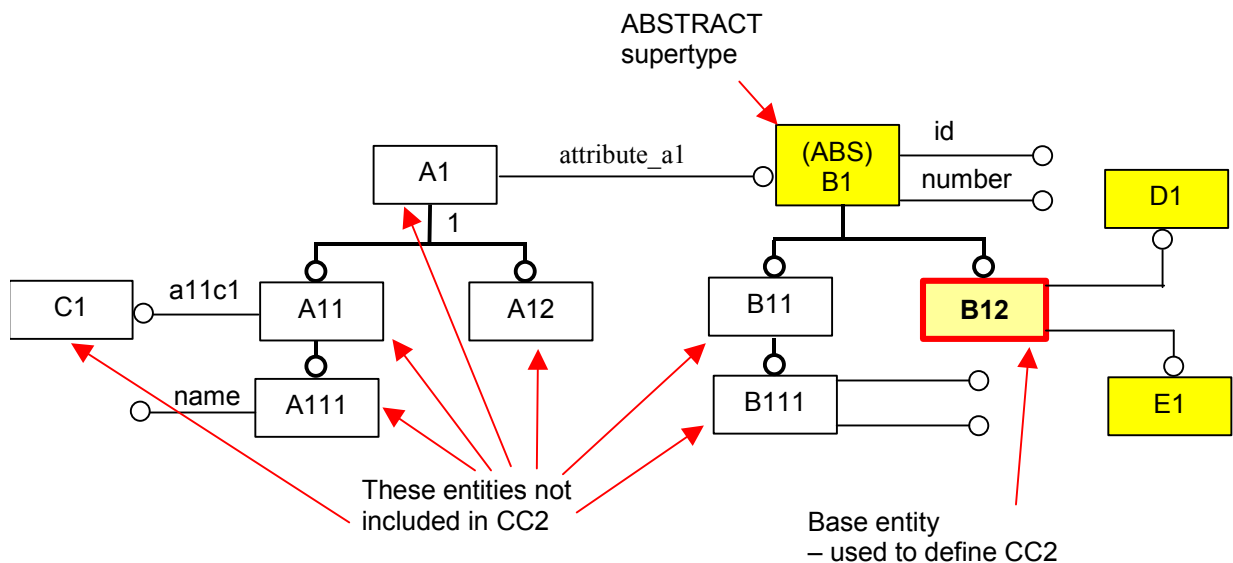


Figure 2.2 *Defining a second Conformance Class for LPM/6*

2.5 Implementation Considerations

In practice, software vendors will support a number of CCs; either individually or (more likely) in combination. If an application can support a combination of (say) CC105 and CC206, it is expected (but it is not required) to be able to support either of these CCs individually. The CCs are defined in such a way that an individual CC is complete and coherent in its own right. This means that combinations of CCs add information together. If an application cannot support a CC individually, but can only support it when it is combined with another CC, the formal specification of the conformance of the

CIS/2-system shall indicate this restriction. For example, an application that can support CC105 and CC206 both individually and in combination shall have its conformance specified as “CC105, CC206”. Whereas, an application that can only support CC105 when combined with CC206 shall have its conformance specified as “(CC105+CC206)”. If that application can also support CC206 individually, its conformance shall be specified as “(CC105+CC206), CC206”.

Remember, CCs define ‘legitimate’ sets of data, which may not necessarily be ‘meaningful’. Defining meaningful sets of data really is beyond the scope of the CIS, since it is very subjective and implementation dependent. It is also recognized that the capabilities of an application’s import translator may be different from that of its export translator. That is, a CIS/2-conformant system may be able to support one set of CCs on export and another set of CCs on import. These different sets of CCs may, but need not, overlap.

2.6 Conformance Class Definitions

Conformance Class definitions have been grouped in three categories as follows:

- Generic CCs (numbered CC001 to CC099), defined for common resources,
- Data management CCs (Numbered CC100 & 101), and
- Specific CCs (Numbered CC102 onwards), defined for data specific to structural engineering.

All the definitions are given as short form EXPRESS schemas.

The Specific CCs can be collected into four sub-categories; analysis, design, manufacturing, and project definition. There is, of course, overlap between each of these four sub-categories such that a ‘design’ CC would also be used in the manufacturing domain.

The Specific CCs may then be combined as required by those software vendors implementing CIS/2. CC combinations are defined in the same way as other CCs, i.e. as short form EXPRESS schemas. Although any individual software vendor is at liberty to define a CC combination, it is strongly recommended that CC combinations are only defined in consultation with the **CIS/2 International Technical Committee**. In this way, the CC combinations will define areas of overlapping information common to several applications.

The identification of which CC combinations are used frequently will be a gradual process of evolution. Once it is recognised that specific combinations are being frequently used by a number of vendors, these combinations will each be ‘legitimised’ by the allocation of an official short-hand designation in the range CC501 onwards. To encourage vendor (and end-user) co-operation in this process, this allocation of official designations is likely to be managed by the **CIS/2 International Technical Committee**.

A suite of EXPRESS files has been created for the Conformance Classes, and is available from the CIS/2 web site (www.cis2.org). Each file captures an individual Conformance Class as a short form EXPRESS schema. When used with a STEP developer’s toolkit, these files allow software vendors to produce a schema for testing their CIS implementations.

Although every care has been taken to ensure, to the best of our knowledge, that all data contained within these files are accurate to the extent that they conform to the published edition of the LPM/6 EXPRESS schema, The Steel Construction Institute, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data or any loss or damage arising from or related to their use. Any errors should be reported directly to The Steel Construction Institute.

For the 2nd Edition, the Conformance Classes are not documented in full, a list of classes is provided in Appendix A.

3 CONFORMANCE REQUIREMENTS

This Section explains the detailed conformance requirements at each level and form of implementation taking into consideration the increasing degrees of complexity. For each type of implementation, conformance requirements, in the form of a specification and commentary, are given under three general headings:

1. Implementation
2. Operational (including error messages & log files)
3. Documentation

The ‘Implementation requirements’ relate to the development and implementation of a CIS/2-conformant system, while the ‘Operational requirements’ relate to the demands made by an engineering end user when using such a system. The ‘Documentation requirements’ relate to the documents that must be supplied with a CIS/2-conformant system; both those supplied to the engineering end user, and those supplied to a formal Conformance Testing body.

3.1 Types of CIS/2 implementations

As discussed in the *Implementation Guide*^[3], CIS/2 may be implemented in any of five degrees of complexity, from basic file exchange through to advanced implementation in a Database Management System (DBMS).

Thus, CIS/2 defines conformance requirements for:

1. CIS translators that support the exchange of engineering data via physical files (known as ‘Basic CIS Translators’)
2. CIS translators that support data sharing and management (known as ‘DMC-Translators’)
3. CIS translators that also provide for incremental data import and incremental updating (known as ‘IDI Translators’)
4. CIS translators that supports data sharing and management through a CIS Product Model Repository (known as ‘PMR-enabled Translators’)
5. CIS Product Model Repositories (PMRs)

These progressive degrees of complexity allow vendors to extend the functionality of their CIS-conformant applications incrementally. The following sections of this publication describe the *additional* requirements that each increase in complexity brings.

It should also be noted that these five degrees of complexity run *parallel* to the levels of implementation referred to by STEP, which are:

- Level 1 – data sharing by means of exchange files
- Level 2 – data sharing using a standard in-memory data format

- Level 3 – data sharing using a database management system as the means of data storage and access
- Level 4 – data storage and access via a knowledge-base system

In theory, CIS/2 may be implemented at any ‘degree of complexity’ and any ‘level of implementation’. Thus, a DMC-translator may use physical data exchange files as its medium of communication. Similarly, a PMR is one type of database implementation of the CIS, but not the only one. Software vendors are required to consider and satisfy both the requirements for the degree of complexity of their CIS implementation and the requirements of the level of STEP implementation.

As CIS/2 does not specify any particular requirements for knowledge-based systems, the following sections specify the requirements for eight different aspects of CIS/2 implementation; three due to the mechanism chosen for data exchange, and five due to the degree of complexity chosen.

3.2 Requirements for physical file data sharing

A STEP Level 1 Implementation of CIS/2 may be used as the means of communication for any CIS/2-conformant software system (e.g. an application with its purpose-written translators), regardless of its degree of complexity. That is, Basic CIS Translators, DMC Translators, IDI Translators and PMRs are all likely to use physical files when sharing data, although they may equally use other forms. For the purposes of data exchange, physical files are seen as the lowest common denominator.

When implementing the CIS for exchanging files between applications, software vendors are required to produce translators that are capable of reading or writing CIS data exchange files in accordance with STEP Part 21 *Implementation Methods: Clear text encoding of the exchange structure* (ISO 10303-21: 2002) which incorporates the 1996 *Technical Corrigendum*^[11]. The requirements of the physical file format are been briefly described in the *Implementation Guide*^[3], and are described in more detail in STEP Part 21. These requirements should be enforced automatically by a good ‘STEP developer’s toolkit’. However, vendors are also advised to consult the ISO document. STEP file exchange implementations simply rely on conformance to an agreed EXPRESS schema and the use of an agreed physical file format. There are also a few limitations imposed by the CIS/2 specifications, which are explained in the following sections.

When sharing data via standard physical files, CIS/2 expects the software vendor’s implementation to fulfil the requirements given in Sections 3.2.1, 3.2.2, and 3.2.3.

3.2.1 Implementation requirements for physical file data sharing

The following requirements are given for an export translator of a CIS/2-conformant system developed for a STEP Level 1 Implementation of CIS/2. An import translator shall have corresponding abilities to read and recognize data in physical files created in accordance with those requirements.

1. The export translator is required to be able to create physical files whose format is in accordance with the 2002 International Standard issue of STEP Part 21 - *Implementation Methods: Clear text encoding of the exchange structure* – which incorporates the 1996 *Technical Corrigendum*^[11].

2. The export translator shall force the addition of the extension ‘.stp’ to the name of all physical files that it creates.
3. The ‘exchange structure’ (the format of the physical file) shall be a sequential file using a clear text encoding.
4. The special token ‘ISO-10303-21;’ shall be used to open an exchange structure, and the special token ‘END-ISO-10303-21;’ shall be used to close an exchange structure.
5. The exchange structure shall consist of two sections: a header section (providing data relating to the exchange structure itself), and a data section (providing the data to be transferred). The special token ‘HEADER;’ and ‘DATA;’ shall be used to open the header and data sections respectively. The special token ‘ENDSEC;’ shall be used to close each section.
6. The export translator is required to support STEP Part 21 implementation conformance class 1.
7. The export translator shall populate one instance of each of three entities in the header section of the physical file. All attributes of all three entities shall be given values. The header section entities shall appear in the order: ‘file_description’, ‘file_name’, and ‘file_schema’.
8. The ‘implementation_level’ attribute of the ‘file_description’ header section entity shall be given the value ‘2;1’.
9. The ‘schema_name’ attribute of the ‘file_schema’ header section entity shall be given the value ‘STRUCTURAL_FRAME_SCHEMA’.
10. CIS implementations are required to place the value of the ‘Object Identifier’ (provided in *Volume 4*^[5]) in the first list element of the attribute ‘description’ of the entity ‘file_description’.
11. The data section of the physical shall contain only instances of entities defined by long form EXPRESS schema of version 6 the Logical Product Model (LPM/6).
12. User-defined entity instances (described in clause 10.4 of STEP Part 21) shall not be used.

Commentary

The header section entities are fully defined in header_section_schema in clause 9.2 of ISO 10303-21:2002^[11].) The HEADER section is populated using a STEP developer’s toolkit, while the DATA section is read with the EXPRESS schema (the data in the DATA section is meaningless without reference to the appropriate schema).

Although it is not a Conformance Requirement of CIS/2, software vendors are strongly recommended to place the list of the Conformance Classes (CCs) relevant to the data in the data section in the second and subsequent list elements of the attribute ‘description’ of the entity ‘file_description’. The data provided in the data section is thereby specified as being conforming with those CCs.

An example of a populated header schema as encoded in the exchange structure with comments is shown below:

```

ISO-10303-21;
HEADER;
FILE_DESCRIPTION ( ('{cimsteel logical product model version (6) object (1) structural-
frame-schema(1)}',
/* OBJECT IDENTIFIER */
'CC110', 'CC170', 'CC118', 'CC009'),
/* CONFORMANCE CLASSES */
'2;1');
/* IMPLEMENTATION LEVEL */
FILE_NAME('dmc1.stp',
/* FILENAME */
'1999-5-28 T18:15:13',
/* CREATION DATE */
('Andrew Crowley'),
/* AUTHOR */
('Steel Construction Institute'),
/* ORGANIZATION */
'1.5.1 (compiled 17:44 11/24/1998)',
/* PROCESSOR VERSION */
'NIST EXPRESSO',
/* PROCESSOR */
'AJC');
/* AUTHORIZATION */
FILE_SCHEMA (('STRUCTURAL_FRAME_SCHEMA'));
ENDSEC;

```

3.2.2 Operational requirements for physical file data sharing

1. Any entity that is instanced in a physical file shall have all its non-OPTIONAL attributes populated. Appropriate values shall be assigned to the attributes of those LPM/6 entities using a mapping defined by the software application vendor.
2. When importing or exporting data, a CIS/2-conformant system is required to automatically generate a log file. This 'log file' shall be displayed incrementally on the screen as the translation proceeds. It shall also be written as an ASCII text file for subsequent inspection and/or printing by the user.
3. For STEP Level 1 implementations, the log file shall be named after the physical file with a '.log' extension; for example, <filename>_import.log for an import log file or <filename>_export.log for an export log file.
4. The details to be written to the log file shall be as given in Section 3.11.

Commentary

A STEP Level 1 Implementation contributes the section 1a of the log file, giving details of the medium of communication (i.e. the physical file). Other sections are added to the log file, depending on the degree of complexity of the CIS implementation

3.2.3 Documentation requirements for physical file data sharing

There are no specific documentation requirements related to physical file data sharing, other than those detailed in other sections of this publication for the particular type of translator.

CIS/2 places no additional requirements on level 1 implementations, but see also Section 2 and Appendix F of the *Implementation Guide*^[3].

3.3 Requirements for in-memory data sharing

A STEP Level 2 Implementation may be used as the means of communication for any CIS/2 implementation, regardless of its degree of complexity. That is, Basic CIS Translators, DMC Translators, IDI Translators and PMRs may all use a standard in-memory format when sharing data, although they may equally use other forms. However, it is recommended that Basic CIS Translators are limited to Level 1 implementation.

When sharing data via a standard in-memory format, CIS/2 expects the software vendor's implementation to fulfil the requirements given in Sections 3.3.1, 3.3.2, and 3.3.3.

3.3.1 Implementation requirements for in-memory data sharing

1. A translator of a CIS/2-conformant system developed for a STEP Level 2 Implementation of CIS/2 shall use an API that is in accordance with the 1998 International Standard issue of the Standard Data Access Interface SDAI (ISO 10303-22: 1998^[12]).

3.3.2 Operational requirements for in-memory data sharing

1. When importing or exporting data, a CIS/2-conformant system is required to automatically generate a log file. This 'log file' shall be displayed incrementally on the screen as the translation proceeds. It shall also be written as an ASCII text file for subsequent inspection and/or printing by the user.
2. For STEP Level 2 implementations, the electronic ASCII text version of the log file shall be given the extension '.log'.
3. The details to be written to the log file shall be as given in Section 3.11.

Commentary

The medium of communication is at the discretion of the software vendor. Software vendors need to consider the requirements of any network protocols for access control and transportation deemed necessary for in-memory data sharing.

A STEP Level 2 Implementation contributes the section 1b of the log file. Other sections are added to the log file, depending on the degree of complexity of the CIS implementation

A default file name may be assigned to the log file by the CIS/2-conformant system, at the discretion of the software vendor.

3.3.3 Documentation requirements for in-memory data sharing

There are no specific documentation requirements related to in-memory data sharing, other than those detailed in other sections of this publication for the particular type of translator.

CIS/2 places no additional requirements on level 2 implementations, but see also Section 3 of the *Implementation Guide*^[3].

3.4 Requirements for DBMS data sharing

A STEP Level 3 Implementation may be used as the means of communication for any CIS/2 implementation, regardless of its degree of complexity. That is, Basic CIS Translators, DMC Translators, IDI Translators and PMRs may all use a database management system (DBMS) when sharing data, although they may equally use other forms. However, it is likely that STEP Level 3 Implementations will be used only with DMC or IDI Translators, or in conjunction with PMRs.

When sharing data via a database management system, CIS/2 expects the software vendor's implementation to fulfil the requirements given in Sections 3.4.1, 3.4.2, and 3.4.3.

3.4.1 Implementation requirements for DBMS data sharing

1. A translator of a CIS/2-conformant system developed for a STEP Level 3 Implementation of CIS/2 shall use an API that is in accordance with the 1998 International Standard issue of the Standard Data Access Interface SDAI (ISO 10303-22: 1998^[12])

3.4.2 Operational requirements for DBMS data sharing

1. When importing or exporting data, a CIS/2-conformant system is required to automatically generate a log file. This 'log file' shall be displayed incrementally on the screen as the translation proceeds. It shall also be written as an ASCII text file for subsequent inspection and/or printing by the user.
2. For STEP Level 3 implementations, the log file shall be named after the 'STEP model' held in the repository, and it shall be given the extension '.log'; for example, *<model name>_import.log* for an import log file or *<model name>_export.log* for an export log file.
3. The details to be written to the log file shall be as given in Section 3.11.

Commentary

STEP Level 3 Implementations contribute the section 1c of the log file, giving details of the medium of communication (i.e. the STEP model held in the translator's repository). Examples are shown in Section 3.11. Other sections are added to the log file, depending on the degree of complexity of the CIS implementation.

3.4.3 Documentation requirements for DBMS data sharing

There are no specific documentation requirements related to DBMS data sharing, other than those detailed in other sections of this publication for the particular type of translator.

CIS/2 places no additional requirements on level 2 implementations, but see also Section 4 and Appendix G of the *Implementation Guide*^[3].

3.5 Requirements for Basic CIS Translators

When sharing data via a Basic CIS Translator, CIS/2 expects the software vendor's implementation to fulfil the requirements given below. These are *additional* to those determined by the STEP implementation level chosen.

3.5.1 Implementation requirements for Basic CIS Translators

1. The translator shall use STEP implementation level 1. That is, it shall support data sharing via physical files.
2. The translator shall be capable of processing (on export or import) data contained within physical files whose format is in accordance with implementation conformance class 1 of STEP Part 21^[1].
3. The scope structure described in clause 10.3 of STEP Part 21 shall not be used by Basic CIS Translators.
4. The translator shall be capable of processing data for the combinations of Conformance Classes, Flavours and units systems stated in the documentation supplied by the software vendor.
5. The translator shall support at least one Conformance Class or CC Combination.
6. Where an application supports standard or proprietary products, the translator shall support the equivalent standard or proprietary reference, such that the relevant information may be '*passed by reference*'. The translator shall also allow the information about those products to be '*passed by attribute value*'.
7. Where an application supports the representation of physical quantities that are provided with units in the application, the translator shall support the equivalent unit system and carry out conversions where necessary so that the information can be exchanged in '*SI units*' or '*US Imperial Units*'. Where an application supports both SI units and US Imperial unit systems, the translator shall allow the user to select which unit system is used for a particular exchange.
8. When an entity is instanced in a physical file, an export translator shall give all the non-OPTIONAL attributes an appropriate value. OPTIONAL attributes may be assigned a null value, which shall be encoded in the physical file as a '\$'. Appropriate values shall be assigned to the attributes of entities of the supported CC(s) using a mapping from the native application defined by the software vendor.
9. When an entity is imported from a physical file, the import translator shall import correctly all the populated attribute values (OPTIONAL attributes may have been given null values). The value of an attribute will normally be mapped by the import translator into the internal data structures of the application using a mapping defined by the software vendor.

Commentary

Implementation Level

In addition to STEP Level 1, a Basic CIS Translator may also use any of the other STEP implementation levels described in the *Implementation Guide*. However, it is

recommended that Basic CIS Translators be limited to STEP Level 1 (physical file exchange).

Conformance Classes

It is likely that a Basic CIS Translator will support several Conformance Classes in various combinations. The particular combinations of LPM entities that an application will be able to export or import will depend upon the list of combinations of CCs that it supports.

Unit Systems

An export translator with multiple unit system capabilities should have the ability to export data to a CIS data exchange file using either of the two ‘standardized’ unit systems (i.e. SI, US Imperial). Such an export translator should allow the user to select which unit system is to be used when writing a CIS data exchange file.

It should be noted that, although LPM/6 allows any unit system to be defined, CIS/2 defines only two ‘standardized’ unit systems: ‘*SI*’ and ‘*US Imperial*’. A set of units for particular physical quantities (e.g. length, mass, time) for each system is documented in Section 5.3 of this publication, and are specified in two separate CIS/2-conformant physical files. These ‘standardized’ unit systems are limited subsets of their parent unit systems. For example, the ‘standardized’ SI Units System (as defined by CIS/2) provides only one value for length (millimetres). Other units of length (e.g. kilometres) are excluded from this ‘standardized’ system. Furthermore, any derived units not defined in the ‘standardized’ SI Unit system that require a length component should be defined in terms of mm.

