Product Data Exchange Specification
First Working Draft

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
(Formerly National Bureau of Standards)
National Engineering Laboratory
Center for Manufacturing Engineering
Factory Automation Systems Division
Gaithersburg, MD 20899

December 1988

National Bureau of Standards became the
National Institute of Standards and Technology
on August 23, 1980, when the Omnibus Trade and
Competitiveness Act was signed. NIST retains
all NBS functions. Its new programs will encourage
improved use of technology by U.S. industry.

Chair, IGES/PDES Organization
Bradford Smith

Coordinator, IGES/PDES Organization
Gaylen Rinaudot

U.S. DEPARTMENT OF COMMERCE
C. William Verity, Secretary
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Ernest Ambler, Director
Title: Classification Methods

Owner: Bradford Smith

Date: 31 October 1988

Corresponding ISO Document Number: N286
ABSTRACT:

This document presents two methods for classifying a STEP implementation: an implementation architecture view and an applications view. It incorporates material on implementation architectures (levels) and on application protocols.

KEY WORDS:

Application Protocols, Classification, Implementation Levels
Clause 6: Classification Methods

The technology embodied in this standard can be seen from two distinctly different viewpoints, from that of a user interested in a specific application use and from an implementer who is reducing the technology to software code to deal with exchange files or to implement a database. Each viewpoint has its own dimension, independent of the other, and is further segregated into finer categories. It is an enumeration of these viewpoint dimensions and categories that provides a taxonomy useful in understanding the planned utility of this standard.

Application Protocols

A basic objective of this standard is to achieve the successful transfer of the information required for a given application area. Information, be it in the form of a sentence or in the form of a digital product model, consists of syntax and semantics. The standard alone only defines the basic syntax and core semantics of the representation format.

In order to ensure complete and reliable information exchange, the application specific semantics and required functionality must also be documented and controlled. Application protocols are a formalized methodology for defining and controlling this semantic content.

The standard is designed to support a broad range of applications and information, and it is recognized that few software products will support all of the standard. Application protocols allow the definition of logical subsets of the standard and their usage, as well as providing necessary benchmarks for validating implementations. The exchange of information using an application protocol ensures that participating organizations agree on the types of information to be exchanged and employ corresponding information control procedures.

A protocol is a set of conventions or rules that govern the operation of functional units to achieve communication. The key concept of application protocols is to explicitly link the application area's information content to the entities and data structures to be exchanged. The procedure for developing application protocols involves identifying the information requirements of an application area and documenting them in an information model. This application information model is then used to select the corresponding subschema of the standard for representing the required information.

An application protocol defines the information content of a specific application area, specifies the mapping of the application information into data constructs, and describes the restrictions and conventions required in implementing these constructs. The key components of an application protocol are:

1. An application reference model
2. The selected subschema of the standard
3. The mappings between the model and the subschema
4. An application protocol format specification with a protocol usage guide
5. A well defined testing methodology with a set of application protocol format test cases.
Implementation Architectures

In the current world of product data exchange, a variety of systems are able to communicate their data only through standards whose implementation is seen through some type of standard exchange format file. Naturally, each system is implemented in a fashion that is suitable for its computer environment and although there are a wide variety of computer environments, common architectural components are found. Significant benefits in the data exchange process can be realized if common implementation components, which were developed because of common computer environments, are recognized and exploited.

Therefore, the purpose of distinguishing levels is to improve data exchange by understanding various computer system architectures. Architectures can be understood by virtue of being compared against a common architecture. Once it is available, this information is meant to be used by system developers who are responsible for designing data exchange/sharing architectures. Those system developers can use that information to recognize "connect points" between different architectures which can then be bridged for more efficient and effective data exchange. In order to bridge those "connect points", standards must be established at those interfaces.

A first assumption is that a variety of implementations are derivable from the same conceptual schema. In this case, all implementations must be derived from the integrated product data model (IPDM) conceptual schema defined in Clause 4. A second assumption is that the implementation levels must not be mutually exclusive. That is, a level 1 implementation must have a well defined mapping into a level 2, 3, or 4 implementation. It would be unacceptable for a level (i) implementation to be isolated and unable to communicate with a level (i+1) implementation.

Current Approach

At this time, three standard interfaces are being investigated. They are:

1. Exchange Format File (included in this document)
2. Access Software (a standard software binding)
3. Query Language (a standard definition/manipulation language)

Eleven criteria have been proposed to be used in the refinement of each interface. They are listed below.