While Basic CIS Translators are required to support either (or both) of the ‘standardized’ unit systems when handling data representing physical quantities, more advanced CIS translators are at liberty to support other unit systems in addition to either (or both) of the ‘standardized’ unit systems.

The Scope Structure

While the Scope Structure may not be used for Basic translators, more advanced CIS implementations may do so.

3.5.2 Operational requirements for Basic CIS Translators

1. The translator shall be capable of processing data for the combinations of Conformance Classes, Flavours and units systems stated in the documentation supplied by the software vendor.
2. The translator shall allow the user to generate an ASCII file that describes the capabilities of the translator for import or export operations. For an export translator, this ‘capability’ file shall be named CIS_OUT.TXT, and it shall describe the scope of the information that the translator supports on export from an application (into a physical file). For an import translator, the file shall be named CIS_IN.TXT and shall describe the scope of the information that the translator supports on import into an application (from a physical file). (The structure of the capability file is shown in Table 3.1 while an example is shown in Figure 3.4.)
3. The translator shall be activated by a single menu command or icon within the source application.

4. The translator shall contribute Sections 2, 3, and 4 to the Log File, giving details of the data exchanged. (Examples are shown in Section 3.11.)
5. The export Log File shall report fully on each and every instance of data not derived from the internal data structure of the application.
6. The import Log File shall report fully on each and every instance of data not replicated within the data structure of the application.
7. Where an export translator populates a physical file with default values, the default values provided by the translator shall be appropriate to the data exported.
8. The translator shall allow the user to configure the default location of the directories (or folders) that hold the imported or exported CIS data exchange files. That is, the location of the default directory from which an import translator reads physical files should be user configurable. Similarly, the default location of the directory to which an export translator writes physical files should be user configurable.
9. By default, the import translator shall operate in a ‘displace’ mode, that is, any existing data within the application will be deleted at the start of the import process. The translator shall show a dialogue box that alerts the user to the mode of operation to ensure that the user does not inadvertently delete useful data. A translator may also provide the option to operate in ‘additive’ mode – such that existing data within the application is preserved during the import process. Details of this mode are left to the individual software vendor. However, the CIS/2-conformant system and its documentation must make clear to the end user that this additive mode of operation is not the same as an advanced Incremental Data Import (IDI) translator.
10. Where a limitation of the translator is encountered during an export or import process, the limitation shall be brought to the attention of the user via a dialogue box. In such an event, the translator shall display the appropriate error message (e.g. “Do you wish to continue exporting data from the application”, or “Do you wish to continue importing data into the application). Such a limitation shall also be recorded in the Log File.
11. Where a limitation of CIS/2 is encountered during an export or import process, it shall be brought to the attention of the user via a dialogue box. In such an event, the translator shall display the appropriate error message (e.g. “Do you wish to continue exporting data from the application”, or “Do you wish to continue importing data into the application). Such a limitation shall also be recorded in the Log File.
12. If a translator supports product references or ‘flavours’, it shall give the user the choice of either passing by reference, or passing by attribute value, or passing by both reference and attribute value.
13. If a user attempts to import a physical file that uses a product reference that is not supported by that translator, the translator shall give a meaningful warning message. The user shall be asked whether they wish to continue with the translation operation. (The translator may, but need not, offer the user an alternative value for the unrecognized product reference. In such a case, the selection shall be recorded in the Log File.)

14. An export translator with multiple unit system capabilities (i.e. one that has the ability to export data to a CIS data exchange file using either unit system) shall allow the user to select which unit system is to be used when writing a CIS data exchange file. (The selection shall be recorded in the Log File.)
15. An import translator with multiple unit system capabilities (i.e. one that has the ability to import data from a CIS data exchange file that was written using either unit system) shall automatically detect which unit system has been used within a particular CIS data exchange file and act accordingly.
16. If a user attempts to import a CIS data exchange file that uses a unit system that is not supported by that translator, the translator shall give a meaningful error message and stop.
17. An export translator will normally export a particular combination of conformance classes. However, it may provide the option of specifying the export of other (typically but not necessarily lesser) combinations of conformance classes. Where the user attempts to export a combination of conformance classes for which the application is not sufficiently populated with data to allow export, the user must be alerted to this fact. In this situation the user may be offered the option of proceeding with a lesser combination of conformance classes. Irrespective of this, the log file must always record which combination of conformance classes was actually exported.
18. An import translator will normally import a particular combination of conformance classes. However, it may provide the option of specifying the import of other (typically but not necessarily lesser) combinations of conformance classes. Where the user attempts to import a combination of conformance class that is not populated within the physical file, the user must be alerted to this fact. In this situation the user may be offered the option of proceeding with a lesser combination of conformance classes. Irrespective of this, the log file must always record which combination of conformance classes was actually imported.
19. An export translator shall have the ability to consider what data is currently held within the application and determine which Conformance Classes or CC Combinations can be exported. An export translator shall also assist the user to select appropriate data for export, giving guidance on the extent of any data that will not be exported.
20. An import translator shall have the ability to consider what CIS data is instanced, and thus determine which of the Conformance Classes or CC Combinations can be imported. An import translator shall also assist the user to select appropriate data for import, giving guidance on the extent of any data that will not be imported.

Commentary

Translator operation

A CIS translator should be activated by a simple command such as ‘CIMsteel Export’ (to trigger the export of all supported CCs) or ‘CIMsteel Import’ (to trigger the import of all supported CCs from a CIS file). Where appropriate, optional sub-commands should be provided to allow the selective export or import of particular CCs, CC combinations, Flavours, or unit systems.

Conformance Classes

The combinations of entities that an application is able to export or import depend upon the lists of combinations of Conformance Classes that the translator supports. These lists are detailed in the capability file.

Attribute values on export

The value of an attribute will normally be derived by the export translator from the internal data structures of the application using a mapping defined by the software vendor. Depending upon the scope of the application, this may mean that the export process is limited because the translator is unable to assign the full range of values of attributes that have ENUMERATION data types. (Where this occurs, this must be described fully in the documentation supplied with the CIS/2-conformant system.) Exceptionally, an attribute may be given a value by:

- The translator itself (for the HEADER section data).
- The user at the time of export (when prompted by the translator).
- The import translator of the CIS/2-conformant system (i.e. data passes through the system unprocessed).
- The software vendor shall ensure that such values are completely consistent with the totality of the exported information.

A default value built into the translator in order to overcome the limitations of the application would not normally be regarded as a valid source. Default values may be used where information is ‘hard-wired’ into an application, or where information is held implicitly rather than explicitly. For example, the unit system used within an application is often implicit rather than explicit, with the result that physical quantities do not have specified units in the application. In such a case, the translator may legitimately populate an export file with default values for the appropriate units. The software vendor shall ensure that the default values provided are appropriate to the data exported.

Attribute values on Import

The value of an attribute will normally be mapped by an import translator into the internal data structures of the application using a mapping defined by the software vendor. Depending upon the scope of the application, this may mean that the import process is limited because the translator is unable to map the full range of values of attributes that have ENUMERATION data types. (Where this occurs, this must be described fully in the documentation supplied with the CIS/2-conformant system.) Exceptionally, a particular attribute may be dealt with in another way. Potentially valid actions on import include:

- Displaying the attribute value (for example, the HEADER data) directly to the user and recording it within the Log File.
- Assigning the attribute value to data that is held implicitly by the application rather than explicitly within the data structures (e.g. the unit system used).
- Making the attribute value available to the CIS export translator for subsequent export (i.e. data passes through the system unprocessed).

Single step activation

A Basic CIS Translator should be seen as part of the source application, rather than as a separate application. Thus, CIS translators are required to be activated by a single menu

command or icon within the source application. For example, ‘CIS Export’ to trigger the export translator, and ‘CIS Import’ to trigger the import translator.

Where appropriate, additional commands may be provided to allow the selective export or import of particular information. For example, where the translator supports standard or proprietary references, the user should be offered the option to *‘pass by reference’*, or *‘pass by attribute value’*, or both. Further options regarding unit systems should also be offered where appropriate.

3.5.3 Documentation requirements for Basic CIS Translators

The following documentation shall be supplied in English to the engineering end user of the CIS/2-conformant system and to the Conformance Testing body. This documentation should distinguish clearly between import and export operations, and shall include:

1. The Vendor’s Based-on Statement.
2. A named contact point for user-support relating to the translator.
3. The name, version and release date of the application that is supported.
4. The identity of the version and release date of the translator.
5. The release (or releases) of the CIS that is (are) supported.
6. The Unit System(s) that is (are) supported.
7. The Combinations of CCs that are supported.
8. A statement of the Standard Products that are supported, (as defined by the ‘Flavours’ or structured lists of Standard Product References).
9. A statement of the Proprietary Products that are supported, (as defined by a structured list of the Proprietary Product References).
10. A statement of any relevant limitations of the CIS/2-conformant system.
11. A structured list of the entities and attributes that the translator is capable of reading and correctly importing into its application (for an import translator).
12. A structured list of the entities and attributes that the translator is capable of creating and assigning meaningful values (for an export translator).

The documentation shall also explicitly identify all situations where the translator may:

- create entity instances with attribute values obtained from direct input by the user as part of the export process (i.e. values that are not stored within the application or derived from the application’s data structures, but populated by the user during the translation process)
- create entity instances with default attribute values (i.e. values that are not stored within the application or derived from the application’s data structures, but populated automatically by the translator itself)

- be unable to export the full range of values for a particular attribute
- be unable to export a value for an attribute (whether OPTIONAL or not)
- be unable to import the full range of values for a particular attribute
- be unable to import an attribute of a particular entity.

To facilitate Conformance Testing, the software vendor shall supply the testing body (on a commercially confidential basis) with details of:

- The internal data structure of the application(s) supported.
- The data mappings implemented by the translator.

Commentary

The statement of any relevant limitations of the CIS/2-conformant system statement needs to distinguish clearly between limitations of the application, limitations of the translator, and limitations of the CIS¹. It should present these limitations in the order of their significance, and describe each limitation in engineering terms followed by any necessary technical detail.

Where the support of a ‘Flavour’ is incomplete, the documentation of the translator should identify those items that are not supported, stating the reason for their exclusion.

If an application supports any product references that are not supported by the translator this should be fully, and explicitly, documented.

Where the scope of the application is such that its translator is unable to import all possible values of an attribute that has an ENUMERATION data type, this should be fully documented.

3.6 Requirements for DMC Translators

The requirements of DMC Translators given below are *additional* to those of Basic CIS Translators described in Section 3.5. Some of the restrictions on Basic Translators are lifted for DMC Translators.

3.6.1 Implementation requirements for DMC Translators

1. A DMC export translator shall be able to create new unique identifiers for the engineering information that its associated application creates. Once created, the DMC Translator shall not assign the same unique identifier to any other data that it processes.
2. A CIS/2-conformant system with both an import and an export translator shall be able to maintain existing unique identifiers for the engineering information that it imports from other DMC applications and shall be able to make those unique identifiers available to the DMC export translator.

¹ Although CIS/2 is extensive, some applications will contain data that lies beyond its scope.

3. If ‘unmanaged data’ is received by a DMC import translator, the translator shall be able to convert that data into ‘managed data’ by assigning meta-data to that data as if the DMC application had created the data itself.
4. The 2nd Edition of CIS/2 introduces the concept of minimal data management, which in effect, is simply the association of an identifier with the data item, rather than maintenance of the history of the data collection. For minimal data management, the unique identifier shall be the ‘instance_id’, which is defined as alphanumeric identifier that is unique throughout the software world. This identifier is a string encoding of a globally unique unsigned 128bit integer and is also known as a Globally Unique Identifier (GUID) or a Universal Unique Identifier (UUID). Algorithms for generating the 128bit integer and for encoding this integer in a string representation are defined in documentation published by ISO^[19] and by the Open Group^[21]. The string representation consists of a sequence of hexadecimal fields separated by single dashes totalling 36 characters. An example of the globally_unique_id is the string “1A81FD96-D7A8-47d3-8DF7-BEEA6FF45184”.

The Microsoft Foundation Class (MFC) function "CoCreateGuid", which is an implementation of the referenced algorithms, may be used by CIS/2 implementers on Windows platforms to create this identifier (as a GUID). (See the Microsoft Online Documentation^[20].) Other implementations are available as UUID generators written in Java and other languages for use on MacOS and the various Unix platforms.

5. Where the data management requirement involves capturing and maintaining the history of the data collection then the GUID needs to be placed into the wider CIS identification context. The unique identifier shall then be made up from three components: the ‘application_id’, the ‘installation_id’ and the ‘instance_id’.
 - The application_id shall be unique to an application and its vendor and is intended to identify the software vendor, and the software application. To ensure its uniqueness within the domain of CIS/2 data sharing, the application_id shall be registered with the **CIS/2 International Technical Committee (ITC)** before it is used. Before developing a DMC translator, a software vendor shall apply to the **CIS/2 ITC** to obtain their registered application_id.
 - The installation_id shall be unique to one installation of a particular application and is intended to identify the ‘registered keeper’ of the software. To ensure its uniqueness within the domain of the particular software application, the software vendor shall assign and maintain the installation_id for each installation of each version of each DMC-application.
 - The instance_id shall be unique to an instance created by a particular implementation of an application, and is intended to identify the instance of data. To ensure its uniqueness within the domain of the particular CIS implementation, the DMC Translator shall assign and maintain these instance_ids.

For the 2nd Edition – the underlying data type of the attribute instance_id has been changed from an INTEGER to a STRING. For upward compatibility, an instance_id written in accordance with the first edition of CIS/2 needs to be converted from an INTEGER to a STRING.

6. Once created, the unique identifier shall be maintained by all DMC Translators.

Commentary

‘DMC Translators’ & ‘DMC Applications’

A translator written for a CIS/2-conformant system that has data management capabilities is known as a DMC Translator. The extent of the data management functionality will depend upon the capabilities of the underlying application for which the translator is written; i.e. whether the native application can manage data. It will also depend upon whether the system has both an import translator and an export translator. Where a native application can manage data and is provided with both an import translator and an export translator it shall be referred to as a DMC application. This is discussed in more detail in Section 6 of the *Implementation Guide*^[3]

Meta-data & data management

In CIS/2, data management is considered to be the process of assigning meta-data to data. In simple terms, ‘Data Management Conformant’ (DMC) applications produce ‘managed data’ such that each item of data is given a unique identification and is placed in a data set, whereas Non-DMC applications will produce ‘unmanaged data’. Although several applications will contribute to and modify data instances, the unique identifier will be assigned by the application that originally created the meta-data to manage that data. Furthermore, ‘unmanaged data’ may be mapped onto meta-data to become ‘managed data’, using a DMC application.

By definition, a DMC translator is required to be capable of supporting all of the (appropriate) data management constructs described in *The Logical Product Model*^[5] and *The Implementation Guide*^[3], specified by the DMC Conformance Class supported. Of most concern to the writers of DMC-Translators are the two entities ‘managed_data_item’ and ‘managed_data_transaction’, which form the essential EXPRESS constructs in the LPM for data management by capturing the meta-data.

A ‘managed_data_item’ represents an individual item of managed data and allows that data item to be assigned a unique identifier, and (optionally) a history. A ‘managed_data_transaction’ is simply a record of the use of a DMC application (e.g. an export or import process). A ‘managed_data_transaction’ collects together managed data, and assigns the additional ‘meta-data’ that is associated with each and every item of data in the collection.

The number of data entity constructs (used to represent engineering information) that a DMC translator supports will be dependent upon the scope of the application for which the DMC translator is written. The creation and processing of these data entity constructs is handled in the same way as those for a Basic CIS Translator.

Minimal data management

The 2nd Edition of CIS/2 introduces the concept of minimal data management, which in effect, is simply the association of an identifier with the data item, rather than maintenance of the history of the data collection. This reduces the role of the entity managed_application_installation in data management. That is, where all that is required is the passing of a GUID then managed_application_installation will not be instanced. However, for more advanced data management, the managed_application_installation entity becomes mandatory and its instances will place the GUID into the wider CIS identification context. This means that all CIS/2 translators (other than Basic Translators) are required to support the entity managed_application_installation, even though it will remain unpopulated for minimal data management.

Inter-working between DMC and Non-DMC applications

If ‘unmanaged data’ is converted into ‘managed data’ by a DMC-Translator, the unique identifier will be determined by the application’s DMC-Translator, and will reflect the name of DMC application that managed the data rather than the application that actually created the data.

Unit Systems

An export translator with multiple unit system capabilities should have the ability to export data to a CIS data exchange file using either of the two ‘standardized’ unit systems (i.e. SI, US Imperial). Such an export translator should allow the user to select which unit system is to be used when writing a CIS data exchange file.

While DMC Translators are required to support either (or both) of the ‘standardized’ unit systems when handling data representing physical quantities, they may also support other unit systems in addition to the either (or both) of the ‘standardized’ unit systems. (This lifts the restriction placed on Basic CIS Translators.)

The Scope Structure

DMC Translators may use the Scope Structure. (This lifts the restriction placed on Basic CIS Translators.)

3.6.2 Operational requirements for DMC Translators

1. When exporting data from a DMC application, the DMC translator shall give the user the choice of either exporting ‘managed data’ or ‘unmanaged data’. (If the latter option is selected, the DMC export translator operates as a Basic CIS export translator and meta-data will not be created.)
2. When importing managed data into a DMC application, the DMC Translator shall maintain the meta-data associated with the engineering data.
3. When importing unmanaged data into a DMC application, the user shall be given the option to create meta-data for the unmanaged data. (If the user chooses to manage the data, the meta-data will be dependent upon the DMC translator rather than the application that created the data. If the user chooses not to manage the data, the DMC Translator acts as a Basic Translator, and simply imports unmanaged data into the application).
4. Where the DMC application is not able to comply with all the implementation requirements for all of the relevant data entity constructs that are used to represent the engineering information created and processed by the application, appropriate warnings shall be displayed to the user during the translation process and recorded in the Log File.
5. The ‘Data Management Tool’ of the DMC Translator shall contribute section 5 of the file, giving details of the management of the data that is being exchanged. Examples are shown in Section 3.11.

Commentary***Managed and unmanaged data***

When operating as a DMC translator, data will be managed, and have appropriate meta-data assigned to it. If the user has chosen not to manage the data, the DMC translator is acting as a Basic CIS Translator. A CIS Translator should never attempt to export a mixture of managed and unmanaged data. Moreover, a particular physical file should not contain a mixture of managed and unmanaged data.

3.6.3 Documentation requirements for DMC Translators

The documentation supplied with a DMC Translator shall include the information required for a Basic CIS Translator, together with:

1. Detailed documentation of its data management capabilities
2. The ‘application-id’ (which shall be that appropriate to its data management operations)
3. A statement of any limitations of the DMC Translator’s data management capabilities including:
 - Whether it allows the user to turn unmanaged data into managed data
 - Whether it can manage all of the engineering data that it exports or imports (i.e. whether it can assign meta-data to the engineering data created by the associated application and maintain that meta-data)
 - Whether it can maintain all of the meta-data that may be assigned to the engineering data (via the managed_data_transaction entity).
 - Whether it can support the scope structure described in clause 10.3 of Part 21.

3.7 Requirements for IDI Translators

The requirements for IDI Translators given below are *additional* to those of Basic CIS Import Translators described in Section 3.4.3 and those of DMC Import Translators described in Section 3.6.

3.7.1 Implementation requirements for IDI Translators

1. An IDI Translator shall be able to maintain the unique identifiers for every managed entity instance during an incremental translation process.
2. An IDI translator shall be able to support either (or both) of the two scenarios for ‘Incremental Data Import’ envisaged by *The Implementation Guide*^[3] (i.e. the ‘master-slave’ scenario or the ‘feedback’ scenario).

Commentary

As explained in the *Implementation Guide*, ‘Incremental Data Import’ is the ability of a software application to add new information to an existing information set within the application while maintaining the integrity of the information set. The purpose of the IDI translator is to import only the differences between new data and old data. To do this, it must be able to ‘recognize’ incoming data instances as modified versions of existing data instances. Because this process is dependent upon the unique identifiers provided for

every instance of every data entity, an IDI Translator must be able to maintain the unique identifiers for every managed entity instance during an incremental translation process.

3.7.2 Operational requirements for IDI Translators

1. The ‘Comparator Tool’ of the IDI Translator shall contribute sections 6a and 7a to the file, giving details of the Comparator tool operation and how the data was imported into the application. (An example is shown in Section 3.11.)

3.7.3 Documentation requirements for IDI Translators

There are no *additional* documentation requirements for IDI Translators. The documentation supplied with an IDI Translator is required to include all of the information supplied with a Basic CIS Translator, together with the detailed documentation of its data management capabilities.

3.8 Requirements for PMR-Enabled Translators

The requirements of PMR-Enabled Translators are given below. A PMR-Enabled Translator is a different type of translator (in that it interacts with a PMR, where data is held in a neutral format). Nevertheless, the requirements for a PMR-enabled translator include those requirements given in Sections 3.5 and 3.6 for Basic and DMC Translators, plus, if appropriate, the requirements in Section 3.7 for IDI Translators.

3.8.1 Implementation requirements for PMR-Enabled Translators

Additionally to the requirements given in Sections 3.5.1, 3.6.1, and (if appropriate) 3.7.1, the PMR-Enabled Translators:

1. shall be able to use both STEP implementation level 3 (i.e. data sharing via a DBMS) and level 1 (i.e. physical file data sharing).

Commentary

In simple terms, a PMR-Enabled Translator is one that has either DMC export capabilities, or IDI import capabilities, or both. The PMR-Enabled Translator has the additional capability of being able to log on directly to a suitable PMR. The data stored in a PMR may be accessed either ‘directly’ or ‘indirectly’. Direct access requires the use of SDAI calls, while indirect access makes use of physical files. It is a prerequisite for direct access that both the PMR-Enabled Translator and the PMR use the same implementation of SDAI. In practice, this currently means that PMR-Enabled Translator has to be written using the same STEP developer’s toolkit as was used to write the PMR. Thus, PMR-Enabled Translators need to be developed in conjunction with a PMR. Even if the PMR is written by a third party software developer, developers of PMR-Enabled Translators should liaise closely with these third parties.

The fundamental capabilities of a PMR-Enabled Export Translator are the same as those of a DMC Export Translator. As shown in Figure 3.1, when exporting data from an application to a PMR, the comparison between the data from the application and the data held in the repository of the PMR is performed by the Comparator Tool **of the PMR**.

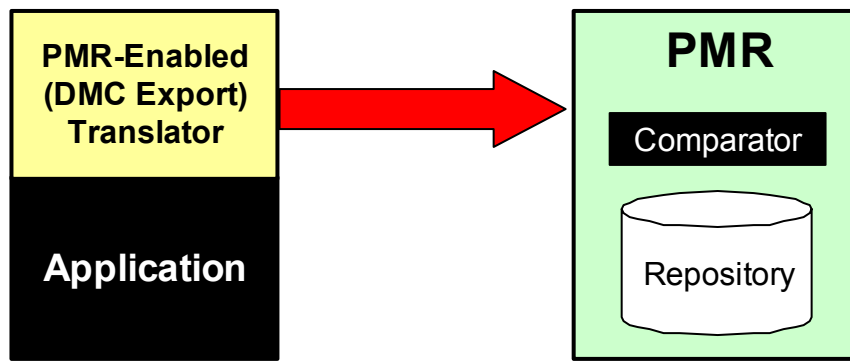


Figure 3.1 *PMR-Enabled Translator Exporting Data to a PMR*

The fundamental capabilities of a PMR-Enabled Import Translator are the same as those of an IDI Translator. As shown in Figure 3.2, when importing data from a PMR into an application, the comparison between the data from the PMR repository and the data exiting in the application is performed by the Comparator Tool of the **PMR-Enabled Import Translator**.

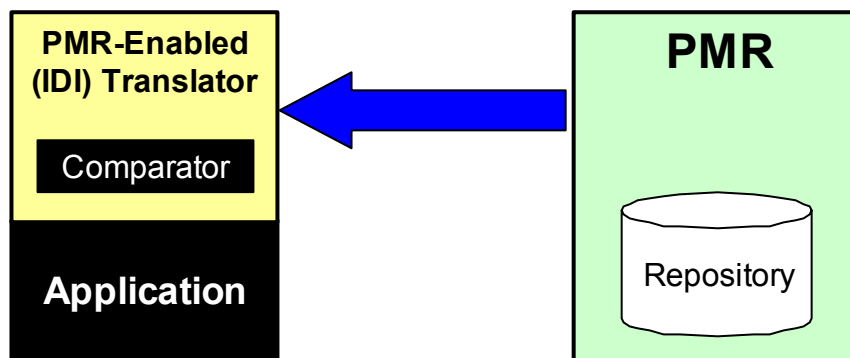


Figure 3.2 *PMR-Enabled Translator importing data from a PMR*

In addition to these fundamental capabilities, all PMR-Enabled Translators are required to be capable of inter-working with at least one specific PMR using direct access; i.e. a STEP implementation level 3. PMR-Enabled Translators should also be able to inter-work with the specified PMR using indirect access; i.e. a STEP implementation level 1.

PMR-Enabled Translators may also allow users to connect to other PMRs using either direct or indirect access. The network protocols for access control and transportation that are used with PMR-Enabled Translators are left to the discretion of the software vendor.

3.8.2 Operational requirements for PMR-Enabled translators

Additionally to the requirements given in Sections 3.5.2, 3.6.2, and (if appropriate) 3.7.2, a PMR-Enabled translator shall:

1. Operate with at least one PMR for which the PMR-Enabled translator has been written.
2. Operate seamlessly with an appropriate PMR and be viewed as part of the underlying application.

3. Allow data to be imported from the PMR into the application and/or exported from the application into the PMR.
4. Operate from a single menu command or toolbar icon from within the application. (An example of an interface for a PMR-Enabled Translator is shown in *The Implementation Guide*.)
5. Offer the user the choice of either performing direct data transfer (via SDAI calls) or indirect data transfer (via physical files).
6. Contribute Sections 6b and 7b to the Log File, giving details of the Comparator tool operations.

Commentary

When exporting data from an application to a PMR, the ‘Comparator Tool’ **of the PMR** is used to populate Sections 6b and 7b. Here, the PMR-Enabled Translator is acting as a DMC Translator and the PMR is acting as an IDI Translator. When importing data from a PMR into an application, the ‘Comparator Tool’ **of the PMR-Enabled Translator** is used to populate Sections 6b and 7b. Here, the PMR-Enabled Translator is acting as an IDI Translator.

3.8.3 Documentation requirements for PMR-Enabled Translator

There are no *additional* documentation requirements for PMR-Enabled Translator. The documentation supplied with a PMR-Enabled Translator shall include all the information supplied for a Basic CIS Translator, together with the detailed documentation of its data management capabilities.

3.9 Requirements for PMRs

The requirements of a Product Model Repository (PMR) are given below.

A PMR should be recognized as a different type of CIS/2 implementation, in that it is not simply a translator, but an application in its own right. In simple terms, it is a database management system with its underlying data model defined by LPM/6. Thus, the data held in a PMR is in the neutral format. It is also expected to operate (either directly or indirectly) with a number of other specialist applications; sharing engineering data via the CIS translators written for those applications. Nevertheless, the requirements for a PMR include all those requirements given in Sections 3.5 and 3.6 (for Basic and DMC Translators) and plus, where appropriate, the requirements given in Sections 3.7 & 3.8 (for IDI and PMR-Enabled Translators).

3.9.1 Implementation requirements for PMRs

1. A PMR is required to be able to use both STEP implementation level 1 (i.e. physical file data sharing) and STEP implementation level 3 (i.e. data sharing via a DBMS).
2. The developer of a PMR shall provide the functionality of all of the components required by a PMR, as shown in Figure 3.3.

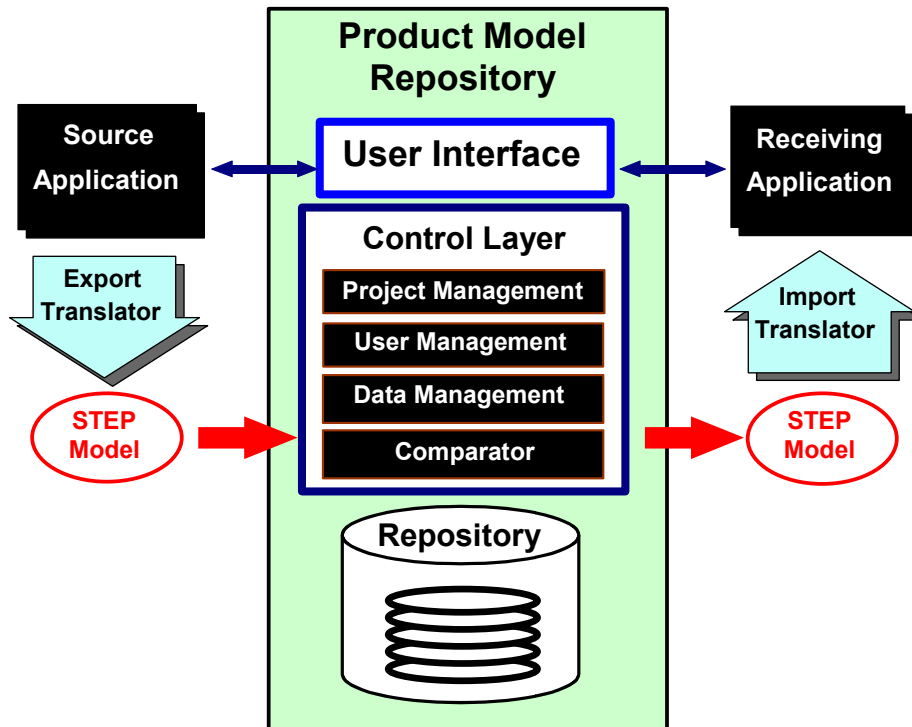


Figure 3.3 *Components of a PMR*

Commentary

The components of a PMR and how they are used are described in Section 9 of the *Implementation Guide*^[3].

Other mechanisms for communication beyond those offered by STEP may also be considered.

3.9.2 Operational requirements for a PMR

1. The PMR shall allow the user to perform the full range of tasks detailed in Section 9 of the *Implementation Guide*^[3].
2. The PMR shall provide its user with the option to operate the PMR either in 'stand-alone' mode or 'integrated' mode. ('Integrated' operation is only possible via a PMR-Enabled translator.)
3. The PMR shall provide its user with the option to access data either directly (via SDAI calls) or indirectly (via physical files). (Direct access is only possible when the PMR is operated in 'integrated mode'.)
4. A PMR shall be able to generate two types of log files:
 - Translation log files - similar in content and format to those produced for PMR-enabled translators, and
 - Project log files – detailing the creation and management of data related to particular projects.
5. The details of all operations of the PMR shall be recorded in the appropriate PMR Project log file. One Project log file shall be created for each Project contained

within the PMR. The Project log file shall be created and maintained as an ASCII text file. Although it is not normally seen during the operation of the PMR, it shall be made available for subsequent inspection and/or printing by the user.

6. For each Project, the Project log file records:

- The name of the Project.
- The description of the Project.
- The current status of the Project as either 'New', 'Modified', or 'Deleted'.
- When the Project was created, and by whom.
- When the Project was last modified, and by whom.
- When the Project was deleted, and by whom.
- The 'Project Manager' for the Project.
- The list of current users with their status.
- The list of STEP Models held for that Project.

And for each STEP Model:

- The current status of the Model as either 'New', 'Modified', or 'Deleted'.
- When it was created, and by whom.
- When it was last opened, and by whom.
- When it was last updated, and by whom.
- When it was deleted and by whom.
- Its status as either 'Locked or Unlocked'.

Commentary

A PMR-Enabled Translator is used to export data from an application into a PMR or to import data from a PMR into an application. When exporting data from an application into a PMR (using any of the 'Create Model', 'Replace Model', or 'Update Model' commands) the **PMR-Enabled Translator** creates an Export Log File and the **PMR** creates an import log file. Here, the PMR-Enabled Translator is acting as a DMC Translator and the PMR is acting as an IDI Translator. Thus, the 'Comparator Tool' of the **PMR** is used to populate Sections 6b and 7b of the Log File.

When importing data from a PMR into an application (using the 'Open Model' command) the **PMR-Enabled Translator** creates an Import Log File and the **PMR** creates an export log file. Here, the PMR-Enabled Translator is acting as an IDI Translator, and the 'Comparator Tool' of the **PMR-Enabled Translator** is used to populate Sections 6b and 7b.

In addition to the translation Log Files, the PMR creates and maintains a number of Project Log Files. Data is held in the PMR under a relevant 'Project'. The 'Project' is simply a convenient way of dividing up the storage area of the PMR so that data can be classified. The PMR can then be used to access and control the data associated with the 'Project', subject to the imposing rules by the 'Project Manager'. Every time the PMR is used to access data within a Project, the Project Log File is updated. The details of

the Project Log File are used by various components of the PMR, and shall be displayed to the user during operation of the PMR.

3.9.3 Documentation requirements for a PMR

1. A PMR should be seen as an application, and as such, it shall be provided with appropriate documentation (i.e. a complete User Manual).
2. A PMR should also be seen as an application with CIS translators embedded within it. Therefore, in addition to the documentation produced for the PMR as an application, the developer of a PMR shall also produce detailed documentation of its data management capabilities. This shall include the ‘application-id’ that the PMR will use for its data management operations.
3. The documentation shall also state any limitations of the PMR’s capabilities including:
 - The scope of the data that may be stored in a PMR (if not the entire LPM/6 schema).
 - The scope of the data that may be managed by the PMR (if not the entire LPM/6 schema).
 - Whether it can turn unmanaged data into managed data.
 - Whether it can manage all of the engineering data that it holds.
 - Whether it can maintain all of the meta-data that may be assigned to the engineering data (via the managed_data_transaction entity).

3.10 Details of the ‘Capability Files’

As explained earlier, all CIS Translators are required to be able to generate an ASCII file that describes the capabilities of the translator for import or export operations. For an export translator, the file shall be named CIS_OUT.TXT, and it shall describe the scope of the information that the translator supports on export from the application (into a physical file). For an import translator, the file shall be named CIS_IN.TXT and shall describe the scope of the information that the translator supports on import into the application (from a physical file).

3.10.1 Capability File syntax

The syntax of the capability file is illustrated in the example CIS_OUT.TXT file shown in Figure 3.4. Each line starts with the record name, which is followed by an equals (=) separator. Where a record contains more than one field, each field is separated by a comma (.). The end of each record is signified by a semi-colon (;). The file begins with a “capability_file” statement and is terminated by a “end_of_capability_file” statement.

```

capability_file =      CIS_OUT.txt
application_id =       1213456;
application_name =     My Application;
application_version =  1b;
application_description = Steel design software;
application_vendor =   ABC plc;
installation_id =      987654;
installation_name =    abcdef123456;
installation_owner =   XYZ ltd;
translator_version =   3.0c;
translator_date =      2003-05-12;
translator_type =      DMC;
cis_release =          2.1;
object_id =            cimsteel logical product model version (6) object (1)
                      structural-frame-schema (1)
flavours =             us_metric_sections.stp, us_metric_materials.stp,
                      my_cold_bits.stp;
unit_systems =         SI;
conformance_classes =  CC009, CC029, CC110, CC118, CC170, CC177,
                      CC180, CC249, CC325, CC331;
entity_list =          global_unit_assigned_context, si_unit,
                      assembly_design_structural_member_linear,
                      design_part, assembly_map, element_curve_simple,
                      element_mapping, part_prismatic_simple,
                      section_profile, item_reference_standard;
end_of_capability_file

```

Figure 3.4 *Example of a CIS_OUT.TXT file (CIS_IN.TXT similar)*

3.10.2 CIS/2 Capability File contents

As can be seen in Table 3.1, a CIS capability file contains 20 consecutive records, with one record per line of the text file. All translators are required to produce all 20 records, although some of records may contain blank fields, depending on the type of CIS implementation.

The first record opens the capability file, and states whether the file is describing the capability of an export translator (CIS_OUT.txt) or an import translator (CIS_IN.txt).

The records 2 to 9 relate to the attributes of the LPM/6 entity managed_application_installation. It is a Conformance Requirement of all but Basic Translators to support this entity. Thus, for these types of translator, the non-optional fields of these records will be populated in the capability file. Basic CIS Translators should populate the fields of records 3 and 4, and (optionally) records 5, 6, and 9. (Records 2, 7, and 8 relate to data management functionality beyond the scope of Basic Translators.)

Table 3.1 *Records required within a CIS/2 capability file*

Record	Record name	Example field	Field format	Required
1	capability_file	CIS_OUT.txt	filename	All
2	application_id	1213456	INTEGER	DMC & above
3	application_name	My Application	STRING	All
4	application_version	1b	STRING	All
5	application_description	Steel design software	STRING	Optional
6	application_vendor	ABC plc	STRING	Optional
7	installation_id	987654	INTEGER	DMC & above
8	installation_name	abcdef123456	STRING	DMC & above
9	installation_owner	XYZ ltd	STRING	Optional
10	translator_version	3.0c	STRING	All
11	translator_date	2003-05-12	date yyyy-mm-dd	All
12	translator_type	DMC	STRING	All
13	cis_release	2.1	SET OF STRINGS	All
14	object_id	cimsteel logical product model version (6) object (1) structural-frame- schema (1)	STRING	All
15	implementation level	1, 3	SET OF INTEGERS	All
16	flavours	us_metric_sections.stp, us_metric_materials.stp, my_cold_bits.stp	SET OF STRINGS (filenames)	All
17	unit_systems	si	SET OF STRINGS	All
18	conformance_classes	CC009, CC029, CC110, CC118, CC170, CC177, CC180, CC249, CC325, CC331	SET OF STRINGS (CC numbers)	All
19	entity_list	global_unit_assigned_co ntext, si_unit, assembly_design_struct ural_member_linear, design_part, assembly_map, element_curve_simple, element_mapping, part_prismatic_simple, section_profile, item_reference_standard	SET OF STRINGS (entity names)	All
20	end_of_capability_file			All

Records 10, 11, and 12 relate to the particular translator and must be populated by all translators. Record 12 should contain a value of either “BASIC”, “DMC”, “IDI”, “PMR-ENABLED”, or “PMR”, depending on the type of CIS translator (or CIS-conformant system) implemented.

Records 13 and 14 refer to the release (or releases) of the CIS supported by the translator. The `object_id` is intended to provide an unambiguous reference to the version(s) of underlying schema supported by the translator.

Record 15 refers to the STEP implementation level (discussed in Section 3.1) supported by the translator. It will be a list of at least one integer. For example, the field will be populated with a value of “1”, if the translator supports physical file exchange.

Record 16 refers to the list of ‘flavour files’ supported by the translator. The list of ‘flavour files’ quoted in the capability file may include both proprietary product items and library items as well as standard product items. (See Section 5.4 for the details of how CIS/2 deals with these product items.) If a translator does not support any flavours, record 15 will contain a blank field.

It should be noted that an export translator must be able to create all of the product references specified in the flavours quoted in the capability file. For Level 1 Implementations, this means that the export translator must be able to create each and every instance of the entity `item_reference` contained in the physical file quoted.

Similarly, an import translator must be able to read all of the product references specified in the flavours quoted in the capability file. It must, therefore, be able to read each and every instance of `item_reference` contained in the physical file quoted.

Record 17 refers to the unit system(s) supported by the translator. In most cases, this will be either “SI”, or “US Imperial”. (See Section 5.3 for the details of how CIS/2 deals with unit systems.) If a translator does not support any units system (because the underlying application does not deal with data concerning physical quantities), record 16 will contain a blank field.

Record 18 refers to the list of Conformance Classes (CCs) or CC Combinations supported by the translator. This record must be populated; a blank field in this record would indicate a non-conformant translator. Where individual CCs are supported, these are presented as a comma separated list; e.g. “CC105, CC206”. Where individual CCs can only be supported in combination with other CCs, the CC Combination are presented as a union of the individual CCs; e.g. “(CC105+CC206)”. Record 18 may be populated with a mixture of individual and combined CCs; e.g. “(CC105+CC206), CC206”. (See Section 2 for details of the CIS/2 Conformance Classes.)

Record 19 presents a comma-separated list of the entities supported by the translator. Each entity name should appear only once in the list. It follows that an export translator must be able to create instances of all of the entities quoted in the entity list in record 19. Similarly, an import translator must be able to read instances of the entities quoted in the entity list.

Finally, the capability file is closed by the token “`end_of_capability_file`” in record 20.

3.11 Details of the Log Files

As explained earlier, all CIS Translators are required to generate a log file automatically when exporting or importing data. The requirements for the log files in terms of the sections supported are summarized in Table 3.2

As can be seen from Table 3.2, the type of log file produced depends upon the STEP implementation level chosen and the degree of complexity of the CIS translator.

A CIS log file includes up to seven different consecutive sections. All translators are required to produce the first four sections of the log file, while only the advanced implementations will produce all seven sections.

The detailed requirements of each section of the log file are given in Table 3.3 through to Table 3.18, inclusive. Software vendors developing CIS translators are strongly recommended to examine the details of the log files very carefully, as the tables show the functionality and error messages required at each level of implementation.

It should also be remembered that it is a Conformance Requirement of CIS/2 that the 'log file' be displayed incrementally on the screen as the translation proceeds, and be written as an ASCII text file for subsequent inspection and/or printing by the user.