1. Can the implementation read / write to the standard exchange format file?
2. Can the system generate / access the standard working form?
3. What is the granularity of the accessible data in this implementation?

   Can query on a single entity attribute
   Can query on multiple entity attributes
   Can query on a single entity
   Can query on multiple entities
   Can query on a single model
   Can query on multiple models
4. What standard forms of a data query are supported for this implementation?

- No standard query form supported
- Standard function calls only
- Standard interactive DML only
- Standard batch DML only
- All standard forms of query

5. What standard forms of data query response are supported for this application?

- No standard forms of data query response are supported
- Standard function's output parameters only
- Standard reply to a file
- Standard reply to an output device
- All standard forms of reply

6. What physical locations for the data can the implementation support?

- In local computer memory
- On local computer disk
- On local and/or remote computers

7. What are the logical navigational capabilities of the implementation against the integrated product data model?

- Hierarchical queries supported
- Network queries supported
- Relational queries supported
- Other

8. What are the constraint checking capabilities of the implementation?

IF IT IS PERFORMED

- No constraint checking
- Some constraint checking
- All constraint checking

WHAT

- Simple constraint checking
- Complex constraint checking

HOW

- Performed by hard-coded function calls
- Performed by executing external directives
- Performed by other means

WHEN

- Performed once when data is loaded into system
- Performed each time data is updated
- Performed each time data is used

9. What is the automation level of the data sharing process?

- Not automated (i.e. manual only)
- Partially automated (e.g. automatic notification, some manual steps)
- Fully automated (updates occur automatically, always current data)
10. Can the implementation accept/use User-Defined data?

Cannot accept User-Defined data
Can accept, store and retrieve for downstream processes but cannot use
Can accept and use User-Defined data

11. What configuration control capabilities are supported?

No configuration controls provided
Limited configuration controls available, if so, what kinds?

The Current Working Definitions

Although not fully comprehensive, the following implementation level definitions are useful in classifying implementations of this standard.

LEVEL 1  FILE EXCHANGE

Product data is translated into or out of the standard exchange format file using non-standard software. The data in the exchange file is derived from the IPDM. The granularity of the data in the exchange file will range from the multi-model level to the entity level. No standard forms of query are defined for the product data when in this form. No standard navigational capabilities are defined for the product data when in this form. No standard validity constraint checks are defined for the product data when in this form. The automation level of the data sharing process is not defined when the product data is in this form.

LEVEL 2  WORKING FORM EXCHANGE

Product data is translated into or out of the standard working form. The product data in the working form is derived from the IPDM. The granularity of the data in the working form will range from the model level to the entity level. There must be a standard mapping of product data from the exchange format file form. Product data in the working form is accessed via standard access software function calls. The access software must support relational, network, or hierarchical types of queries against the product data. The access software has no database capabilities beyond data manipulation and navigational capabilities. The access software does not enforce validity constraints. The automation level of the data sharing process is not defined when the product data is in this form.

LEVEL 3  DATABASE EXCHANGE

Product data is translated into or out of a Database Management System (DBMS). The product data in the DBMS is derived from the IPDM. The granularity of the data in the DBMS will range from the multi-model level to the entity level. The DBMS must be accessible by standard exchange format files, standard access software function calls, or standard Data Manipulation Language statements. The DBMS must support relational, network, or hierarchical types of queries against the product data. The DBMS will not enforce validity constraints. The data sharing process is fully automated when the product data is in this form.
LEVEL 4

KNOWLEDGEBASE EXCHANGE

Product data is translated into or out of a Knowledgebase Management System (KBMS). The product data in the KBMS is derived from the IPDM. The granularity of the data in the KBMS will range from the multi-model level to the entity level. The KBMS must be accessible by standard exchange format files, standard access software function calls, or standard Data Manipulation Language statements. The KBMS must support relational, network, or hierarchical types of queries against the product data. The KBMS will enforce all of the validity constraints specified in the standard. The data sharing process is fully automated when the product data is in this form.

References

INFORMATION MODELING LANGUAGE EXPRESS

DOCUMENT NUMBER: ANNEX A

TITLE: Information Modeling Language EXPRESS

ABSTRACT:

This document has the same technical content as WG1 N268 but has been reprinted in LaTeX. The preface to N268 has been removed, page headings have been changed and minor changes to paragraph numbering have been made to conform to ISO document style.