Table 3.2 *Log File Sections required for various types of implementation*

Section	Type of Implementaion	Details	Details given in table:	
			Export	Import
1a	Physical file (Level 1)	STEP File information	Table 3.3	Table 3.4
1b	In-memory (Level 2)	STEP Data information		
1c	DBMS (Levels 3) KBS (Levels 4)	STEP Model information		
2	All	CIS data conformance details	Table 3.5	Table 3.6
3	All	Application export log CIS data reading log	Table 3.7	Table 3.8
4	All	CIS data writing log Application data import log	Table 3.9	Table 3.10
5	DMC, IDI, PMR-enabled Translators, and PMRs	Data Management details	Table 3.11	Table 3.12
6a	IDI Translators	Operational Details	Table 3.13	Table 3.14
6b	PMR-enabled / PMR			Table 3.15
7a	IDI Translators	Data Comparison & Updating details	Table 3.16	Table 3.17
7b	PMR-enabled / PMR			Table 3.18

Notes to Tables 3.3 to 3.18

- i. The detailed requirements for each of section of the Log File are given in Table 3.3 though to Table 3.18, inclusive.
- ii. In Table 3.3 though to Table 3.18, one line of (unformatted ASCII) text is required for each bullet point.
- iii. Where the CIS Translator inserts a variable name, this is denoted with the symbols `< >`.
- iv. The name of an attribute is denoted in *italics* between the `< >` symbols; i.e. `<attribute name>`
- v. The name of an entity is denoted in **bold** between the `< >` symbols; i.e. `<entity name>`
- vi. `<entity name>.<attribute name>` denotes an entity-attribute construct.
- vii. Where the lines of the log file are optional, they are preceded by the word 'OPTIONAL'.
- viii. Where there are alternatives (e.g. on error), these are introduced by the word 'EITHER' and 'OR'.
- ix. Comments are shown in *italics*
- x. Messages to display on-screen during the translation process are numbered sequentially and are shown in "quotes". They are also cross-referenced to the Log File text by their reference number.

Table 3.3 *Section 1 of the Export Log File*

SECTION 1a: CIS data exchange file information <i>(Version used with physical file implementations – data values written to in HEADER section)</i>
Write to log file
<input type="checkbox"/> File exported as filename: <file_name.name> <input type="checkbox"/> File created on <file_name.time_stamp> <input type="checkbox"/> File created by <file_name.author> <input type="checkbox"/> File created using version <file_name.preprocessor_version> of the translator <input type="checkbox"/> Data originated from <file_name.orginating_system> <input type="checkbox"/> Data created in accordance with <file_schema.schema_identifiers>
Message to display on-screen
<i>On completion of creation of STEP Part 21 physical file</i> 1. "Data exported to file <file_name.name> in accordance with <file_schema.schema_identifiers>"
SECTION 1b: CIS STEP data information <i>(Version used with in-memory implementations)</i>
Write to log file
<input type="checkbox"/> STEP data created on <date> at <time> <input type="checkbox"/> STEP data created by <name> <input type="checkbox"/> STEP data created using version <label> of the translator <input type="checkbox"/> Data originated from <system_name> <input type="checkbox"/> Data created in accordance with <schema_name>
SECTION 1c: CIS STEP model information <i>(Version used with DBMS implementations – data values written to translator's repository)</i>
Write to log file
<input type="checkbox"/> STEP model exported as: <model_name> <input type="checkbox"/> STEP model created on <date> at <time> <input type="checkbox"/> STEP model created by <name> <input type="checkbox"/> STEP model created using version <label> of the translator <input type="checkbox"/> Data originated from <system_name> <input type="checkbox"/> Data created in accordance with <schema_name> ²
Message to display on-screen
<i>On completion of STEP model creation in repository</i> 2. "Data exported to model <model_name> in accordance with <schema_name>"
<i>See page 35 for the Notes to Tables 3.3 to 3.18</i>

Table 3.4 *Section 1 of the Import Log File*

SECTION 1a: CIS data exchange file information
<i>(Version used with physical file implementations – data values extracted from HEADER section)</i>
Write to log file
<input type="checkbox"/> File exported as filename: <file_name.name> <input type="checkbox"/> File created on <file_name.time_stamp> <input type="checkbox"/> File created by <file_name.author> <input type="checkbox"/> File created using version <file_name.preprocessor_version> of the translator <input type="checkbox"/> Data originated from <file_name.originating_system> <input type="checkbox"/> Data created in accordance with <file_schema.schema_identifiers> ³
Message to display on-screen
<i>IF <file_schema.schema_identifiers> not equal to 'STRUCTURAL_FRAME_SCHEMA'</i> 3. "Error – the imported file contains a schema other than 'STRUCTURAL_FRAME_SCHEMA', do you wish to continue"

SECTION 1b: CIS STEP data information
<i>(Version used with in-memory implementations)</i>
Write to log file
<input type="checkbox"/> STEP data created by <user name> <i>(if available)</i> <input type="checkbox"/> STEP data created using version <label> of the translator <i>(if available)</i> <input type="checkbox"/> Data originated from <system_name> <i>(if available)</i> <input type="checkbox"/> Data created in accordance with <schema_name>

SECTION 1c: CIS STEP model information
<i>(Version used with DBMS implementations – data values extracted from translator's repository)</i>
Write to log file
<input type="checkbox"/> STEP model name: <model_name> <input type="checkbox"/> STEP model last accessed on <date> at <time> <input type="checkbox"/> STEP model created by <user name> <i>(if available)</i> <input type="checkbox"/> STEP model created using version <label> of the translator <i>(if available)</i> <input type="checkbox"/> Data originated from <system_name> <i>(if available)</i> <input type="checkbox"/> Data created in accordance with <schema_name> ⁴
Message to display on-screen
<i>IF <schema_name> not equal to 'STRUCTURAL_FRAME_SCHEMA'</i> 4. "Error – the imported model has been written to a schema other than 'STRUCTURAL_FRAME_SCHEMA', do you wish to continue"
<i>See page 35 for the Notes to Tables 3.3 to 3.18</i>

Table 3.5 *Section 2 of the Export Log File*

SECTION 2: CIS data conformance details	
Write to log file	
<input type="checkbox"/> Data conforms to release <number> of the CIS <input type="checkbox"/> Data conforming to CC(s) <List of numbers> exported <i>... i.e. the CC(s) actually populated</i> <input type="checkbox"/> Data conforming to the <label> unit system exported <i>... repeat as required, i.e. a line is added for every unit system populated. Thus, there will be two lines created when a translator that supports both SI and US Imperial units exports a file containing physical quantities measured in both sets of units</i> <input type="checkbox"/> Data conforming to the <label> flavour exported <i>... repeat for all flavours in file, i.e. a line is added for every flavour populated.</i>	
Notes <p>An export translator shall have the ability to consider what data is currently held within the application, and recognize that the data conforms with a particular set of Conformance Classes (or CC Combinations), flavours and unit systems, as appropriate. It shall also assist the user to select appropriate data for export, giving guidance on the extent of any data that will not be exported.</p>	

Table 3.6 *Section 2 of the Import Log File*

SECTION 2: CIS data conformance details	
Write to log file	
<input type="checkbox"/> Data conforming to CC(s) <numbers> identified <i>... i.e. the CC(s) actually encountered in incoming data</i> <input type="checkbox"/> Data conforming to the <label> unit system identified <i>... repeat as required for every unit system encountered in incoming data</i> <input type="checkbox"/> Data conforming to the <label> flavour identified <i>... repeat as required for every flavour encountered in incoming data</i>	<input type="checkbox"/> Warning – data appears not to conform to a unit system supported by this implementation ⁵ <input type="checkbox"/> Warning – data appears not to conform to a flavour supported by this implementation ⁵
Message to display on-screen	
5. “Warning – some of the incoming data is beyond the scope of this implementation - Do you wish to continue importing data”	
Notes <p>An import translator is expected to be able to recognize data that conforms with the CCs (or CC combinations), flavours and unit systems that it supports. It should do this by a detailed examination of the data instances it encounters. (The HEADER section of the physical file may also list the CCs that the exporting translator populated, but this information should not be relied upon as its existence is not a conformance requirement.) It shall assist the user to select appropriate data for import, giving guidance on the extend of any data that will not be imported. It shall also warn the user when it encounters data beyond its capability.</p>	
<i>See page 35 for the Notes to Tables 3.3 to 3.18</i>	

Table 3.7 *Section 3 of the Export Log File*

SECTION 3: Application export log	
Write to log file	
<ul style="list-style-type: none"> ❑ On exporting data from the application a total of <number> entity instances were processed and exported. ❑ Some application data has been ignored and not exported. ... if appropriate 	<ul style="list-style-type: none"> ❑ Warning - Data export may be incomplete⁶ EITHER <ul style="list-style-type: none"> - Data lies beyond capability of export translator OR <ul style="list-style-type: none"> - Data lies beyond scope of CIS/2
Message to display on-screen	
6. "Warning - Data export may be incomplete - Do you wish to continue"	

Table 3.8 *Section 3 of the Import Log File*

SECTION 3: CIS data reading log	
Write to Log File	
<ul style="list-style-type: none"> ❑ The Import Translator found a total of <number> entity instances. ❑ <number> instances of type <entity name> ... repeat for all entity types in incoming data. ❑ <number> instances of <entity name> .<attribute name> without a required value. ... repeat for each instance encountered ❑ <number> 'Standard References' of type <item_reference_standard.ref> ... repeat as required ❑ <number> 'Proprietary References' of type <item_reference_proprietary.ref> ... repeat as required ❑ <number> 'Library References' of type <item_reference_library.ref> ... repeat as required 	<ul style="list-style-type: none"> ❑ Warning - Type <entity name> not supported by this implementation repeat for every unsupported entity type encountered ❑ Warning - <attribute name> requires a value before <entity name> can be imported into the application⁷. repeat for each instance encountered ❑ Warning - <item_reference_standard.ref> not supported by this implementation⁸. repeat for every unsupported reference encountered ❑ Warning - <item_reference_proprietary.ref> not supported by this implementation⁸. repeat for every unsupported reference encountered ❑ Warning - <item_reference_library.ref> not supported by this implementation⁸. repeat for every unsupported reference encountered
Messages to display on-screen	
7. "Warning – <attribute name> requires a value before <entity name> can be imported into the application" EITHER – "Please input a substitute value" OR – "Do you wish to continue importing data"	
8. "Warning – reference <reference> not supported by this implementation" EITHER – "Would you like to accept the substitute value" if the translator has one to offer and display OR – "Do you wish to continue importing data"	
See page 35 for the Notes to Tables 3.3 to 3.18	

Table 3.9 Section 4 of the Export Log File

SECTION 4: CIS data writing log	
Write to log file	
<ul style="list-style-type: none"> ❑ The Export Translator created: ❑ a total of <number> entity instances. ❑ <number> instances of type <entity name> ... repeat for all entity types in file. ❑ <number> instances of <entity name> with a default value for <attribute name>. ... repeat for each entity type created ❑ <number> instances of <entity name> with substitute values for <attribute name> ... repeat as required ❑ <number> 'Standard Product Items' with 'Standard References' ❑ <number> 'Standard Product Items' as non-standard items ❑ <number> 'Manufacturer's Product Items' with 'Proprietary References' ❑ <number> 'Manufacturer's Product Items' as non-standard items ❑ <number> 'Library Product Items' with 'Library References' ❑ <number> 'Library Product Items' exported as non-standard items 	<ul style="list-style-type: none"> ❑ Warning - Data may have been modified from the original version contained in the application⁹ ❑ Warning - Data may have been modified from the original version contained in the application⁹ ❑ Warning - User has elected to pass data by: EITHER - reference only OR - both reference & attribute value⁹ ❑ Warning - <item name> not supported by this implementation – data passed by attribute value only⁹ ❑ Warning - User has elected to pass data by: EITHER - reference only⁹ OR - both reference & attribute value only ❑ Warning - <item name> not supported by this implementation – data passed by attribute value only⁹ ❑ Warning - User has elected to pass data by: EITHER - reference only⁹ OR - both reference & attribute value ❑ <item name> not supported by this implementation – data passed by attribute value only⁹
<ul style="list-style-type: none"> ❑ Summary Report of Export Process: Export of data from <application_name> to <file_name.name> / <model_name> (As appropriate to the STEP Implementation level chosen) EITHER complete – all native application data translated successfully into neutral form and exported¹⁰ OR partial – some native application data translated into neutral form and exported, other data ignored¹¹ OR incomplete – insufficient native data to translate into valid information in neutral form¹² OR failed – fatal error in export process¹³ OR aborted – export process terminated by user¹⁴ 	
Messages to display on-screen	
<ol style="list-style-type: none"> 9. "Data export may be incomplete - do you wish to continue exporting data from the application?" 10. "Export of data complete – all native application translated successfully into neutral form and exported" 11. "Warning – Only partial export of data occurred – some native application translated into neutral form and exported, other data ignored" 12. "Error – data export incomplete – insufficient native data to translate valid information into neutral form" 13. "Error – data export failed – fatal error in export process" 14. "Error – data export aborted – export process terminated by user" 	
See page 35 for the Notes to Tables 3.3 to 3.18	

Table 3.10 Section 4 of the Import Log File

SECTION 4: Application data import log	
Write to log file	
<ul style="list-style-type: none"> ❑ On importing data into the application: ❑ a total of <number> entity instances were processed and imported. ❑ <number> instances of type <entity name> were processed and imported ... repeat for all entity types imported. ❑ <number> instances of type <entity name> were ignored ... repeat for all entity types ignored ❑ <number> instances of <entity name> were imported with substitute values for <attribute name> ... repeat as required for all attributes that were given substitute values on import ❑ <number> 'Standard References' of type <item_reference_standard.ref> were ignored and not imported ... repeat as required for all item references that were ignored on import ❑ <number> 'Standard References' of type <item_reference_standard.ref> were imported with substitute values ... repeat as required for all item references that were given substitute values on import ❑ <number> 'Proprietary References' of type <item_reference_proprietary.ref> were ignored and not imported ... repeat as required for all item references that were ignored on import ❑ <number> 'Proprietary References' of type <item_reference_proprietary.ref> were imported with substitute values ... repeat as required for all item references that were given substitute values on import ❑ <number> 'Library References' of type <item_reference_library.ref> were ignored and not imported ... repeat as required for all item references that were ignored on import ❑ <number> 'Library References' of type <item_reference_library.ref> were imported with substitute values ... repeat as required for all item references that were given substitute values on import 	<ul style="list-style-type: none"> ❑ Warning – <entity name> is beyond the scope of this application - Data import may be incomplete¹⁵ ... repeat for all entity types ignored ❑ Warning – Data may have been modified from the original version contained in the imported STEP file / model¹⁵ ❑ Warning – <item_reference_standard.ref> not recognized – Data is beyond the scope of this application / Data import may be incomplete¹⁵ ❑ Warning – <item_reference_standard.ref> not recognized – Data may have been modified from original version contained in imported STEP file / model¹⁵ ❑ Warning – <item_reference_proprietary.ref> not recognized – Data is beyond the scope of this application / Data import may be incomplete¹⁵ ❑ Warning – <item_reference_proprietary.ref> not recognized – Data may have been modified from original version contained in imported STEP file / model¹⁵ ❑ Warning – <item_reference_library.ref> not recognized – Data is beyond the scope of this application / Data import may be incomplete¹⁵ ❑ Warning – <item_reference_library.ref> not recognized – Data may have been modified from original version contained in the exchange file¹⁵
See page 35 for the Notes to Tables 3.3 to 3.18	

Table 3.10 *Section 4 of the Import Log File (Continued)*

<p>❑ Summary Report of Import Process: Import of data from <file_name.name> / <model_name> to <application_name> <i>(As appropriate to the STEP Implementation level chosen)</i></p> <p>EITHER complete – all data imported successfully into receiving application¹⁶ OR partial – some data imported successfully into receiving application, other data ignored¹⁷ OR incomplete – insufficient neutral data to import valid information into receiving application¹⁸ OR failed – fatal error in import process¹⁹ OR aborted – import process terminated by user²⁰</p>
<p>Message to display on-screen</p>
<p>15. "Data import may be incomplete - do you wish to continue importing data?"</p> <p>16. "Import of data complete – all data translated successfully from neutral form and imported successfully into receiving application"</p> <p>17. "Warning – Only partial import of data occurred – some data translated from neutral form and imported successfully into application, other data ignored"</p> <p>18. "Error – data import incomplete – insufficient neutral data to import valid information into application"</p> <p>19. "Error – data import failed – fatal error in import process"</p> <p>20. "Error – data import aborted – import process terminated by user"</p>
<p><i>See page 35 for the Notes to Tables 3.3 to 3.18</i></p>

Table 3.11 Section 5 of the Export Log File

SECTION 5: Data Management details
<i>Write to log file</i>
<ul style="list-style-type: none"> <input type="checkbox"/> The Export Translator processed <number> instances of type managed_data_transaction <input type="checkbox"/> Of which ... <ul style="list-style-type: none"> <input type="checkbox"/> <number> were of the type managed_data_creation <ul style="list-style-type: none"> <input type="checkbox"/> of which <number> were created by <person_name⁽ⁱ⁾> on <date⁽ⁱⁱⁱ⁾> at <time⁽ⁱⁱⁱ⁾> ... repeat as required for each instance of managed_data_creation <input type="checkbox"/> <number> were of the type managed_data_modification <ul style="list-style-type: none"> <input type="checkbox"/> of which <number> were modified by <person_name⁽ⁱ⁾> on <date⁽ⁱⁱⁱ⁾> at <time⁽ⁱⁱⁱ⁾> ... repeat as required for each instance of managed_data_modification
<ul style="list-style-type: none"> <input type="checkbox"/> The Export Translator processed <number> instances of type managed_data_item <input type="checkbox"/> Of which ... <ul style="list-style-type: none"> <input type="checkbox"/> <number> were marked as deleted <input type="checkbox"/> <number> were deleted by <person_name⁽ⁱ⁾> on <date⁽ⁱⁱⁱ⁾> at <time⁽ⁱⁱⁱ⁾> ... repeat as required for each instance of managed_data_item marked as deleted <input type="checkbox"/> <number> had a new unique id created for this export operation, <input type="checkbox"/> <number> had an existing unique id maintained during this export operation <input type="checkbox"/> List of new unique identifiers created for this export operation: LIST OF <unique id^(iv)> <input type="checkbox"/> List of existing unique identifiers maintained during this export operation: LIST OF <unique id^(iv)>
<ul style="list-style-type: none"> <input type="checkbox"/> Total number of managed data entity instances: <number> ^{21, 22, 23} (That is, new, modified, or deleted instances of engineering data entities exported with instances of data_management_item assigned to them) <input type="checkbox"/> Total number of unmanaged data entity instances: <number> ^{21, 22, 23} (That is, instances without instances of data_management_item assigned to them)
<i>Messages to display on-screen</i>
<p>If <number> of managed data entity instances = 0 and <number> of unmanaged data entity instances = 0</p> <p>21. "Error – no data exported"</p> <p>If <number> of managed data entity instances = 0 and <number> of unmanaged data entity instances > 0</p> <p>22. "Warning - The DMC Translator has not exported any managed data"</p> <p>If <number> of managed data entity instances > 0 and <number> of unmanaged data entity instances > 0</p> <p>23. "Warning - The DMC Translator has exported both managed and unmanaged data"</p>
<p>Notes</p> <p>(i) Obtained from <managed_data_transaction.user.person_and_organization.the_person> (if present)</p> <p>(ii) Obtained from <managed_data_transaction.processing_date.date_and_time.date_component> (if present)</p> <p>(iii) Obtained from <managed_data_transaction.processing_date.date_and_time.time_component> (if present)</p> <p>(iv) Obtained from <managed_data_item.instance_id> plus <managed_data_item.originating_application.managed_application_installation.application_id> plus <managed_data_item.originating_application.managed_application_installation.installation_id></p>
See page 35 for the Notes to Tables 3.3 to 3.18

Table 3.12 Section 5 of Import Log File

SECTION 5: Data Management details
<i>Write to log file</i>
<ul style="list-style-type: none"> <input type="checkbox"/> The Import Translator processed <number> instances of type managed_data_transaction <input type="checkbox"/> Of which ... <ul style="list-style-type: none"> <input type="checkbox"/> <number> were of the type managed_data_creation, <ul style="list-style-type: none"> <input type="checkbox"/> <number> were created by <person_name> on <date> at <time> ... repeat as required for each instance of managed_data_creation encountered <input type="checkbox"/> <number> were of the type managed_data_modification <ul style="list-style-type: none"> <input type="checkbox"/> <number> were modified by <person_name> on <date> at <time> ... repeat as required for each instance of managed_data_modification encountered
<p>The Import Translator encountered</p> <ul style="list-style-type: none"> <input type="checkbox"/> A total of <number> managed data entity instances^{24, 25, 26} <input type="checkbox"/> A total of <number> unmanaged data entity instances^{24, 25, 26} <ul style="list-style-type: none"> <input type="checkbox"/> And has created a new instance of managed_data_item for each unmanaged data entity instance and placed it in a new instance of managed_data_creation ... if appropriate
<ul style="list-style-type: none"> <input type="checkbox"/> The Import Translator encountered <number> instances of type managed_data_item <input type="checkbox"/> Of which ... <ul style="list-style-type: none"> <input type="checkbox"/> <number> were marked as deleted <input type="checkbox"/> <number> were deleted by <person_name> on <date> at <time> ... repeat as required for each instance of managed_data_item marked as deleted <input type="checkbox"/> List of existing unique identifiers encountered during this import operation: ... LIST OF <unique id> <input type="checkbox"/> The Import Translator created <number> instances of type managed_data_item <ul style="list-style-type: none"> <input type="checkbox"/> List of new unique identifiers created during this import operation for the unmanaged data: ... LIST OF <unique id> ... if appropriate
<i>Messages to display on-screen</i>
<p><i>If <number> of managed data entity instances = 0 and <number> of unmanaged data entity instances = 0</i></p> <p>24. "Error – no data found"</p> <p><i>If <number> of managed data entity instances = 0 and <number> of unmanaged data entity instances > 0</i></p> <p>25. "Warning - The DMC Translator has not found any managed data in the incoming data set – Would you like the DMC Translator to manage the data?"</p> <p><i>If <number> of managed data entity instances > 0 and <number> of unmanaged data entity instances > 0</i></p> <p>26. "Warning - The DMC Translator found both managed and unmanaged data in the incoming data set - the DMC Translator has therefore managed all the data"</p>
<p><i>See page 35 for the Notes to Tables 3.3 to 3.18</i></p>

Table 3.13 *Section 6b of the Export Log File*
(version used with PMR-Enabled export translators)

SECTION 6b: Operational details
Write to log file
<p>During the export operation:</p> <p><i>EITHER</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The user requested a 'Create Model' operation <ul style="list-style-type: none"> – all the exported application data has been placed in a new repository model <STEP model name> in the PMR <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The user requested a 'Replace Model' operation <p><i>EITHER</i></p> <ul style="list-style-type: none"> – An equivalent STEP model <STEP model name> was found in the PMR's repository – all the exported application data has replaced the data in the repository model <p><i>OR</i></p> <ul style="list-style-type: none"> – An equivalent STEP model was not found in the PMR's repository – all the exported application data has been placed in a new repository model <STEP model name> in the PMR <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The user requested an 'Update Model' operation <p><i>EITHER</i></p> <ul style="list-style-type: none"> – An equivalent STEP model <STEP model name> was found in the PMR's repository and was used for the Comparator operations <p><i>OR</i></p> <ul style="list-style-type: none"> – An equivalent STEP model was not found in the PMR's repository – one has been created from a reference STEP file <file name> and was used for Comparator operations <p><i>OR</i></p> <ul style="list-style-type: none"> – An equivalent STEP model was not found in the translator's repository – the translator was unable to create one from a reference STEP file - no Comparator operations were performed – all the exported data has been placed in a new model <STEP model name> in the PMR's repository. <p><i>EITHER</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The user requested an 'Update & Replace Model' operation <ul style="list-style-type: none"> – some of the exported application data has replaced equivalent data in the repository model <STEP model name>, while some of the data in the repository model has been deleted <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The user requested an 'Update & Maintain Model' operation <ul style="list-style-type: none"> – some of the exported application data has replaced equivalent data in the repository model <STEP model name>, while some of the data in the repository model has been maintained
See page 35 for the Notes to Tables 3.3 to 3.18

Table 3.14 *Section 6a of the Import Log File
(Version used with IDI translators)*

SECTION 6a: Operational details
Write to log file
<p>During the Import operation:</p> <p><i>EITHER</i></p> <p><input type="checkbox"/> An equivalent STEP model was found in the IDI translator’s repository and was used for the Comparator operations.</p> <p>– some of the imported data has replaced equivalent data in the translator’s repository model <STEP model name>, while some of the data in the repository model has been maintained</p> <p><i>OR</i></p> <p><input type="checkbox"/> An equivalent STEP model was not found in the translator’s repository – one has been created from a reference STEP file <file name> and was used for the Comparator operations.</p> <p>– some of the imported data has replaced equivalent data in the translator’s repository model <STEP model name>, while some of the data in the repository model has been maintained</p> <p><i>OR</i></p> <p><input type="checkbox"/> An equivalent STEP model was not found in the translator’s repository – the translator was unable to create one from a reference STEP file - no Comparator operations were performed.</p> <p>– all the imported data has been placed in a new model <STEP model name> in the translator’s repository</p>

Table 3.15 *Section 6b of the Import Log File
(version used with PMR-Enabled import translators)*

SECTION 6b: Operational details
Write to log file
<p>During the Import operation from the PMR:</p> <p><i>EITHER</i></p> <p><input type="checkbox"/> An equivalent STEP model was found in the translator’s repository and was used for the Comparator operations.</p> <p>– some of the data imported from the PMR has replaced equivalent data in the translator’s repository model <STEP model name>, while some of the data in the repository model has been maintained.</p> <p><i>OR</i></p> <p><input type="checkbox"/> An equivalent STEP model was not found in the translator’s repository – one has been created from a reference STEP file <file name> and was used for the Comparator operations.</p> <p>– some of the data imported from the PMR has replaced equivalent data in the translator’s repository model <STEP model name>, while some of the data in the repository model has been maintained</p> <p><i>OR</i></p> <p><input type="checkbox"/> An equivalent STEP model was not found in the translator’s repository – the translator was unable to create one from a reference STEP file - no Comparator operations were performed.</p> <p>– all the data imported from the PMR has been placed in a new model <STEP model name> in the translator’s repository</p>
See page 35 for the Notes to Tables 3.3 to 3.18

Table 3.16 *Section 7b of the Export Log File*
(Version used with PMR-Enabled export translators)

SECTION 7b: Data Comparison & Updating details
Write to log file
<p>When comparing the exported application data with the equivalent STEP model in the PMR repository:</p> <p><i>For each unique identifier...</i></p> <p><i>EITHER</i></p> <ul style="list-style-type: none"> ❑ The unique identifier <unique id> was found in the application data but not in the existing repository model. <ul style="list-style-type: none"> – The identifier and the associated data entity instance of type <entity name> were added to the repository model during the [Update & Maintain / Update & Replace] <p><i>OR</i></p> <ul style="list-style-type: none"> ❑ The unique identifier <unique id> was found in the existing repository model but not in the application data. <ul style="list-style-type: none"> – The identifier and the associated data entity instance of type <entity name> have been [maintained / deleted] during the [Update & Maintain / Update & Replace] process. <p><i>OR</i></p> <ul style="list-style-type: none"> ❑ The unique identifier <unique id> is matched in both the application data and the existing repository model. <ul style="list-style-type: none"> – This identifier was assigned to the entity type <entity name> in the repository model, and has been assigned to the entity type <entity name> in the incoming data. – the PMR's Comparator has updated the entity instance accordingly <p><i>... repeat as required</i></p>
See page 35 for the Notes to Tables 3.3 to 3.18

Table 3.17 *Section 7a of the Import Log File*
(Version used with IDI translators)

SECTION 7a: Data Comparison & Updating details
Write to log file
<p>When comparing the incoming data with the equivalent STEP model in the translator's repository:</p> <p><i>For each unique identifier...</i></p> <p><i>EITHER</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The unique identifier <unique id> was found in the incoming data but not in the existing repository model. <ul style="list-style-type: none"> – The identifier and the associated data entity instance of type <entity name> were added to the data included in the incremental data import process. <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The unique identifier <unique id> was found in the existing repository model but not in the incoming data. <ul style="list-style-type: none"> – The identifier and the associated data entity instance of type <entity name> were ignored for the incremental data import process. <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The unique identifier <unique id> is matched in both the incoming data and the existing repository model. <ul style="list-style-type: none"> – This identifier was assigned to the entity type <entity name> in the repository model, and has been assigned to the entity type <entity name> in the incoming data. <p><i>EITHER</i></p> <ul style="list-style-type: none"> – All the attributes of this entity in the repository model are exactly the same as those in the incoming data. The identifier and the associated data entity instance of type <entity name> have been ignored for the incremental data import process <p><i>OR</i></p> <ul style="list-style-type: none"> – Some, or all, of the attributes of this entity in the repository model are different from those in the incoming data – the Comparator has updated the entity instance accordingly <p><i>... repeat as required</i></p>
<ul style="list-style-type: none"> <input type="checkbox"/> Summary of the Incremental Data Import: <p>Of a total of <number> data entity instances that were encountered in the incoming data set, a total of <number> were processed and imported into the application, while <number> were ignored for the incremental data import process.</p>
<p><i>See page 35 for the Notes to Tables 3.3 to 3.18</i></p>

Table 3.18 *Section 7b of the Import Log File*
(Version used with PMR-Enabled import translators)

SECTION 7b: Data Comparison & Updating details
<i>Write to log file</i>
<p>When comparing the data imported from the PMR with the equivalent STEP model in the translator's repository:</p> <p><i>For each unique identifier...</i></p> <p><i>EITHER</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The unique identifier <unique id> was found in the incoming data but not in the existing repository model. <ul style="list-style-type: none"> – The identifier and the associated data entity instance of type <entity name> were added to the data included in the incremental data import process. <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The unique identifier <unique id> was found in the existing repository model but not in the incoming data. <ul style="list-style-type: none"> – The identifier and the associated data entity instance of type <entity name> were ignored for the incremental data import process. <p><i>OR</i></p> <ul style="list-style-type: none"> <input type="checkbox"/> The unique identifier <unique id> is matched in both the incoming data and the existing repository model. <ul style="list-style-type: none"> – This identifier was assigned to the entity type <entity name> in the repository model, and has been assigned to the entity type <entity name> in the incoming data. <p><i>EITHER</i></p> <ul style="list-style-type: none"> – All the attributes of this entity in the repository model are exactly the same as those in the incoming data. The identifier and the associated data entity instance of type <entity name> have been ignored for the incremental data import process <p><i>OR</i></p> <ul style="list-style-type: none"> – Some, or all, of the attributes of this entity in the repository model are different from those in the incoming data – the Comparator has updated the entity instance accordingly <p><i>... repeat as required</i></p>
<ul style="list-style-type: none"> <input type="checkbox"/> Summary of the Incremental Data Import: <p>Of a total of <number> data entity instances that were encountered in the incoming data set, a total of <number> were processed and imported into the application, while <number> were ignored for the incremental data import process.</p>
<i>See page 35 for the</i>
<i>Notes to Tables 3.3 to 3.18</i>

4 CONFORMANCE TESTING

This Section specifies the requirements of the formal Conformance Testing procedures that demonstrate whether a specific implementation conforms to the CIS/2 specifications. Thus, this Section is relevant to both software vendors developing CIS/2 translators and to third party testing bodies performing Conformance Testing of CIS/2 systems. It discusses how a CIS translator should be submitted to a testing body by a software vendor for conformance testing and how it will be tested for compatibility with CIS/2 and particular Conformance Classes. It also describes how problems with CIS/2 implementations are dealt with, and what degrees of ‘CIS Conformance’ can result.

4.1 Introduction

Conformance testing is the evaluation of an implementation of CIS/2 (i.e. a translator or PMR) for all the required characteristics to determine whether the implementation conforms to the CIS/2 specifications. The characteristics include the Logical Product Model (its entities, types, attributes, functions, rules, and the full range of values), and any implementation requirements defined in the ‘Conformance Requirements’ specific to the type of CIS/2 translator (specified in Sections 3.6, 3.7, 3.8, 3.9) and the of STEP implementation level chosen (specified in Sections 3.2, 3.3, and 3.4).

The CIS/2 framework and methodology for conformance testing reflects the importance of testing to the success of any implementation of the CIS. Experience in other domains has shown that such a strategy is an essential prerequisite to repeatability and consistency of testing, and therefore of mutual recognition of test results across regional and national boundaries. The more formal and independent the testing regime is, the more confidence end-users will have in the application’s capabilities.

It should be remembered that any Conformance Testing regime is testing the capabilities of the translator as claimed by the software vendor. The testing body will need to take into account any limitations of the application and/or its translator.

Conformance of an implementation of the CIS (e.g. a translator or PMR) is expressed as conformance to one or more Conformance Classes (CCs), including all types, entities, attributes, functions and rules contained in the CC(s). An implementation may also exhibit conformance to particular ‘flavours’ (i.e. lists of standard or proprietary product references) or unit systems. (See Section 3.5.)

Conformance testing takes place at two levels. The first basic level tests whether a translator is capable of exchanging data. The second tests whether the translator is capable of understanding the data. There is a significant difference between reading or writing the data, and understanding the underlying information content held therein.

4.1.1 Import vs export

It is recognized that testing the capability of a translator will be more difficult for import translators than export translators. This is because the data imported into an application may not be easily visible, or may be held implicitly in an application. Any testing body must be assured that an import translator has imported the neutral data into the application and translated it into native data. Thus, the validation criteria used when testing a CIS translator needs to include the requirement for an import translator to be able to show the effect of the imported data. It is up to the software vendor to define

how the data is held within their application and describe how a testing body can detect the effect of the imported data on the application and its native data.

4.1.2 ‘Intelligent’ Translators

CIS translators are required to exhibit a certain degree of ‘intelligence’. If a translator is not able to understand a piece of information, it is required to state (as an error message) why it cannot understand the underlying information related to the data it encounters, and it must be clear what the translator will do in such an event. The most likely failure will be because the functionality required to use the information encountered lies beyond the scope of the application for which the translator has been written.

CIS translators are required to be able to process the full range of values for every attribute of every entity that it claims to support. For example, the attribute ‘model_type’ of the entity ‘analysis_model’ has the data type ‘frame_type’, which is defined thus;

```
TYPE frame_type = ENUMERATION OF (space_frame, space_truss, plane_frame,
    plane_truss, grillage, undefined);
END_TYPE;
```

All translators claiming to support the entity ‘analysis_model’ are required to be able to read (on import) or create (on export) the enumerated label for all types of analysis model. This does not imply that all types of analysis models will appear in a CIS exchange file (or even any, as the attribute may be set to ‘undefined’), merely that a system can distinguish analysis models of the different types, if desired.

The above requirement means, for example, that if an import translator written for an application that cannot handle grillage type structures encounters an analysis model of the type ‘grillage’, it must recognize that a grillage is beyond the scope of the application, and must display an error message to that effect.

This ‘intelligence’ will be tested as part of the capability tests. (See Section 4.2.4.)

4.2 Types of Conformance Testing

The CIS Strategy for conformance testing takes a staged approach, as outlined later in this Section. There are three types of conformance tests recognized by the CIS:

- Basic tests,
- Inter-working tests, and
- Capability tests.

4.2.1 Stage 0 - Pre-testing

Requirements of the documentation

Before physical testing takes place,

1. The vendor of a candidate CIS/2-system is required to submit a formal statement to the testing body of the capabilities of CIS/2-system (see Sections 3.5.3, 3.6.3, 3.7.3, 3.8.3, 3.9.3). This statement will include a list of the CCs, Flavour(s) and unit system(s) that the CIS/2-system supports on import and /or export. If the

capabilities of a CIS/2-system are different on import and export, then two separate statements shall be submitted on each occasion: one describing the system's import capabilities and one describing its export capabilities.

2. The testing body shall examine the documentation submitted with the candidate CIS/2-system, and if the vendor has not provided the information required - as specified by the 'Conformance Requirements' (See Section 3 of this document) - then physical conformance testing may not proceed.

4.2.2 Stage 1 - Basic tests

Requirements of Basic Tests

1. During basic tests, the testing body shall evaluate whether the CIS/2-system meets all of the 'Conformance Requirements' appropriate to the complexity of the CIS translator (specified in Sections 3.5, 3.6, 3.7, 3.8, 3.9) and level of STEP implementation (specified in Sections 3.2, 3.3, 3.4).
2. The testing body shall also evaluate whether the log files created by the CIS/2 system meets all of the requirements are specified in Section 3.11.

Commentary

Basic tests check whether the CIS/2-system can support the appropriate physical file for the combinations of CCs claimed by the software vendor. For example, if an import translator claims to support CC201, then it shall be capable of reading a CIS exchange file that has been created in accordance with CC201. Similarly, if an export translator claims to support CC201, then it shall be capable of creating a CIS exchange file in accordance with CC201. (A selection of test files is included with the complete CIS/2 documentation.)

At this stage, conformance testing is merely showing whether the translator can exchange data; i.e. whether the translator is capable of reading (or writing) the values of the attributes of the appropriate entities. Whether the translator is capable of 'understanding' the data is the subject of stages 2 and 3.

4.2.3 Stage 2 - Inter-working tests

The second stage of testing involves inter-working tests with a number of known 'control' CIS/2-conformant systems working with the candidate CIS/2-system. A sequence of CIS export files will be generated by the 'control' CIS/2-conformant systems, in the case of an import translator, or by the candidate CIS/2-system, in the case of an export translator.

Requirements of Inter-working Tests

1. The data in the test files shall be generated by the testing body during the testing in a random manner and shall conform to the appropriate CC(s) and flavour(s).
2. These test files will then be imported into the candidate CIS/2-system using the candidate's import translator, or imported into the control CIS/2-conformant systems, as appropriate.
3. The testing body shall then report the results of these tests to the vendor of the candidate CIS/2-system, and any inconsistencies found between the information imported into the different CIS/2-conformant systems shall be highlighted.

Commentary

Typical problems that are likely to occur are those involving the conversion of units, e.g. converting to metres rather than millimetres, or when transforming axes, e.g. transposing the x and y ordinates.

4.2.4 Stage 3 - Capability tests

Capability tests check that the observable capabilities of the implementation are in accordance with the capabilities claimed by the developer of the CIS implementation. They endeavour to be as comprehensive as possible over the full range of conformance requirements as specified by the CIS.

Requirements of Capability Tests

During capability tests, the testing body shall

1. Check all mandatory capabilities and those optional ones that are claimed by the developer as being supported by the implementation.
2. Evaluate whether all entities, types, and their associated constraints identified in a particular Conformance Class are supported.
3. Evaluate whether the treatment of options and default values conforms to the Logical Product Model.
4. Test whether the CIS/2-system supports all valid combinations of entities and their attributes contained in the CC(s) under consideration. That is, whether an export translator is capable of creating populated instances of these entities, and whether an import translator is capable of reading instances of these entities.
5. Check that the constructs produced or accepted by an implementation are only those constructs specified in the Logical Product Model. (That is, ‘user-defined’ entities shall not be deemed acceptable.)
6. Check that the implementation satisfies all the general requirements applicable to the implementation form given in the appropriate part of the 20-series class of STEP (e.g. Part 21) and any specific options given in the ‘Conformance Requirements’. (See Section 3. for details of these requirements).

Commentary

This third stage of testing requires the use of formalized test cases with appropriate test methods. Implementations of CIS/2 may choose any of the implementation methods specified in the *Implementation Guide*^[3]. Consequently, there are several ways in which implementations can be controlled and observed during the conformance testing process. A test method is required for each implementation method.

A number of test cases are used when assessing the conformance of an implementation to the data specification contained in the Logical Product Model. Test cases are documented in Section 6. Each gives a precise description of the objective that a test case is designed to achieve. A test case is used as the basis for generating a number of executable tests that are independent of the implementation under test.

4.3 Conformance Testing Statements

4.3.1 General requirements

Software vendors shall describe the conformance of their CIS/2-conformant system using one of the following terms:

- ‘Based on CIS/2’
- ‘Accepted for CIS Conformance Testing’
- ‘CIS Conformance Tested’

Only the ‘Based on CIS/2’ statement is produced by the software vendor of the candidate CIS/2 system. The ‘Accepted for CIS Conformance Testing’ and the ‘CIS Conformance Tested’ statements are created by a Conformance Testing body. Software vendors may reproduce (but not change) the statements provided by the testing body.

4.3.2 Vendor's ‘Based-on CIS/2’ Statement

Requirements for a ‘Based on CIS/2’ statement

1. Once the software vendor is satisfied that their CIS/2 system conforms to the requirements of the relevant CCs, they may issue their own statement to that effect. The statement shall (as a minimum) provide the extent of the information given in the example statement in Figure 4.1.
2. Until a ‘final’ ‘Acceptance for Testing Report’ has been issued, a vendor shall not refer to their CIS/2 system as anything other than being ‘Based-on the CIS’.
3. Until superseded by an Acceptance Statement, this ‘**Based-on Statement**’ shall be reproduced in full (without changes), and be given to all users of the application. Based-on Statements may include any riders that the vendor considers appropriate.

Commentary

The format of the example ‘Based-on Statement’ is shown in Figure 4.1 provides sufficient information and it is recommended that the general format of this statement be adopted by vendors of CIS/2-conformant systems. The example indicates that the software vendor has developed an import translator for this application that can deal with data specified by entities in Conformance Class 124 when combined with data specified by entities in Conformance Class 234. The vendor is also claiming that the translator can handle both the European and American Flavours as well as SI and US Imperial unit systems. The operation of the translator is limited to that of a Basic CIS Translator and is implemented using only STEP implementation level 1 (i.e. data sharing via physical files). The vendor has placed no riders on the statement – implying that the scope of the data exchanged is not limited by the scope of the application or any limitations of the translator. The application’s translator is described as being “Based on CIS/2 for the import of data conforming to (CC124+CC234)”.

4.3.3 ‘Accepted for CIS/2 Testing’ Statement

Requirements for an ‘Accepted for CIS/2 Testing’ statement

1. When the software vendor has been issued with a final ‘**Acceptance for Testing Report**’ by the testing body, that vendor is entitled to refer to the application’s translators as being ‘**Accepted for CIS/2 Conformance Testing**’.

2. The ‘Accepted for CIS/2 Testing’ statement shall include all the information shown in the example in Figure 4.2.
3. Until superseded by a ‘Compliance Statement’, the ‘Accepted for CIS/2 Testing’ shall be reproduced in full (without changes), and be given to all users of the CIS/2 system.
4. Acceptance Statements shall include any riders that the testing body considers appropriate.