KEY WORDS:

Data Definition Language, Grammar, Information Models, Language, Syntax, Universe of Discourse

DATE: 31 October 1988
OWNER: Douglas Schenck
PHONE NUMBER: (314) 234-5258
FAX NUMBER: (314) 777-1704
ISO REPRESENTATIVE: Douglas Schenck
STATUS: Draft
Title: Information Modeling Language EXPRESS

Owner: Douglas Schenck

Date: 31 October 1988

Corresponding ISO Document Number: N287
Annex A The Express Language

A.1 Introduction to Express

This section introduces the major concepts on which the Express language is based and explains the objectives and requirements established for an information modeling language.

A.1.1 Purpose

The goal of an information modeling endeavor is to precisely describe some subset of the world we know.

Express is a language that allows us to formulate a precise information specification in terms that people can understand, and that computers can directly employ.

A.1.2 Formal Languages

A language such as English is a means of communicating ideas. English can be written, spoken, or signed. Meaning is largely independent of the method of articulation, although some communication methods are more robust and precise than others.

Express is a specialized language used to communicate about a subset the knowledge we have about the world around us. This language allows us to say what we know about information that may be used by an information system.

Express provides the words, syntax, and grammar needed to describe a Universe of Discourse in a uniform, precise, and compact manner. English could be used for this purpose, but English (or any natural language) is anything but uniform, precise, or compact. A formal structured language is needed to restrict meanings in order that a high degree of common understanding can be achieved.

Express is not itself a methodology. A methodology is an orderly process that helps to accomplish some end objective. But Express is a formal language that could be used as part of a methodology aimed at creating an information model.

A.1.3 Objectives

The design of Express was guided by three basic objectives:

- to provide a lot of expressive power,
- to enable the separation of abstract and concrete ideas, and
- to treat information units as general information resources.
A.1.4 Expressive Power

Producing an information model of a complex subject area like Product Data demands a lot of expressive power. The first objective of Express is to allow knowledgeable people to formally document their knowledge. That formal expression can then be used by a variety of people. The people that will read Express are likely to have varying degrees of understanding about the subject.

A.1.5 Separation of Abstract and Concrete Ideas

Every user will not have the same need for detail. Experts in a subject area will demand a greater degree of detail than those casually interested the subject. The separation of abstract from concrete views helps to tailor the specification to the needs of the user.

The separation of abstract and concrete concepts requires a different vocabulary. The vocabulary used to express abstract ideas is, by intention, vague. The vocabulary used to express concrete ideas is, by necessity, exact. For example, during the abstract stage we might say that the X component of a point is simply a number. The concrete view might be that X must be an integer number with nine digits of precision, or that it must be a real (floating point) number of some kind.

A.1.6 General Information Resource

An entity represents some idea that is important to the UoD being described. The context in which an entity is used should not alter its meaning, nor should the evolution of the information model over time.

The idea that an information object has to be a stable resource is of paramount importance to the orderly growth of this standard.

A.1.7 Requirements

Express is designed to satisfy the four basic requirements described below. Other requirements such as ease of use are addressed also, but are not discussed in detail in this document.

The four basic requirements are:

- modeling the things of interest,
- defining the constraints that are to be imposed upon those things,
- defining the operations in which those things participate, and
- modeling in a computer sensible manner.
A.1.8 Modeling of Things

An entity is something of interest. An entity represents an idea and is defined in terms of a set of attributes by which the idea is realized. The set of attributes that make up an entity often do not have a one to one correspondence to the physical record structures found in a database.

Abstraction separates the idea an entity represents from the set of attributes chosen to represent it. For example, point is an idea, and X, Y, Z are the names of the attributes commonly used to realize that idea. It is important to be able to deal with either the idea or the details that make up the idea separately and selectively. The term abstract is used for the idea that the entity represents. The term concrete is used for the details of which the entity is composed.

The abstract component provides a high level view of the schema. The most important information is raised to a level where it may be seen easily. The details of how those things are defined are hidden. At the abstract level the idea of a geometric line is important, not the particulars of how the line is defined. At the abstract level it is not even important that the line has a concrete definition.

The concrete component reveals the ingredients that make up the definition of an entity. At the concrete stage of the development of a schema it becomes important to fix on a set of attributes that define an entity.