4.3.4 ‘Acceptance for Testing’ Report

An Acceptance for Testing Report provides a clear and concise report on the outcome of pre-testing (as described in Section 4.2), and can be either ‘*intermediate*’ or ‘*final*’. Intermediate Acceptance for Testing Reports will be issued only to the vendor and will report on the reasons why the CIS/2 system (e.g. an application’s translator) has not been formally accepted for testing. Final Acceptance for Testing Reports will also be issued to the vendor, but will also be made available to any end-user who requests a copy.

Requirements of an ‘Acceptance for CIS/2 Testing’ Report

An Acceptance for CIS/2 Testing Report shall include the following sections:

1. Intermediate or final test outcome
2. Full details of the software that was tested (which application and translators were tested and for which CCs, flavours, units, etc.)
3. Details of the test that were carried out (by whom and when, and which test cases and what methods were used)
4. Test records of the test procedures used and the key outcomes
5. Tester’s Riders
6. Draft or final Acceptance Statement

Commentary

An ‘intermediate test outcome’ will list the remaining problems, while a ‘final test outcome’ will indicate how the accepted application can now be formally described.

4.3.5 ‘CIS/2 Conformance’ Statement

Requirements for a ‘CIS/2 Conformance’ statement

1. When a vendor has been issued with a final ‘**Test Summary Report**’ that vendor is entitled to refer the CIS/2 system as being ‘**CIS/2 Conformance Tested**’.
2. The ‘CIS/2 Conformance’ Statement shall include all the information shown in the example in Figure 4.3.
3. The ‘CIS/2 Conformance’ Statement shall be reproduced in full (without changes), and be given to all users of the application.

4. Conformance Statements shall include any riders that the testing body considers appropriate.

4.3.6 ‘CIS/2 Conformance Testing’ Report

A ‘CIS/2 Conformance Testing’ Report provides clear and concise report on the outcome of formal conformance testing, and can be either *intermediate* or *final*. Intermediate Test Reports will be issued to the vendor only and will report on the reasons why the application has still not been deemed ‘CIS/2 Conformant’. Final Test Reports shall be issued to the vendor and will also be made available to any end-user who requests a copy.

Requirements of a ‘CIS/2 Conformance Testing’ Report

A ‘CIS/2 Conformance Testing’ Report shall include the following sections:

1. Outcome(s) of the test(s)
2. Intermediate or Final Test Report
3. Full details of the software that was tested (which application and translators were tested and for which CCs, flavours, units, etc.)
4. Details of the test that were carried out (by whom and when, and which test cases and methods were used and with which applications the inter-working tests were conducted)
5. Test Record(s) of the test procedures used and the key outcomes
6. Tester’s Riders (Statements that are to be used with the Conformance Statement)
7. Draft or Final Test Statement

An ‘intermediate test outcome’ will list the remaining problems, while a ‘final test outcome’ will indicate how the tested application can now be formally described.

4.3.7 Incremental Testing

It is anticipated that applications with complex translators (which might support many different combinations of CCs, several unit systems, multiple flavours, and comprehensive list of references for manufacturer’s products) will be the subject of incremental testing. Thus, different aspects of the translator may be eligible to different degrees of conformance statement. In such cases, each test report will only address the current aspect of the testing, and should read as one of a sequence of related reports, with references to the previous test reports for that application.

The result of this incremental testing will be a collection of statements for each aspect of the translator that is either ‘Based-on CIS/2’, has been ‘Accepted for CIS/2 Testing’, or has been ‘CIS/2 Conformance Tested’

Based-on CIS/2 Statement	
Application Name: <label>	
Application Version: <label>	Date: <date>
Translator Version: <label>	Date: <date>
Software Vendor: <organization> <address>	
The translator(s) for this application have been implemented in accordance with the second release of the CIMsteel Integration Standards (CIS/2.1) for the following (combination of) Conformance Classes:	
(CC123+CC234)	
Type of CIS Translator:	<u>Basic</u> DMC IDI PMR-enabled
Data exchange capabilities:	<u>Import</u> Export Import & Export
Level of implementation:	<u>File Exchange</u> In memory DBMS KBS
Flavour(s) supported:	<u>EU</u> <u>US</u> UK Other <input type="text"/>
Unit System(s) supported:	<u>SI</u> <u>US Imperial</u> Other <input type="text"/>
The vendor places the following riders on the operation of the translators:	
none	
Date of Statement: <date>	
Statement made by: <person> of <organization>	

Figure 4.1 Example 'Vendor's Based-on Statement'

Note

If the capabilities of a CIS/2-system are different on import and export, then two separate statements shall be submitted: one describing the system's import capabilities and one describing its export capabilities.

Accepted for CIS/2 Testing Statement

Application Name: *<label>*

Application Version: *<label>* Date: *<date>*

Translator Version: *<label>* Date: *<date>*

Software Vendor: *<organization>* *<address>*

Version *<label>* of the translator for this application was pre tested on *<date>* by *<testing organization>* against the Conformance Requirements of the second release of the CIMsteel Integration Standards (**CIS/2.1**) and was formally accepted onto the testing programme for the following (combination of) Conformance Classes:

(CC123+CC234)

Type of CIS Translator: Basic | DMC | IDI | PMR-enabled

Data exchange capabilities: Import | Export | Import & Export

Level of implementation: File Exchange | In memory | DBMS | KBS

Flavour(s) supported: EU | US | UK | Other

Unit System(s) supported: SI | US Imperial | Other

The testers have placed the following riders on their acceptance:

none

A copy of the 'Acceptance for Testing' report (Report Number *<number>*) is available on request from *<testing organization>*

Date of Statement: *<date>*

Statement made by: *<person>* of *<organization>*

Figure 4.2 Example 'Acceptance for Testing Statement'

Note

If the capabilities of a CIS/2-system are different on import and export, then two separate statements shall be submitted: one describing the system's import capabilities and one describing its export capabilities.

CIS/2 Conformance Statement	
Application Name: <label>	
Application Version: <label>	Date: <date>
Translator Version: <label>	Date: <date>
Software Vendor: <organization> <address>	
Testing Body: <testing organization> <address>	
Version <label> of the translator for this application was tested on <date> by <testing organization> against the Conformance Requirements of the second release of the CIMsteel Integration Standards (CIS/2.1) and, within the limitations of the testing programme, was deemed to have performed satisfactorily for the following (combination of) Conformance Classes:	
(CC123+CC234)	
Type of CIS Translator:	Basic DMC IDI PMR-enabled
Data exchange capabilities:	Import Export Import & Export
Level of implementation:	File Exchange In memory DBMS KBS
Flavour(s) supported:	EU US UK Other <input type="text"/>
Unit System(s) supported:	SI US Imperial Other <input type="text"/>
The testers have placed the following riders on the outcome:	
none	
A copy of the 'Conformance Test Report' (Report Number <number>) is available on request from <testing organization>	
Date of Statement: <date>	
Statement made by: <person> of <organization>	

Figure 4.3 Example 'Conformance Statement'

Note

If the capabilities of a CIS/2-system are different on import and export, then two separate statements shall be submitted: one describing the system's import capabilities and one describing its export capabilities.

5 INDUSTRIAL REALIZATION

This Section includes a textual description of how CIS/2 deals with national conventions and variations in the referencing of standard and manufacturers' product items and unit systems.

5.1 International considerations

The CIS is intended to provide a data exchange standard that is acceptable, and applicable, across the world. There are, however, significant national and regional differences in practice, terminology and custom. Three primary areas of international variation can be identified:

- Conventions (such as axis systems)
- Units of measure
- Product items (such as section profiles, bolt, nuts, etc.)

Additionally, business practices in the industry continue to change, both nationally and internationally, as innovations in technology, construction techniques and products evolve. These problems will be familiar to vendors of engineering applications addressing more than one market, and are, therefore, addressed by the CIS.

In theory, a governing schema could be defined that was able to accommodate all local variations in practice. However, the complexity of the resulting data exchange standard would mean that application vendors would find them very difficult to implement, and when implemented, they would be inefficient to use.

The Sections that follow describe how CIS/2 addresses international considerations in each of the three areas identified above, thus providing a flexible and appropriate approach to international deployment. Within this context, CIS/2 provides two unit systems (SI and US Imperial) and employs international 'Standard Identifiers'².

5.2 Conventions

Experience has shown that one of the more difficult things to define unambiguously in a data exchange standard is the convention relating to axis systems, co-ordinate systems, etc. Additionally, international trade may require the CIS to be applied in situations where, for example, section profiles from more than one country will be accommodated within the same data exchange file. For simplicity, the CIS specifies only one set of (internal) conventions. The treatment of conventions within the CIS is such that:

- When seen from the perspective of an **end-user**, the CIS conventions are an internal (implementation) matter.
- When seen from the perspective of an **implementer**, only one set of CIS conventions needs to be understood.

² In both cases, the concept is similar to that used in CIS/1 although the implementation differs.

Where particular software applications employ different conventions from the CIS, these conventions shall be mapped by the developer of the CIS/2 system to (or from) the CIS conventions. Once correctly established, these mappings should be simple to implement, with the result that the implementation complexity of exchanging data between applications that use different conventions is avoided.

The CIS conventions are fully defined in *The Logical Product Model*^[5]. Software vendors are advised to take particular note of these conventions before developing CIS translators.

5.3 Units of Measure

It should be noted that, although LPM/6 allows any unit system to be defined, CIS/2 defines only two ‘standardized’ (and constrained) unit systems: ‘SI’ and ‘US Imperial’, such that all attributes representing a particular physical quantity use the pre-defined unit. A set of units for particular physical quantities (i.e. length, mass, time, etc.) for each system is documented later in this Section. If a CIS Translator uses one of these ‘standardized’ unit systems, unit conversion is not a big problem for the translator implementer, provided that the required CIS unit is clearly defined, and belongs to the same unit system as the corresponding data within the application.

It should be noted that these ‘standardized’ unit systems are limited subsets of their parent unit systems. For example, the ‘standardized’ SI Units System (as defined by CIS/2) provides only one value for length (millimetres). Other units of length (e.g. kilometres) are excluded from this ‘standardized’ system. Furthermore, any derived units not defined in the ‘standardized’ SI Unit system that require a length component should be defined in terms of mm.

It is a Conformance Requirement of CIS/2 that Basic CIS Translators (see Section 3.5.1) support either (or both) of the ‘standardized’ unit systems when handling data representing physical quantities. More advanced CIS translators are at liberty to support other unit systems in addition to the either (or both) of the ‘standardized’ unit systems.

A CIS data exchange file explicitly specifies the units associated with every instance of those numeric attributes representing physical quantities that are not dimensionless. In theory, each numeric value can be expressed using different units (of the required dimensionality). In practice, the same unit is likely to be used for all occurrences of an attribute, and may well be used for all numeric values having the same dimensionality. Thus, at its most generic, a particular CIS data exchange file can be said to have a ‘soft’ unit system - meaning that the units used have been selected entirely by the vendor who wrote the export translator (and thus are required to be checked on reading for each attribute). In practice, if a new unit system is required for a non-Basic Translator (one different from the standardized SI and US Imperial unit systems), it can only be agreed in consultation with a number of other software vendors, and should only be defined in consultation with the **CIS/2 International Technical Committee**.

An application vendor may implement an export translator that is capable of exporting files that comply with one (or more) unit systems. If a translator can export files that comply with more than one system, it shall allow the user to specify which will be used. Similarly, a given import translator may be able to import files that comply with one or more unit systems. All import translators are required to automatically detect which unit system has been used in a file and act accordingly.

Within a particular exchange file, all numeric attribute values are required to be written - and read - using the units of measure specified for the particular unit system selected. As will be seen in the next Section, standard and manufacturer's product items can be exchanged either by reference, or by specifying values for their attributes. If the latter option is selected, the attribute values will be written in accordance with the governing unit system. However, when product items are passed by reference, their characteristic attribute values are implied rather than specified. By this means, a physical file that uses the SI unit system can (implicitly) convey attributes relating say to American sections that are usually specified in imperial units.

The LPM EXPRESS constructs that are used to represent units and unit systems are fully defined in Section 15 of *The Logical Product Model*^[5]. Vendors are advised to take particular note of these definitions before developing CIS translators.

5.3.1 The CIS/2 Standardized Units Systems

The sets of units that constitute the two standardized units systems used with CIS/2 are specified by the two physical files below. Electronic versions of these files are available on the CIS/2 web site (<http://www.cis2.org/>).

The physical files – named 'si_units.stp' and 'us_units.stp' – were tested using version 3.07 of the 'Express Engine' toolkit³. Although every care has been taken to ensure, to the best of our knowledge, that all data contained within these files are accurate to the extent that they conform to the published edition of the LPM/6 EXPRESS schema, The Steel Construction Institute, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data or any loss or damage arising from or related to their use. Any errors should be reported directly to The Steel Construction Institute.

Definition of the SI Unit System used with CIS/2

The following physical file listing contains the set of units that constitute the SI Unit System that is used with CIS/2. It should be noted that this is a limited subset of the complete SI Unit System as defined by ISO 1000^[8]. For example, the physical file below provides only one value for length (mm). Any derived units not defined in the physical file below that require a length component should be defined in terms of mm.

ISO-10303-21;

HEADER;

FILE_DESCRIPTION(('{cimsteel logical product model version (6) object (1) structural-frame-schema(1) }', 'CC034', 'SI Units'), '2;1');

FILE_NAME('si_units.stp', '2003-05-07 T14:25:07', ('Andrew Crowley'), ('Steel Construction Institute'), 'Expresso 3.0.7 (compiled 2003-Feb-20 21:07)', 'Express Engine', 'AJC');

FILE_SCHEMA(('STRUCTURAL_FRAME_SCHEMA'));

ENDSEC;

DATA;

#1 = REPRESENTATION ('Any representation that uses parameters defined in the SI Unit System', (#2), #3);

³ Now maintained as a project on SourceForge.net <http://www.exp-engine.sourceforge.net/>

```

#2 = REPRESENTATION_ITEM ('Any item of a representation that uses parameters
    defined in the SI Unit System');
#3 = GLOBAL_UNIT_ASSIGNED_CONTEXT ('context for any items that use SI Units',
    'CIS/2 standardized units for the SI Unit system', (#21, #22, #23, #24, #25, #26, #27,
    #28, #28, #30, #31, #32, #33, #34, #35, #36, #37, #38, #39, #40, #41, #42, #43));
/*
CIS/2 defines a constrained set of SI units comprising 12 base units (defined in terms of
metres, Kilograms, Seconds and Degrees Centigrade) and 11 other units derived from these
base units.
*/
/* 12 Base Units */
/*
All are instances of NAMED_UNIT. 10 are externally mapped complex instances with
SI_UNIT (for which the dimensionality is pre-defined via a function)
Of the 7 ISO dimensionalities (Length, Mass, Time, current, temperature, amount of
substance, luminosity) only 4 are used by the CIS.
*/
/* Length Unit (millimetre) */
#21 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.MILLI.,.METRE.));

/* Mass Unit (kilogram) */
#22 = (MASS_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.GRAM.));

/* Time Unit (second) */
#23 = (NAMED_UNIT(*) SI_UNIT($,.SECOND.) TIME_UNIT());

/* Thermodynamic Temperature (Degrees Celsius) */
#24 = (NAMED_UNIT(*) SI_UNIT($,.DEGREE_CELSIUS.)
    THERMODYNAMIC_TEMPERATURE_UNIT());

/* Plane Angle Unit (Radian) */
#25 = (NAMED_UNIT(*) PLANE_ANGLE_UNIT() SI_UNIT($,.RADIAN.));

/* Solid Angle Unit (Steradian) */
#26 = (NAMED_UNIT(*) SI_UNIT($,.STERADIAN.) SOLID_ANGLE_UNIT());

/* Area Unit (square centimetre) */
#27 = (AREA_UNIT() CONVERSION_BASED_UNIT('square centimetre', #272)
    NAMED_UNIT(#275));
#271 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.CENTI.,.METRE.));
#272 = MEASURE_WITH_UNIT (NUMERIC_MEASURE(1.0), #273);
#273 = DERIVED_UNIT ((#274));
#274 = DERIVED_UNIT_ELEMENT (#271, 2.0);
#275 = DIMENSIONAL_EXPONENTS (2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

/* Volume Unit (cubic metre) */
#28 = (CONVERSION_BASED_UNIT('cubic metre', #282) NAMED_UNIT(#285)
    VOLUME_UNIT());
#281 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

```

```

#282 = MEASURE_WITH_UNIT (NUMERIC_MEASURE(1.0), #283);
#283 = DERIVED_UNIT ((#284));
#284 = DERIVED_UNIT_ELEMENT (#281, 3.0);
#285 = DIMENSIONAL_EXPONENTS (3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

/* Ratio Unit (Percent) */
#29 = (CONTEXT_DEPENDENT_UNIT('percent') NAMED_UNIT(#291) RATIO_UNIT());
#291 = DIMENSIONAL_EXPONENTS (0.0,0.0,0.0,0.0,0.0,0.0,0.0);

/* Force Unit (kiloNewton) */
#30 = (FORCE_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.NEWTON.));

/* Frequency Unit (Hertz) */
#31 = (FREQUENCY_UNIT() NAMED_UNIT(*) SI_UNIT($,.HERTZ.));

/* Pressure Unit (kiloPascal) */
#32 = (NAMED_UNIT(*) PRESSURE_UNIT() SI_UNIT(.KILO.,.PASCAL.));

/* 11 Derived Units */
/*
    All are instances of SUBTYPES of DERIVED_UNIT and all have associated instances of
    DERIVED_UNIT_ELEMENT to specify their dimensionality
    For clarity, the required base units are repeated here. The physical file could be reduced
    by removing the copy instances of centimetre, metre, second, etc.
*/
/* Force per Length Unit (kiloNewton per metre) */
#33 = FORCE_PER_LENGTH_UNIT ((#331, #332));
#331 = DERIVED_UNIT_ELEMENT (#333, 1.0);
#332 = DERIVED_UNIT_ELEMENT (#334, -1.0);
#333 = (FORCE_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.NEWTON.));
#334 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

/* Inertia Unit (centimetre to the fourth power) */
#34 = INERTIA_UNIT ((#341));
#341 = DERIVED_UNIT_ELEMENT (#342, 4.0);
#342 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.CENTI.,.METRE.));

/* Linear Acceleration Unit (metre per second per second) */
#35 = LINEAR_ACCELERATION_UNIT ((#351, #352));
#351 = DERIVED_UNIT_ELEMENT (#353, 1.0);
#352 = DERIVED_UNIT_ELEMENT (#354, -2.0);
#353 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));
#354 = (NAMED_UNIT(*) SI_UNIT($,.SECOND.) TIME_UNIT());

/* Linear Stiffness Unit (kiloNewton per millimetre) */
#36 = LINEAR_STIFFNESS_UNIT ((#361, #362));
#361 = DERIVED_UNIT_ELEMENT (#363, 1.0);

```

```

#362 = DERIVED_UNIT_ELEMENT (#364, -1.0);
#363 = (FORCE_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.NEWTON.));
#364 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.MILLI.,.METRE.));

/* Linear Velocity Unit (metre per second) */
#37 = LINEAR_VELOCITY_UNIT ((#371, #372));
#371 = DERIVED_UNIT_ELEMENT (#373, 1.0);
#372 = DERIVED_UNIT_ELEMENT (#374, -1.0);
#373 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));
#374 = (NAMED_UNIT(*) SI_UNIT($,.SECOND.) TIME_UNIT());

/* Mass per Length Unit (kilogram per metre) */
#38 = MASS_PER_LENGTH_UNIT ((#381, #382));
#381 = DERIVED_UNIT_ELEMENT (#383, 1.0);
#382 = DERIVED_UNIT_ELEMENT (#384, -1.0);
#383 = (MASS_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.GRAM.));
#384 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

/* Modulus Unit (cubic centimetre) */
#39 = MODULUS_UNIT ((#391));
#391 = DERIVED_UNIT_ELEMENT (#392, 3.0);
#392 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.CENTI.,.METRE.));

/* Moment Unit (kiloNewton-metre) */
#40 = MOMENT_UNIT ((#401, #402));
#401 = DERIVED_UNIT_ELEMENT (#403, 1.0);
#402 = DERIVED_UNIT_ELEMENT (#404, 1.0);
#403 = (FORCE_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.NEWTON.));
#404 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));

/* Rotational Acceleration Unit (Radians per second per second) */
#41 = ROTATIONAL_ACCELERATION_UNIT ((#411, #412));
#411 = DERIVED_UNIT_ELEMENT (#413, 1.0);
#412 = DERIVED_UNIT_ELEMENT (#414, -2.0);
#413 = (NAMED_UNIT(*) PLANE_ANGLE_UNIT() SI_UNIT($,.RADIAN.));
#414 = (NAMED_UNIT(*) SI_UNIT($,.SECOND.) TIME_UNIT());

/* Rotational Stiffness Unit (kiloNewton-metre per radian) */
#42 = ROTATIONAL_STIFFNESS_UNIT ((#421, #422, #423));
#421 = DERIVED_UNIT_ELEMENT (#424, 1.0);
#422 = DERIVED_UNIT_ELEMENT (#425, 1.0);
#423 = DERIVED_UNIT_ELEMENT (#426, -1.0);
#424 = (FORCE_UNIT() NAMED_UNIT(*) SI_UNIT(.KILO.,.NEWTON.));
#425 = (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT($,.METRE.));
#426 = (NAMED_UNIT(*) PLANE_ANGLE_UNIT() SI_UNIT($,.RADIAN.));

```



```

/* Rotational Velocity Unit (Radians per second) */
#43 = ROTATIONAL_VELOCITY_UNIT ((#431, #432));
#431 = DERIVED_UNIT_ELEMENT (#433, 1.0);
#432 = DERIVED_UNIT_ELEMENT (#434, -1.0);
#433 = (NAMED_UNIT(*) PLANE_ANGLE_UNIT() SI_UNIT($,.RADIAN.));
#434 = (NAMED_UNIT(*) SI_UNIT($,.SECOND.) TIME_UNIT());
ENDSEC;

END-ISO-10303-21;

```

Definition of the US Imperial Unit System used with CIS/2

The following physical file listing contains the set of units that constitute the US Imperial Unit System that is used with CIS/2. It should be noted that this is a limited subset of the complete US Imperial Unit System. For example, the physical file below provides only one value for length (inch). Any derived units not defined in the physical file below that require a length component should be defined in terms of inches.