Each attribute is made up of two components: a type and a name that describes the role that the type plays in the definition of an entity. For example, a number plays the role of the X component of a Point entity. A type is a realization of the role. At the abstract level these types have little or no meaning. However, at the concrete level, types are necessary to shape the choices made about how a particular entity is viewed by the enterprise. For example, at the abstract level, "circle" is self-evident. A circle is a plane curve everywhere equidistant from a given fixed point, its center. However, this definition is not very useful when the circle is processed by a computer. Choices must be made about the set of attributes necessary to realize this concept. Therefore, we might elect to use attributes named Center, Axis, and Radius, where Center is a point (type), Axis is a vector, and Radius is a floating point number.

A well chosen role name has an abstractive quality. The role name Radius is abstract. The fact that Radius is to be represented as a floating point number is concrete. At some stage in the development of a schema, a specific type must be associated with each attribute.

A.1.9 Modeling of Rules

A rule (constraint) stipulates a condition which must be true within a UoD. That is, it establishes a fact. A database is in a valid state when every rule is true.

Rules can be written to stipulate a variety of conditions:

- discrete values; e.g., radius cannot be less than zero.
- combinations of discrete values; e.g., June has no more than 30 days.
• uniqueness; e.g., every employee identification in a database must be different; no point can be the same as any other point in the definition of a spline.

• cardinality; e.g., for each spline there must be at least two, but not more than 30 control points; no edge can exist unless it is used in the definition of exactly two loops.

• a variety of different global constraints; e.g., the sum of the debits must equal the sum of the credits; there must be one registered nurse on duty for every eight occupied beds; the machine tool spindle direction must agree with the handedness of the cutter; etc.

A rule written as part of an entity declaration is called a local rule. Local rules apply only to the values of those attributes that exist within an entity instance or to values accessed through a defining relationship. Local rules can deal with single attributes or combinations of attributes.

A rule written as part of a schema is called a global rule. Global rules potentially deal with every instance of every type of entity to which the rule applies. For example, a global rule might say that at least one point must exist at the origin. An examination of the set of point instances might find an origin point right away, but it could be that every point in the set has to be examined to determine whether the rule is satisfied or not.

The rules described in an Express schema define a valid end state. The state of an Information System during a transition from one state to another is not addressed. Furthermore, a rule does not address the problem of recovering from a rule violation.

A.1.10 Modeling of Operations

An operation establishes how an entity may be employed within the Universe of Discourse (UoD). Operations on numbers such as addition, subtraction, multiplication, and division are universal. The set of operations that apply to numbers provide a framework for all who wish to employ mathematics. The same reasoning applies to geometric elements, bills of material, and other product data. Once a set of operations is defined for an entity, that knowledge becomes a valuable resource to those who wish to deal with product definition data in a uniform manner.

An operation often has a symbolic representation. Addition is usually written in infix notation as in:

\[
\text{number} + \text{number}
\]

The two numbers are the operands and the + symbol is the operator.

In computer programming languages, operations not accounted for symbolically are written as functions. For example, a square root operation is often implemented as a function. Functions are sometimes used instead of operations because keyboards do not have appropriate symbols.

Various programming languages implement the same operation in different ways. Pascal implements the modulo operation as the compound symbol MOD (X MOD Y) while PL/I implements the same operation as the built-in function MOD(X,Y). In either case, the built-in operations almost always permit generic inputs, i.e., either SQRT(10) or SQRT(10.0) or SQRT(1.E1) is
permitted. The compiler recognizes the input type and invokes the appropriate algorithm.

*Express* allows operations to be written as functions. Generic types can be used to avoid the problem of developing a specific function for each permissible input type.

A.1.11 Computer Processable

The problem of constructing an information model of Product Data is too large to manage without computer assist. For this reason, the modeling language must be processable by a computer while still being usable by people. The effect of this requirement is that *Express* must be a structured language; computers cannot deal effectively with unstructured input such as natural language or graphics. Alternate forms of *Express* may be generated as a by-product of computer processing to aid human understanding. Some of these alternate forms of output could be syntactic and semantic analysis, graphics, a table of contents or index, an implementation data structure, or even a form of natural language.

A.1.12 Overview of the Elements of Express

The main descriptive elements of *Express* are:

```
Schema
  Type
  Entity
  Algorithm
    Function
    Procedure
  Rule
```

The schema is a description of a UoD and is composed of declarations of types, entities, functions and the like. The schema is roughly similar to a description of a database.

A type describes the characteristics of the data that is used to represent entities.

The entity describes the things of interest to a UoD. These elements are roughly similar to records that occur in a database.

Rules define the constraints on entities and attributes that must be enforced by the information system.

Functions, procedures, and rules are used to describe the behavior of things.
A.1.13 The Schema

A schema is a description of all of the things of interest to a UoD. The entities declared in a schema will later appear as record occurrences in a database (although not necessarily on a one-to-one basis). Rules and operations will exist as part of application programs or database management systems.

A.1.14 Types

A type is a representation of data. Express provides a number of built-in types as well as a facility for defining extensions to the built-in type set. These built-in types are grouped into the following categories: base types, aggregation types, entity types, function types, and enumeration types.

- base types are the stuff of which databases are built: real and integer numbers, logical (truth values), and (character) strings.
- aggregation types define collections of like things. They are the array type, list type, and set type.
- entity may be used as a data type.
- function may be used as a type to show how attributes may be derived from other attributes.
- enumeration type specifies, in order, all of its possible values of something by name.

A.1.15 Entity

An entity is an object, concept, or idea that has meaning to the UoD. It represents an information unit, which may be exchanged between users.

The name given to an entity should describe the role it plays within the UoD. An entity is defined by a set of attributes. The behavior of an entity is defined by a set of rules. Its purpose is defined by the operations performed on it.

An attribute is either a fact about an entity or the data from which facts may be derived. Each attribute has a name that is unique within the scope of an entity. The name (of the attribute) should describe the role the it plays in the definition of an entity. An attribute is defined in terms of a type. The type defines how the role is realized. The behavior of an attribute is defined by the characteristics of its type and any additional rules used to further constrain its behavior.

A.1.16 Algorithms - Function and Procedure

Algorithms are an integral part of a schema and are used to define the constraints that govern entities and attributes. The liberal use of algorithms in a schema contribute to the understanding of entities and their behavior.
An algorithm is a sequence of operations on data that produces an end state. The end state depends on whether the algorithm is a function, procedure, or rule.

- the function yields a type as a result,
- the procedure may operate on data in a variety of ways.
- the rule operates on the model as a whole, and signals any global inconsistencies in it.

A.1.17 Reference Manual

The Reference Manual explains how the elements of Express are used. Each element is presented in its own context with examples. Simple elements of Express are introduced first, and more complex ideas are presented in an incremental manner.

1. Character Set
2. Operators
3. Keywords
4. Identifiers
5. Literals
6. Directives
7. Types
8. Built-in Constants
9. Built-in Functions
10. Built-in Procedures
11. Statements and Blocks
12. Expressions

Note - See Part iii (the Syntax Specification) for a description of the notation used for the syntax examples in this section.

A.2 Character Set

The Express language is written as a string of characters which conform to certain rules. These characters, except for whitespace, are combined to form tokens. Tokens are called symbols, keywords, identifiers, or literals. The base character set used to write Express is:

A.2.1 Digits

The Arabic digits:

\[ 0 1 2 3 4 5 6 7 8 9 \]
A.2.2 Letters

The lower and upper case letters of the English alphabet:

```
abcdefgijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

A.2.3 Underscore

The underscore character can be used as a letter when forming identifiers, except the underscore cannot be used as the first character.

A.2.4 Special Characters