ISO-10303-21;

```

HEADER;
FILE_DESCRIPTION ( ('{cimsteel logical product model version (6) object (1) structural-
    frame-schema (1) }', 'CC035', 'US Imperial Units', '2;1') ;
FILE_NAME ('us_units.stp', '2003-05-07 T14:34:51', ('Andrew Crowley'), ('Steel Construction
    Institute'), 'Expresso 3.0.7 (compiled 2003-Feb-20 21:07)', 'Express Engine', 'AJC') ;
FILE_SCHEMA ( ('STRUCTURAL_FRAME_SCHEMA') ) ;
ENDSEC;

```

```

DATA;
#1 = REPRESENTATION ('Any representation that uses parameters defined in the US
    Imperial Unit System', (#2), #3);
#2 = REPRESENTATION_ITEM ('Any item of a representation that uses parameters
    defined in the US Imperial Unit System');
#3 = GLOBAL_UNIT_ASSIGNED_CONTEXT ('context for any items that use US Imperial
    Units', 'CIS/2 standardized units for the US Imperial Unit system', (#11, #21, #31, #41,
    #51, #60, #71, #81, #91, #101, #111, #121, #132, #141, #152, #162, #172, #182,
    #191, #202, #212, #223, #232));

```

/*

CIS/2 defines a constrained set of US Imperial units comprising 12 base units (defined in terms of metres, Kilograms, Seconds and Degrees Centigrade) and 11 other units derived from these base units.

*/

/* 12 Base Units */

/*

All are externally mapped complex instances of NAMED_UNIT with the subtype CONTEXT_DEPENDENT_UNIT for which the dimensionality must be specified by an associated DIMENSIONAL_EXPONENTS instance

Of the 7 ISO dimensionalities (Length, Mass, Time, current, temperature, amount of substance, luminosity) only 4 are used by the CIS.

*/

/* Length Unit (inch) */

#10 = DIMENSIONAL_EXPONENTS (1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0);

#11 = (CONTEXT_DEPENDENT_UNIT('INCH') LENGTH_UNIT() NAMED_UNIT(#10));

/* Mass Unit (pound) */

#20 = DIMENSIONAL_EXPONENTS (0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0);

#21 = (CONTEXT_DEPENDENT_UNIT('POUND') MASS_UNIT() NAMED_UNIT(#20));

/* Time Unit (Second) */

#30 = DIMENSIONAL_EXPONENTS (0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0);

#31 = (CONTEXT_DEPENDENT_UNIT('SECOND') NAMED_UNIT(#30) TIME_UNIT());

/* Thermodynamic Temperature (Degree Fahrenheit) */

#40 = DIMENSIONAL_EXPONENTS (0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0);

#41 = (CONTEXT_DEPENDENT_UNIT('DEGREE_FAHRENHEIT') NAMED_UNIT(#40)
THERMODYNAMIC_TEMPERATURE_UNIT());

/* Plane Angle Unit (Degree) */

#50 = DIMENSIONAL_EXPONENTS (0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0);

#51 = (CONTEXT_DEPENDENT_UNIT('DEGREE') NAMED_UNIT(#50)
PLANE_ANGLE_UNIT());

/* Solid Angle Unit (Degree) */

#60 = (CONTEXT_DEPENDENT_UNIT('DEGREE') NAMED_UNIT(#50)
SOLID_ANGLE_UNIT());

/* Area Unit (square inch) */

#70 = DIMENSIONAL_EXPONENTS (2.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0);

#71 = (AREA_UNIT()CONTEXT_DEPENDENT_UNIT('SQUARE_INCH')
NAMED_UNIT(#70));

/* Volume Unit (cubic inch) */

#80 = DIMENSIONAL_EXPONENTS (3.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0);

#81 = (CONTEXT_DEPENDENT_UNIT('CUBIC_INCH') NAMED_UNIT(#80)
VOLUME_UNIT());

/* Ratio Unit (Percent) */

#90 = DIMENSIONAL_EXPONENTS (0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0);

#91 = (CONTEXT_DEPENDENT_UNIT('PERCENTAGE') NAMED_UNIT(#90)
RATIO_UNIT());

/* Force Unit (kip) */

#100 = DIMENSIONAL_EXPONENTS (1.0,1.0,-2.0,0.0,0.0,0.0,0.0,0.0);

#101 = (CONTEXT_DEPENDENT_UNIT('KIP') FORCE_UNIT() NAMED_UNIT(#100));

/* Frequency Unit (Hertz) */

#110 = DIMENSIONAL_EXPONENTS (0.0,0.0,-1.0,0.0,0.0,0.0,0.0,0.0);

```

#111 = (CONTEXT_DEPENDENT_UNIT('HERTZ') FREQUENCY_UNIT()
        NAMED_UNIT(#110));

/* Pressure Unit (kip per square inch) */
#120 = DIMENSIONAL_EXPONENTS (-1.0,1.0,-2.0,0.0,0.0,0.0,0.0);
#121 = (CONTEXT_DEPENDENT_UNIT('KIPS_PER_SQUARE_INCH')
        NAMED_UNIT(#120) PRESSURE_UNIT());

/* 11 Derived Units */
/*
All are instances of SUBTYPES of DERIVED_UNIT. All have associated instances of
DERIVED_UNIT_ELEMENT to specify their dimensionality
*/
/* Force per Length Unit (kip per inch) */
#130 = DERIVED_UNIT_ELEMENT (#101, 1.0);
#131 = DERIVED_UNIT_ELEMENT (#11, -1.0);
#132 = FORCE_PER_LENGTH_UNIT ((#130, #131));

/* Inertia Unit (inch to the fourth power) */
#140 = DERIVED_UNIT_ELEMENT (#11, 4.0);
#141 = INERTIA_UNIT ((#140));

/* Linear Acceleration Unit (inch per second per second) */
#150 = DERIVED_UNIT_ELEMENT (#11, 1.0);
#151 = DERIVED_UNIT_ELEMENT (#31, -2.0);
#152 = LINEAR_ACCELERATION_UNIT ((#150, #151));

/* Linear Stiffness Unit (kip per inch) */
#160 = DERIVED_UNIT_ELEMENT (#101, 1.0);
#161 = DERIVED_UNIT_ELEMENT (#11, -1.0);
#162 = LINEAR_STIFFNESS_UNIT ((#160, #161));

/* Linear Velocity Unit (inch per second) */
#170 = DERIVED_UNIT_ELEMENT (#11, 1.0);
#171 = DERIVED_UNIT_ELEMENT (#31, -1.0);
#172 = LINEAR_VELOCITY_UNIT ((#170, #171));

/* Mass per Length Unit (pound per inch) */
#180 = DERIVED_UNIT_ELEMENT (#21, 1.0);
#181 = DERIVED_UNIT_ELEMENT (#11, -1.0);
#182 = MASS_PER_LENGTH_UNIT ((#180, #181));

/* Modulus Unit (cubic inch) */
#190 = DERIVED_UNIT_ELEMENT (#11, 3.0);
#191 = MODULUS_UNIT ((#190));

/* Moment Unit (kip-inch) */
#200 = DERIVED_UNIT_ELEMENT (#101,1.0);

```

```
#201 = DERIVED_UNIT_ELEMENT (#11,1.0);
#202 = MOMENT_UNIT ((#200, #201));

/* Rotational Acceleration Unit (degree per second per second) */
#210 = DERIVED_UNIT_ELEMENT (#51, 1.0);
#211 = DERIVED_UNIT_ELEMENT (#31, -2.0);
#212 = ROTATIONAL_ACCELERATION_UNIT ((#210, #211));

/* Rotational Stiffness Unit (kip-inch per degree) */
#220 = DERIVED_UNIT_ELEMENT (#101, 1.0);
#221 = DERIVED_UNIT_ELEMENT (#11, 1.0);
#222 = DERIVED_UNIT_ELEMENT (#51, -1.0);
#223 = ROTATIONAL_STIFFNESS_UNIT ((#220, #221, #222));

/* Rotational Velocity Unit (Degree per second) */
#230 = DERIVED_UNIT_ELEMENT (#51, 1.0);
#231 = DERIVED_UNIT_ELEMENT (#31, -1.0);
#232 = ROTATIONAL_VELOCITY_UNIT ((#230, #231));
ENDSEC;

END-ISO-10303-21;
```

5.4 Product Items

5.4.1 Types of Item

Of the products used in a steel frame, the great majority comply with some form of ‘standard’. Application software vendors frequently build libraries of such items into their applications. A goal of the international application of the CIS is to take advantage of this fact to be able:

- to transfer unambiguous references to such product items,

and thus

- to reduce size of data exchange files (and increase the efficiency of the data transfer).

This is realized by allowing as many product items as possible to be ‘passed-by-reference’ rather than being ‘passed-by-attribute-value’. To facilitate this, the CIS defines four classes of product item:

1. **Standard Items:** Internationally recognized *generic* product items (such as hot-rolled sections) that comply with formal standards that have international visibility. Such items are normally passed-by-reference in a CIS exchange file using agreed human readable ‘standard references’, published in the ‘Flavour lists’ that form part of the formal CIS documentation⁴.

⁴ The ‘Flavour lists’ are contained in physical files; electronic versions of these files are available on the CIS/2 web site (<http://www.cis2.org/>).

2. **Proprietary Items:** Manufacturer-specific *proprietary* product items (such as cold rolled sections) that comply with manufacturer's catalogues (would also include 'semi-standard' product items that comply with specifications agreed between a consortium of manufacturers). Such items are normally passed-by-reference in a CIS exchange file using agreed human readable 'proprietary reference' published by the manufacturer as 'manufacturer's lists'.
3. **Library Items:** Product items that are not explicitly covered in standard or manufacturer's catalogues. Third party trade associations may compile lists of product items and place them in a library for anyone to use.
4. **Non-standard Items:** These are product items that cannot be passed-by-reference because an agreed reference does not exist, or is not supported by both translators. These items are required to be passed-by-attribute-value. These will normally be true (bespoke) non-standard product items. Alternatively, providing they can be adequately characterised by the available attributes, these items may be standard items or manufacturer's items, for which a supported standard reference does not exist.

The LPM EXPRESS constructs that are used to represent product items and flavours are fully defined in Section 16 of *The Logical Product Model*^[5]. Vendors are advised to take particular note of these definitions before developing CIS translators.

5.4.2 Treatment of Standard Product Items

The treatment of Standard Product Items within the CIS is such that:

- When seen from the perspective of an **end-user**, the only requirement is that the translator on the importing application can support those standard references that were used by the translator on the exporting application. This means that:
 - i) The standard references that were used at export will be recorded within the export log file and the translator will issue appropriate warnings to the user in case of any missing information.
 - ii) The standard references that were recognized on import will be recorded within the import log file and the translator will issue appropriate warnings to the user in case of any mismatches in scope.
- When seen from the perspective of an **implementer**, standard references provide precise shorthand substitutes for specific data that may be imported or exported from their application, and which will be correctly mapped to and from the appropriate information at export or import.

If the native application of a CIS/2-conformant system is capable of creating data that represents a particular standard product, then the CIS/2 system will demonstrate 'support' for standard references if:

- its **export translator** can recognize the particular standard product item defined within the exporting application as being a standard item, associate an appropriate standard reference, and allow the user to exchange data by reference.
- its **import translator** can recognize that an entity instance in the incoming neutral data has been assigned a standard item reference and can map that reference onto a

particular standard product item in the importing application and provide values for all the native attributes appropriate to that standard product.

CIS/2-conformant systems are required to display appropriate warning messages if a standard item is encountered that is not supported. These messages will be displayed on the screen during the translation process and be written to the log file. (See Section 3.11.)

As discussed earlier in Section 3.5.3, it is a conformance requirement of CIS/2 that software vendors developing CIS translators will include a list of the standard references that their implementation supports in the documentation for the translator.

5.4.3 Translator support for Proprietary Product Items

The treatment of Proprietary Product Items within the CIS is such that:

- When seen from the perspective of an **end-user**, the only requirement is that the translator on the importing application can support those proprietary references that were used by the translator on the exporting application. This means that:
 - (i) The proprietary references that were used at export will be recorded within the export log file and the translator will issue appropriate warnings to the user in case of any missing information.
 - (ii) The proprietary references that were recognized on import will be recorded within the import log file and the translator will issue appropriate warnings to the user in case of any mismatches in scope.
- When seen from the perspective of an **implementer**, proprietary items provide precise shorthand substitutes for specific data that may be imported or exported from their application, and which will be correctly mapped to and from the appropriate information at export or import.

If the native application of a CIS/2-conformant system is capable of creating data that represents a particular manufacturer's product, then the CIS/2 system will demonstrate 'support' for proprietary references if:

- its **export translator** can recognize the particular standard product item defined within the exporting application as being a proprietary item, associate an appropriate proprietary reference, and allow the user to exchange data by reference.
- its **import translator** can recognize that an entity instance in the incoming neutral data has been assigned a proprietary item reference and can map that reference onto a particular proprietary product item in the importing application and provide values for all the native attributes appropriate to that manufacturer's product.
- **On import** - if an entity instance has been assigned an item reference and the receiving application supports the equivalent manufacturer's product, the importing application is required to provide values for all the native attributes appropriate to that manufacturer's product.

CIS/2-conformant systems are required to display appropriate warning messages if a proprietary item is encountered that is not supported. These messages shall be displayed on the screen during the translation process and written to the log file. (See Section 3.11.)

As discussed earlier in Section 3.5.3, it is a conformance requirement of CIS/2 that the documentation provided with a CIS/2-conformant system should include a list of the proprietary references that it supports. Software vendors are expected to support those proprietary item lists (and hence those manufacturer's catalogues) that are relevant to the market to which they are selling. It is not necessary for every CIS/2-conformant system to support every proprietary item list. Indeed, it is unlikely that the two translators involved in a particular data exchange will support identical proprietary lists. The only prerequisite for successful data exchange is that the importing translator can understand those proprietary items that are actually encountered in the exchange file and map them onto the appropriate information within the application.

5.4.4 Treatment of Library Items

Library items are treated in the same way as proprietary product items.

5.4.5 Treatment of 'Non-standard' Items

Since an item reference has not been defined, non-standard items have to be passed-by-attribute-value.

When dealing with standard, proprietary, or library items, CIS/2-conformant systems are required to offer the user the choice of either passing the information explicitly by attribute value, or implicitly by reference, or both. Thus, there will be occasions when standard, proprietary, or library items will be treated as non-standard items.

6 TEST CASES

A suite of test cases has been created for testing implementations claiming conformance to the CIS/2 specifications. Each test case was written for a given Conformance Class and represents a legitimate population of a valid subset of the CIS/2 EXPRESS schema (LPM/6). Each test case is used during formal conformance testing to ascertain whether the CIS implementation under test can support the given combination of conformance classes.

It should be noted that each test case represents only one of several thousand possible legitimate populations of that particular combination of conformance classes. The test cases are not intended to have any semantic value – they have been written merely to illustrate and test the syntax of certain LPM/6 constructs when encoded in a STEP Part 21 file.

A suite of STEP physical files has been created for the test cases and is available from the CIS/2 web site (www.cis2.org). Each physical file captures an individual test case. Although every care has been taken to ensure, to the best of our knowledge, that all data contained within these files are accurate to the extent that they conform to the published edition of the LPM/6 EXPRESS schema, The Steel Construction Institute, the authors and the reviewers assume no responsibility for any errors in or misinterpretations of such data or any loss or damage arising from or related to their use. Any errors should be reported directly to The Steel Construction Institute.

7 REFERENCES

1. CROWLEY, A.J.
The Development and Implementation of a Product Model for Constructional Steelwork
PhD Thesis, University of Leeds, UK, 1998
2. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 1 - Overview
SCI Publication P265, The Steel Construction Institute, UK, 2003
3. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 2 - Implementation Guide
SCI Publication P266, The Steel Construction Institute, UK, 2003
4. CROWLEY, A.J., WARD, M.A. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 3 - The Information Requirements
SCI Publication P267, The Steel Construction Institute, UK, (in preparation)
5. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 4 - The Logical Product Model (LPM/6)
SCI Publication P268, The Steel Construction Institute, UK, 2003
6. CROWLEY, A.J. & WATSON, A.S.
CIMsteel Integration Standards Release 2: Second Edition,
Volume 6 - Worked Examples
SCI Publication P270, The Steel Construction Institute, UK, (in preparation)
7. HAMBURG, S.E. & HOLLAND, M.V.
Leaping ahead with EDI, pp. 42-48 in *Modern Steel Construction*, Vol. 39, No. 2
American Institute of Steel Construction, Inc., Chicago, USA, February 1999
8. ISO 1000:1992
SI units and recommendations for the use of their multiples and of certain other units
ISO/IEC, Geneva, Switzerland, 1992
(Plus Amendment 1 published 1998-11-19)
9. ISO 10303-1: 1994
Industrial automation systems - Product data representation and exchange
Part 1: Overview and Fundamental Principles
ISO/IEC, Geneva, Switzerland, 1994
10. ISO 10303-11: 1994
Industrial automation systems - Product data representation and exchange
Part 11: The EXPRESS Language Reference Manual
ISO/IEC, Geneva, Switzerland, 1994
11. ISO 10303-21: 2002
Industrial automation systems - Product data representation and exchange
Part 21: Clear text encoding of the exchange structure
ISO/IEC, Geneva, Switzerland, 2002
(Incorporates Technical Corrigendum 1 published 1996-08-15)
12. ISO 10303-22: 1998
Industrial automation systems - Product data representation and exchange
Part 22: Standard Data Access Interface
ISO/IEC, Geneva, Switzerland, 1998

13. ISO 10303-41: 2000
Industrial automation systems - Product data representation and exchange
Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
14. ISO 10303-42: 2000
Industrial automation systems - Product data representation and exchange
Part 42: Integrated Generic Resources: Geometric & Topological Representation
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
15. ISO 10303-43: 2000
Industrial automation systems - Product data representation and exchange
Part 43: Integrated Generic Resources: Representation Structures
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
16. ISO 10303-44: 2000
Industrial automation systems - Product data representation and exchange
Part 44: Integrated Generic Resources: Product Structure Configuration
2nd Edition, ISO/IEC, Geneva, Switzerland, 2000
17. ISO FDIS 10303-225 (N710): 1997
Industrial automation systems - Product data representation and exchange
Part 225: Application protocol: Building Elements using Explicit Shape Representation
ISO/IEC, Geneva, Switzerland, 1997
18. ISO/WD 10303-230 (N551): 1996
Industrial automation systems - Product data representation and exchange
Part 230: Application protocol: Building Structural Frame: Steelwork
ISO TC184/SC4/WG3 (T12), 1996
19. ISO/IEC 11578: 1996
Information technology - Open Systems Interconnection - Remote Procedure Call (RPC)
ISO/IEC, Geneva, Switzerland, 1996
20. MICROSOFT INC
COM and ActiveX Object Services Microsoft Visual Studio Online Documentation
Microsoft Developer Network (<http://msdn.microsoft.com/>)
21. THE OPEN GROUP
DCE 1.1: Remote Procedure Call, Open Group Technical Standard, Document number
C706, August 1997. (<http://www.opengroup.org/publications/catalog/c706.htm>)