```
# ( ) + - . / : < > [ ] _ { } |
```

A.2.5 Whitespace

Whitespace can be used to separate tokens and is often used to improve the structure and readability of the written language, i.e., to indent sections or leave blank lines. The liberal use of whitespace is encouraged.

A.2.5.1 Space Character

One or more blank spaces can appear between two tokens.

A.2.5.2 Tab Character

The tab character is treated as a space.

A.2.5.3 Newline

The newline marks the physical end of a source line. If the record is variable length, the newline is a sequence of non-printable characters. For fixed length records the newline is the physical end of the record. In either case, newline is treated as a space. Newline is significant only when it terminates a tail remark or abnormally terminates a string literal.

A.2.5.4 Remark

A remark is treated as whitespace. There are two forms of a remark. The embedded remark can be placed between any two tokens. The tail remark must be at the end of a line. In either case, a remark is treated as documentation and is not processed as part of the language.
A.2.5.4.1 Embedded Remark

The character pair '(*' denotes the start of an embedded remark and the character pair '*)' denotes its end. An embedded remark can appear between any two tokens.

Any character can be written between the start and end of a remark. Embedded remarks can span several physical lines.

Embedded remarks can be nested. For example, the following remark would be handled successfully.

(* The (* symbol starts a remark, and *) ends it *)

Example:

(* anything can go here *)

A.2.5.4.2 Tail Remark

The tail remark must be written at the end of a physical line. The '--' character pair starts the tail remark and a newline terminates it.

Example:

-- this is a remark that ends with a newline

A.3 Operators

Operators denote some action that is performed on an operand or a pair of operands.
The relationship *cardinality* operator produces a logical result given objects that participate in a relationship. It is not written as a single token.

Note - Some operators such as * are overloaded; i.e., the meaning of the operator depends on the operands and the operator. The * operator means addition when used with number operands, but it means inclusion when used with sets and set elements. The exact usage of operators is explained for each type.

### A.4 Reserved Words

The reserved words of *Express* are either keywords or the names of built-in constants, functions, or procedures. None of the reserved words can be used as an identifier. The reserved words of *Express* are shown below. A reserved word can be written using any combination of upper and lower case letters.

**Reserved Words**

<table>
<thead>
<tr>
<th>AGGREGATE</th>
<th>AND</th>
<th>ARRAY</th>
<th>AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSUME</td>
<td>BEGIN</td>
<td>BY</td>
<td>CASE</td>
</tr>
<tr>
<td>DEFINE</td>
<td>DERIVE</td>
<td>DIV</td>
<td>DYNAMIC</td>
</tr>
<tr>
<td>ELSE</td>
<td>END</td>
<td>END_CASE</td>
<td>END_ENTITY</td>
</tr>
<tr>
<td>END_FUNCTION</td>
<td>END_IF</td>
<td>END_LOCAL</td>
<td>END_MAP</td>
</tr>
<tr>
<td>END_PROCEDURE</td>
<td>END_REPEAT</td>
<td>END_RULE</td>
<td>END_SCHEMA</td>
</tr>
<tr>
<td>END_TYPE</td>
<td>ENTITY</td>
<td>ENUMERATION</td>
<td>ESCAPE</td>
</tr>
</tbody>
</table>
Built-in Functions

<table>
<thead>
<tr>
<th>ABS</th>
<th>ACOS</th>
<th>ASIN</th>
<th>ATAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>EXISTS</td>
<td>EXP</td>
<td>HIBOUND</td>
</tr>
<tr>
<td>LOBOUND</td>
<td>LOG</td>
<td>LOG10</td>
<td>LOG2</td>
</tr>
<tr>
<td>ODD</td>
<td>SIN</td>
<td>SIZEOF</td>
<td>SQRT</td>
</tr>
<tr>
<td>TAN</td>
<td>TYPEOF</td>
<td>VALUE</td>
<td></td>
</tr>
</tbody>
</table>

Built-in Procedures

<table>
<thead>
<tr>
<th>INSERT</th>
<th>NULLIFY</th>
<th>VIOLATION</th>
</tr>
</thead>
</table>

Built-in Constants

<table>
<thead>
<tr>
<th>#</th>
<th>FALSE</th>
<th>PI</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note – The "pound sign" character represents the notion of an indeterminate bound in an aggregate type.

A.5 Identifiers

Identifiers are names given to the elements being defined in a schema. An identifier must not be the same as an Express reserved word.
The first character of an identifier must be a letter. The remaining characters, if any, can be any combination of letters, digits, or the underscore character. Letters are not case sensitive as upper and lower case letters are treated as equal.

Identifiers have scope. An identifier can be declared only once within the same scope.

Sometimes it is necessary to "qualify" an identifier. This occurs when addressing an attribute of an entity or when specifying a particular array or list elements. The dot character is used to qualify attribute names. Array, List, and Set elements are specified by enclosing a numeric expression within square brackets.

Syntax

148 ID  =  ID-SIMPLE | ID-QUALIFIED .
152 ID-SIMPLE  =  LETTER { ID-CHAR } .
149 ID-CHAR  =  LETTER | DIGIT | UNDERSCORE .
151 ID-QUALIFIED  =  ID-INDEHXED { DOT ID-INDEHXED } .
150 ID-INDEHXED  =  ID { SUBSCRIPT } .

Example:

Valid Identifiers

POINT  line  Circle  Rational_B_Spline
item406

Valid Qualified Identifiers

Point.x  Z[i=2]

Invalid Identifiers

_POINT underscore can't be first character
line?  ? can't be part of identifier
3dcircle  digit can't be first character
schema  schema is an Express keyword

A.5.1 Simple Identifiers

A simple identifier is any identifier that has no qualification.
A.5.2 Qualified Identifiers

Qualified identifiers must be used when it is impossible to determine the scope of a simple identifier or when an element is subscripted as in an array.

Example:

\[ x = x + y.c[37]; \quad -- y.c[37] \text{ is a qualified identifier.} \]

Note - See the Function Call, which can also be qualified.

A.5.3 Scope of Identifiers

A particular kind of object is created by a declaration of an identifier at some place in a schema. When that identifier is used again, it must be within the scope of its declaration.

The scope of an identifier is that part of the schema in which a reference to an identifier refers to the same object the declaration created. A scope encompasses its declaration to the end of the current block, including all blocks declared within the current block.

The blocks that establish a scope in this manner are:

1. Schema
2. Entity
3. Map
4. Function
5. Procedure
6. Rule

An identifier can be redeclared in another block. That is, the same name can be used in another context. In such a case, the identifier represents some entirely different thing and it should not be confused with any other like identifier. A given identifier cannot be redeclared within the same block however. For example, if there is an entity called \( e1 \) in schema \( s1 \), the name \( e1 \) cannot be redeclared in \( s1 \) for any purpose.

An identifier cannot normally be seen outside its scope. The Assume directive can be used to overcome this restriction.

Example:

\[ \text{Universe} \quad \text{Built-in Constants} \]
\[ \text{Built-in Functions} \quad \text{Built-in Procedures} \]
The outermost schema (S1) exists within the "universe", which can be imagined as a schema that encloses every other schema. All of the built-in constants, types, functions, and procedures can be thought of as having been declared within the universe. As a consequence, those built-in elements can be referenced from anywhere.

Entity e1 is declared in the scope established by Schema s2 and entity e2 is declared in the scope established by Schema s3. The attribute named a2 in entity e2 attempts to use e1 as its type. That is illegal since e1 is not in the scope of s3.

The Assume directive can be used to cross normal scope boundaries. Had Schema S3 contained the directive "Assume (S2);" the declarations in the scope of S2 could have been seen within S3.

A.6 Literals

A literal is a self-defining constant value. The type of a literal depends on how characters are composed to form a token. The literals are integer, real, string, or logical.

A.6.1 Integer Literal

An integer literal is composed entirely of digits. It defines an integer (whole) number.

Syntax
155 INTEGER-LITERAL = DIGIT { DIGIT }.

Example:

4016
38

A.6.2 Real Literal

A real literal is composed of a mantissa and an optional exponent. It defines a fractional number.

Syntax

169 REAL-LITERAL = INTEGER-LITERAL DOT [ INTEGER-LITERAL ] [ 'E' [ SIGN ] INTEGER-LITERAL ].

Example:

1.e6
3.5e-5
359.62

Illegal Real Literal

.001 At least one digit must precede the decimal point
1e10 A decimal point must be part of the literal

A.6.3 String Literal

A string literal is a sequence of characters enclosed by quote marks (apostrophes). If a quote mark is itself part of the string, two consecutive quote marks must be written. A string literal cannot span a physical line.

Syntax

176 STRING-LITERAL = \Q { \Q | ( \Q \Q ) } \Q.

Note - \a represents any character except a newline; \q represents a quote mark.

Example:

'Baby needs a new pair of shoes!'
Reads ... Baby needs a new pair of shoes!

'Ed''s Computer Store'
Reads ... Ed's Computer Store
Illegal String Literal

'Ed's Computer Store'
There must always be an even number of quote marks.

A.6.4 Logical Literal

A logical literal is any of the words 'True', 'False', or 'Unknown'.

Syntax

160 LOGICAL-LITERAL = TRUE | FALSE | UNKNOWN .

Example:

x := true;
y := false;
z := unknown;

A.7 Directives

The compiler directives allow the schema writer to control the behavior of the compiler that processes an Express source file. The Assume directive is used to override normal scoping rules. Define directives allow identifiers to stand for a substitution string. The Export directive allows a schema to control what may or may not be seen outside its scope. Include directives allow separate source to be combined into a single logical file. The Uses directive allows individual declarations in some other schema to be used as if they are part of another schema.

A.7.1 Assume Directive

Description:

The Assume directive is used to gain access to declarations that are not normally known via ordinary scoping rules. The directive specifies one or more schema names. When an identifier reference is unresolved, an attempt will be made to resolve that identifier in the schemas mentioned by the Assume directive.

Note that the author of a schema can use the Export directive to prevent other schemas from using an identifier, even when the other schema uses the Assume directive.

Syntax

184 ASSUME-DIRECTIVE = ASSUME LEFT-PAREN SCHEMA-ID { COMMA SCHEMA-ID } RIGHT-PAREN SEMICOLON .
Example:

See the example given in 6.3. (Export Directive).

Restrictions:

- The ASSUME directive must appear before any identifier references that require the assumed schemas to resolve those identifiers.

A.7.2 Define Directive

Description:

A lexical constant is created by the Define directive. It equates an identifier to a sequence of characters which must be enclosed by quotes. Whenever this lexical constant identifier is encountered, the sequence of characters equated to it are substituted in its place. The substitution can consist of zero or more characters.

Syntax