APPENDIX A CONFORMANCE CLASS LISTING

A.1 Generic Conformance Classes

Table A.1: *Generic Conformance Classes*

CC No	Schema Name	Base Entity	Subclass	Status
3	LPM5_CC003	cartesian_point	Geometry	Modified
8	LPM5_CC008	shape_representation	Geometry	Modified
12	LPM5_CC012	plane	Geometry	Unchanged
15	LPM5_CC015	surface	Geometry	Modified
16	LPM5_CC016	curve	Geometry	Modified
17	LPM5_CC017	line	Geometry	Unchanged
18	LPM5_CC018	bounded_surface	Geometry	Unchanged
22	LPM5_CC022	composite_curve	Geometry	Modified
23	LPM5_CC023	axis2_placement_2d	Geometry	Unchanged
24	LPM5_CC024	axis2_placement_3d	Geometry	Unchanged
25	LPM5_CC025	edge_curve	Geometry	Unchanged
28	LPM5_CC028	bounded_curve	Geometry	Unchanged
32	LPM5_CC032	direction	Geometry	Modified
33	LPM5_CC033	rectangular_trimmed_surface	Geometry	Unchanged
36	LPM6_CC036	block	Geometry	New
37	LPM6_CC037	geometric_representation_item	Geometry	New
2	LPM5_CC002	length_measure_with_unit	Units	Modified
5	LPM5_CC005	plane_angle_measure_with_unit	Units	Unchanged
9	LPM5_CC009	global_unit_assigned_context	Units	Modified
10	LPM5_CC010	shape_representation_with_units	Units	Unchanged
13	LPM5_CC013	time_measure_with_unit	Units	Unchanged
14	LPM5_CC014	positive_length_measure_with_unit	Units	Unchanged
19	LPM5_CC019	ratio_measure_with_unit	Units	Unchanged
20	LPM5_CC020	area_measure_with_unit	Units	Modified
21	LPM5_CC021	mass_measure_with_unit	Units	Unchanged
26	LPM5_CC026	measure_with_unit	Units	Modified
27	LPM5_CC027	thermodynamic_temperature_measure_with_unit	Units	Unchanged
29	LPM5_CC029	si_unit	Units	Modified
30	LPM5_CC030	context_dependent_unit	Units	Modified
31	LPM5_CC031	conversion_based_unit	Units	Unchanged
34	LPM5_CC034	si_unit (SI Unit System)	Units	Unchanged
35	LPM5_CC035	context_dependent_unit (US Imperial Unit System)	Units	Unchanged
125	LPM5_CC125	linear_stiffness_measure_with_unit	Units	Modified
126	LPM5_CC126	rotational_stiffness_measure_with_unit	Units	Modified
172	LPM5_CC172	force_measure_with_unit	Units	Modified
194	LPM5_CC194	moment_measure_with_unit	Units	Modified
195	LPM5_CC195	pressure_measure_with_unit	Units	Modified
196	LPM5_CC196	frequency_measure_with_unit	Units	Modified
197	LPM5_CC197	linear_acceleration_measure_with_unit	Units	Modified
198	LPM5_CC198	rotational_acceleration_measure_with_unit	Units	Modified
199	LPM5_CC199	linear_velocity_measure_with_unit	Units	Modified
200	LPM5_CC200	rotational_velocity_measure_with_unit	Units	Modified
234	LPM5_CC234	force_per_length_measure_with_unit	Units	Modified
260	LPM5_CC260	inertia_measure_with_unit	Units	Modified

CC No	Schema Name	Base Entity	Subclass	Status
3	LPM5_CC003	cartesian_point	Geometry	Modified
261	LPM5_CC261	modulus_measure_with_unit	Units	Modified
262	LPM5_CC262	derived_measure_with_unit	Units	Modified
263	LPM5_CC263	mass_per_length_measure_with_unit	Units	Modified
1	LPM5_CC001	date_and_time		Unchanged
4	LPM5_CC004	address		Modified
6	LPM5_CC006	group_assignment_approved		Modified
7	LPM5_CC007	group_assignment_actioned		Modified
11	LPM5_CC011	document_usage_constraint		Unchanged
38	LPM6_CC038	group		New
309	LPM5_CC309	coord_system_cartesian_2d		Unchanged
310	LPM5_CC310	coord_system_cartesian_3d		Unchanged
311	LPM5_CC311	coord_system_child		Unchanged

A.2 DMC Conformance Classes

Table A.2: Data Management Conformance Classes

CC No	Schema Name	Base Entity	Subclass	Status
100	LPM5_CC100	managed_data_item		Modified
101	LPM5_CC101	managed_data_item_with_history		Modified

A.3 Specific Conformance Classes

Table A.3: Analysis Conformance Classes

CC No	Schema Name	Base Entity	Subclass	Status
193	LPM5_CC193	applied_load_static_displacement	Loading	Unchanged
201	LPM5_CC201	applied_load_static_force	Loading	Unchanged
202	LPM5_CC202	applied_load_static_pressure	Loading	Unchanged
203	LPM5_CC203	applied_load_dynamic	Loading	Unchanged
204	LPM5_CC204	applied_load_dynamic_acceleration	Loading	Unchanged
205	LPM5_CC205	applied_load_dynamic_velocity	Loading	Unchanged
206	LPM5_CC206	physical_action	Loading	Modified
207	LPM5_CC207	physical_action_accidental	Loading	Unchanged
208	LPM5_CC208	physical_action_permanent	Loading	Unchanged
209	LPM5_CC209	physical_action_seismic	Loading	Unchanged
210	LPM5_CC210	physical_action_variable_long_term	Loading	Unchanged
211	LPM5_CC211	physical_action_variable_short_term	Loading	Unchanged
212	LPM5_CC212	physical_action_variable_transient	Loading	Unchanged
213	LPM5_CC213	load_case	Loading	Unchanged
215	LPM5_CC215	load_element_distributed_curve	Loading	Unchanged
216	LPM5_CC216	load_element_distributed_curve_line	Loading	Unchanged
217	LPM5_CC217	load_element_distributed_surface_uniform	Loading	Unchanged
218	LPM5_CC218	load_element_distributed_surface_varying	Loading	Unchanged
219	LPM5_CC219	load_element_thermal	Loading	Unchanged
221	LPM5_CC221	loading_combination	Loading	Unchanged
223	LPM5_CC223	load_case_documented	Loading	Unchanged
160	LPM5_CC160	analysis_method_documented	Modelling	Unchanged
161	LPM5_CC161	analysis_method_dynamic	Modelling	Unchanged
162	LPM5_CC162	analysis_method_pseudo_dynamic	Modelling	Unchanged

CC No	Schema Name	Base Entity	Subclass	Status
163	LPM5_CC163	analysis_method_static	Modelling	Unchanged
164	LPM5_CC164	analysis_model	Modelling	Unchanged
165	LPM5_CC165	analysis_model_2D	Modelling	Unchanged
166	LPM5_CC166	analysis_model_3D	Modelling	Unchanged
167	LPM5_CC167	analysis_model_located	Modelling	Unchanged
168	LPM5_CC168	analysis_model_child	Modelling	Unchanged
169	LPM5_CC169	analysis_model_mapping	Modelling	Unchanged
170	LPM5_CC170	assembly_map	Modelling	Unchanged
171	LPM5_CC171	analysis_model_relationship	Modelling	Unchanged
173	LPM5_CC173	boundary_condition_spring_linear	Modelling	Unchanged
174	LPM5_CC174	boundary_condition_warping	Modelling	Unchanged
175	LPM5_CC175	boundary_condition_spring_non_linear	Modelling	Unchanged
176	LPM5_CC176	boundary_condition_skewed	Modelling	Unchanged
177	LPM5_CC177	element_curve_simple	Modelling	Unchanged
178	LPM5_CC178	element_curve_complex	Modelling	Unchanged
180	LPM5_CC180	element_mapping	Modelling	Unchanged
181	LPM5_CC181	element_point_grounded_damper	Modelling	Unchanged
182	LPM5_CC182	element_point_grounded_spring	Modelling	Unchanged
183	LPM5_CC183	element_point_stationary_mass	Modelling	Unchanged
184	LPM5_CC184	element_surface_simple	Modelling	Unchanged
185	LPM5_CC185	element_surface_complex	Modelling	Unchanged
186	LPM5_CC186	element_surface_plane	Modelling	Unchanged
187	LPM5_CC187	element_surface_profiled	Modelling	Unchanged
188	LPM5_CC188	element_volume_simple	Modelling	Unchanged
189	LPM5_CC189	element_volume_complex	Modelling	Unchanged
190	LPM5_CC190	release_spring_linear	Modelling	Unchanged
191	LPM5_CC191	release_warping	Modelling	Unchanged
192	LPM5_CC192	release_spring_non_linear	Modelling	Unchanged
307	LPM5_CC307	element_with_material	Modelling	Unchanged
225	LPM5_CC225	reaction_acceleration	Response	Unchanged
226	LPM5_CC226	reaction_velocity	Response	Unchanged
227	LPM5_CC227	reaction_displacement	Response	Unchanged
228	LPM5_CC228	reaction_force	Response	Unchanged
229	LPM5_CC229	reaction_dynamic	Response	Unchanged
230	LPM5_CC230	reaction_equilibrium	Response	Unchanged
231	LPM5_CC231	analysis_result_element_curve	Response	Unchanged
232	LPM5_CC232	analysis_result_element_point	Response	Unchanged
233	LPM5_CC233	analysis_result_element_surface_stresses	Response	Unchanged
235	LPM5_CC235	analysis_result_element_surface_tractions	Response	Unchanged
236	LPM5_CC236	analysis_result_element_volume_stress_tensor	Response	Unchanged
237	LPM5_CC237	analysis_result_node	Response	Unchanged
238	LPM5_CC238	analysis_result_element_node	Response	Unchanged
239	LPM5_CC239	analysis_results_set	Response	Unchanged
240	LPM5_CC240	analysis_results_set_basic	Response	Unchanged
241	LPM5_CC241	analysis_results_set_combined	Response	Unchanged
242	LPM5_CC242	analysis_results_set_envelope	Response	Unchanged
243	LPM5_CC243	analysis_results_set_redistributed	Response	Unchanged
319	LPM5_CC319	group_of_analysis_data		Modified

Table A.4: *Design Conformance Classes*

CC No	Schema Name	Base Entity	Subclass	Status
292	LPM5_CC292	fastener	Joints	Unchanged
293	LPM5_CC293	joint_system_mechanical	Joints	Unchanged
294	LPM5_CC294	joint_system_welded	Joints	Modified
295	LPM5_CC295	joint_system_welded_point	Joints	Unchanged
296	LPM5_CC296	joint_system_welded_linear	Joints	Unchanged
297	LPM5_CC297	joint_system_welded_surface	Joints	Unchanged
298	LPM5_CC298	joint_system_chemical	Joints	Unchanged
299	LPM5_CC299	joint_system_amorphous	Joints	Unchanged
300	LPM5_CC300	joint_system_complex	Joints	Unchanged
301	LPM5_CC301	fastener_simple_bolt_hexagonal_head	Joints	Unchanged
302	LPM5_CC302	fastener_simple_nut_hexagonal	Joints	Unchanged
303	LPM5_CC303	fastener_simple_washer	Joints	Unchanged
304	LPM5_CC304	fastener_simple_shear_connector	Joints	Unchanged
352	LPM6_CC352	fastener_mechanism_with_position	Joints	New
353	LPM6_CC353	joint_system_welded_with_shape	Joints	New
354	LPM6_CC354	weld_mechanism_complex	Joints	New
355	LPM6_CC355	weld_mechanism_fillet_continuous	Joints	New
356	LPM6_CC356	weld_mechanism_fillet_intermittent	Joints	New
357	LPM6_CC357	weld_mechanism_groove_beveled	Joints	New
358	LPM6_CC358	weld_mechanism_groove_butt	Joints	New
359	LPM6_CC359	weld_mechanism_prismatic	Joints	New
360	LPM6_CC360	weld_mechanism_spot_seam	Joints	New
344	LPM6_CC344	load_member_concentrated	Loading	New
345	LPM6_CC345	load_member_distributed_curve	Loading	New
346	LPM6_CC346	load_member_distributed_surface_uniform	Loading	New
347	LPM6_CC347	load_member_distributed_surface_varying	Loading	New
248	LPM5_CC248	part	Parts	Unchanged
249	LPM5_CC249	part_prismatic_simple	Parts	Unchanged
250	LPM5_CC250	part_prismatic_complex	Parts	Unchanged
251	LPM5_CC251	part_prismatic_complex_tapered	Parts	Unchanged
252	LPM5_CC252	part_prismatic_simple_campered_absolute	Parts	Unchanged
253	LPM5_CC253	part_prismatic_simple_campered_relative	Parts	Unchanged
254	LPM5_CC254	part_prismatic_simple_castellated	Parts	Unchanged
255	LPM5_CC255	part_prismatic_simple_curved	Parts	Unchanged
256	LPM5_CC256	part_derived	Parts	Unchanged
257	LPM5_CC257	part_sheet	Parts	Unchanged
258	LPM5_CC258	part_sheet_bounded_simple	Parts	Unchanged
259	LPM5_CC259	part_sheet_bounded_complex	Parts	Unchanged
265	LPM5_CC265	part_sheet_profiled	Parts	Unchanged
266	LPM5_CC266	part_complex	Parts	Unchanged
102	LPM5_CC102	assembly_design		Unchanged
103	LPM5_CC103	assembly_design_child		Unchanged
104	LPM5_CC104	assembly_design_structural_connection		Unchanged
105	LPM5_CC105	assembly_design_structural_member		Unchanged
106	LPM5_CC106	assembly_design_structural_connection_internal		Unchanged
107	LPM5_CC107	assembly_design_structural_connection_external		Unchanged
108	LPM5_CC108	assembly_design_structural_frame		Unchanged
109	LPM5_CC109	assembly_design_structural_member_cubic		Unchanged
110	LPM5_CC110	assembly_design_structural_member_linear		Unchanged

CC No	Schema Name	Base Entity	Subclass	Status
111	LPM5_CC111	assembly_design_structural_member_planar		Unchanged
114	LPM5_CC114	assembly_relationship		Unchanged
115	LPM5_CC115	assembly_with_shape		Unchanged
117	LPM5_CC117	design_criterion_documented		Unchanged
118	LPM5_CC118	design_part		Unchanged
119	LPM5_CC119	design_joint_system		Unchanged
120	LPM5_CC120	effective_buckling_length		Unchanged
122	LPM5_CC122	functional_role_documented		Unchanged
124	LPM5_CC124	restraint_logical		Unchanged
127	LPM5_CC127	restraint_spring		Unchanged
128	LPM5_CC128	restraint_warping		Unchanged
220	LPM5_CC220	loaded_product		Unchanged
244	LPM5_CC244	design_result_part		Unchanged
245	LPM5_CC245	design_result_joint_system		Unchanged
246	LPM5_CC246	design_result_connection		Unchanged
247	LPM5_CC247	design_result_member		Unchanged
264	LPM5_CC264	section_properties		Unchanged
305	LPM5_CC305	material_isotropic		Modified
306	LPM5_CC306	material_strength		Unchanged
308	LPM5_CC308	structural_frame_product_with_material		Unchanged
320	LPM5_CC320	group_of_design_data		Modified
325	LPM5_CC325	section_profile		Unchanged
326	LPM5_CC326	section_profile_centreline		Unchanged
327	LPM5_CC327	section_profile_compound		Unchanged
328	LPM5_CC328	section_profile_derived		Unchanged
329	LPM5_CC329	section_profile_edge_defined		Unchanged
331	LPM5_CC331	item_reference_standard		Unchanged
332	LPM5_CC332	item_reference_proprietary		Unchanged
333	LPM5_CC333	item_property_assigned		Unchanged
340	LPM6_CC340	assembly_design_structural_member_linear_campered_absolute		New
341	LPM6_CC341	assembly_design_structural_member_linear_campered_relative		New
342	LPM6_CC342	assembly_with_bounding_box		New
368	LPM6_CC368	assembly_design_structural_member_linear_beam		New
369	LPM6_CC369	assembly_design_structural_member_linear_brace		New
370	LPM6_CC370	assembly_design_structural_member_linear_column		New

Table A.5: Manufacturing Conformance Classes

CC No	Schema Name	Base Entity	Subclass	Status
267	LPM5_CC267	feature	Features	Unchanged
268	LPM5_CC268	feature_cutting_plane	Features	Unchanged
269	LPM5_CC269	feature_edge_chamfer_straight	Features	Unchanged
270	LPM5_CC270	feature_edge_chamfer_fillet	Features	Unchanged
271	LPM5_CC271	feature_edge_chamfer_rounding	Features	Unchanged
272	LPM5_CC272	feature_surface_complex	Features	Unchanged
273	LPM5_CC273	feature_surface_simple	Features	Unchanged
274	LPM5_CC274	feature_surface_name_tag	Features	Unchanged
275	LPM5_CC275	feature_surface_treatment	Features	Unchanged

CC No	Schema Name	Base Entity	Subclass	Status
276	LPM5_CC276	feature_surface_with_layout	Features	Unchanged
277	LPM5_CC277	feature_thread	Features	Unchanged
278	LPM5_CC278	feature_volume_complex	Features	Unchanged
279	LPM5_CC279	feature_volume_curved	Features	Unchanged
280	LPM5_CC280	feature_volume_hole_circular	Features	Unchanged
281	LPM5_CC281	feature_volume_hole_circular_threaded	Features	Unchanged
282	LPM5_CC282	feature_volume_hole_rectangular	Features	Unchanged
283	LPM5_CC283	feature_volume_hole_slotted	Features	Unchanged
284	LPM5_CC284	feature_volume_hole_slotted_curved	Features	Unchanged
285	LPM5_CC285	feature_volume_prismatic_chamfer	Features	Unchanged
286	LPM5_CC286	feature_volume_prismatic_flange_notch	Features	Unchanged
287	LPM5_CC287	feature_volume_prismatic_flange_chamfer	Features	Unchanged
288	LPM5_CC288	feature_volume_prismatic_notch	Features	Unchanged
289	LPM5_CC289	feature_volume_prismatic_skewed_end	Features	Unchanged
290	LPM5_CC290	feature_volume_with_layout	Features	Unchanged
291	LPM5_CC291	feature_volume_with_process	Features	Unchanged
348	LPM6_CC348	feature_surface_point	Features	New
349	LPM6_CC349	feature_surface_point_mark	Features	New
350	LPM6_CC350	feature_volume_with_depth	Features	New
351	LPM6_CC351	feature_volume_with_limit	Features	New
343	LPM6_CC343	zone_of_structure_sequence_lot	MIS	New
361	LPM6_CC361	located_assembly_marked	MIS	New
362	LPM6_CC362	located_part_marked	MIS	New
363	LPM6_CC363	media_file_cnc	MIS	New
364	LPM6_CC364	media_file_drawing	MIS	New
366	LPM6_CC366	item_cost_code	MIS	New
371	LPM6_CC371	weld_arc	Process	New
112	LPM5_CC112	assembly_manufacturing		Unchanged
113	LPM5_CC113	assembly_manufacturing_child		Unchanged
179	LPM5_CC179	located_part		Unchanged
312	LPM5_CC312	located_assembly		Unchanged
313	LPM5_CC313	located_joint_system		Unchanged
315	LPM5_CC315	located_feature_for_located_part		Unchanged
316	LPM5_CC316	located_feature_joint_dependent		Unchanged
321	LPM5_CC321	located_part_joint		Unchanged
322	LPM5_CC322	group_of_physical_data		Modified
330	LPM5_CC330	structural_frame_item_priced		Unchanged
334	LPM5_CC334	project_process_item		Unchanged
335	LPM5_CC335	assemble		Unchanged
336	LPM5_CC336	procure		Unchanged
337	LPM5_CC337	surface_treatment_clean		Unchanged
338	LPM5_CC338	surface_treatment_coat		Unchanged
339	LPM5_CC339	surface_treatment_thermal_timed		Unchanged

Table A.6: *Project definition Conformance Classes*

CC No	Schema Name	Base Entity	Subclass	Status
129	LPM5_CC129	structure		Unchanged
130	LPM5_CC130	building		Unchanged
131	LPM5_CC131	site		Modified
132	LPM5_CC132	site_with_shape		Unchanged
133	LPM5_CC133	building_complex		Unchanged
134	LPM5_CC134	building_with_shape		Unchanged
135	LPM5_CC135	global_location		Unchanged
136	LPM5_CC136	map_location		Unchanged
137	LPM5_CC137	grid		Unchanged
138	LPM5_CC138	grid_intersection		Unchanged
139	LPM5_CC139	grid_intersection_resolved		Unchanged
140	LPM5_CC140	grid_of_building		Unchanged
141	LPM5_CC141	grid_of_site		Unchanged
142	LPM5_CC142	grid_of_structure		Unchanged
143	LPM5_CC143	grid_offset		Unchanged
144	LPM5_CC144	grid_orthogonal		Unchanged
145	LPM5_CC145	grid_radial		Unchanged
146	LPM5_CC146	grid_skewed		Unchanged
147	LPM5_CC147	organization_relationship_contractual		Unchanged
148	LPM5_CC148	project		Unchanged
149	LPM5_CC149	project_organization		Unchanged
150	LPM5_CC150	project_plan		Unchanged
151	LPM5_CC151	project_plan_item_relationship		Unchanged
152	LPM5_CC152	setting_out_point		Unchanged
153	LPM5_CC153	zone_bounded		Unchanged
154	LPM5_CC154	zone_of_building		Unchanged
155	LPM5_CC155	zone_of_building_storey		Unchanged
156	LPM5_CC156	zone_of_project		Unchanged
157	LPM5_CC157	zone_of_site		Unchanged
158	LPM5_CC158	zone_of_structure		Unchanged
317	LPM5_CC317	located_site		Unchanged
318	LPM5_CC318	located_structure		Unchanged
323	LPM5_CC323	group_of_project_definition_data		Modified
324	LPM5_CC324	group_of_structural_data		Modified
365	LPM6_CC365	project_data_group		New
367	LPM6_CC367	structural_frame_item		New