```
193 DEFINE-DIRECTIVE = DEFINE DEFINED-TYPE-ID STRING-LITERAL SEMICOLON .
```

Example:

```
DEFINE muchprecision 'REAL(30)';
```

Whenever the identifier muchprecision is seen in the source file the character string REAL(30) is substituted for it. The enclosing quotes are not part of the substitution and the substitution can consist of zero or more characters. Thus if the following line appeared in the schema:

```
x : muchprecision ;
```

the line would be interpreted as:

```
x : REAL(30) ;
```

A substitution string can contain other defined lexical constants as in:

```
DEFINE a '30';
DEFINE b 'real(a)';
```

then

```
x : b;
```

yields:
x : real(30);

Restrictions:

- A Define directive must appear before a reference to a reference to the identifier.
- Circular defines are prohibited. If substitution string A references identifier B, then substitution string B cannot reference identifier A either directly or indirectly.

A.7.3 Export Directive

Description:

The Export directive is used to allow access to objects declared within a schema. If a schema does not supply an Export directive, nothing can be exported; i.e., seen outside its scope.

Syntax

46 EXPORT = 'EXPORT' .

Restrictions:

1. Indirect declarations cannot be exported. For example, a function declared within a function cannot be exported in this manner.

Example:

The following example shows how the Export and Assume directives work together. The schema called main encompasses two inner schemas: inside1 and inside2. Some of the declarations in inside1 are private and others are public (exported). Declarations in inside2 reference entities declared in inside1.

```
SCHEMA MAIN;
ENTITY X;  <-----------------------------+
END_ENTITY;
SCHEMA INSIDE1;
  EXPORT (I11, I12);
  ENTITY I11;  <---------------------+
  END_ENTITY.
  ENTITY I12;
  END_ENTITY.
  ENTITY I13;  <--------+
  END_ENTITY.
END_SCHEMA;
```
SCHEMA INSIDE2;
ASSUME (INSIDE1);
ENTITY 121;
  A : X; (1)----------------->
END_ENTITY;
ENTITY 122;
  A : I13; (2)------>
END_ENTITY;
ENTITY 123;
  A : I11; (3)----------------->
END_ENTITY;
END_SCHEMA;

- The reference to (entity) X is resolved according to normal scoping rules.
- The reference to (entity) I13 is ILLEGAL! Even though I13 is declared within INSIDE1, which is assumed, I13 was not marked as exportable; i.e., it is private.
- The reference to (entity) I11 is resolved correctly. INSIDE1 is assumed and I13 was marked as exported.

A.7.4 Include Directive

Description:

A schema can be made up of several separate source files. These files can be composed into a single logical source file by the use of the Include directive. Whenever the Include directive is encountered, the file specified is processed just as if it were physically present in the file in which the Include appears. Includes can be nested to the depth permitted by the operating system environment. The form of the file name specification is also dependent on the operating system.

Syntax

216 INCLUDE-DIRECTIVE = INCLUDE STRING-LITERAL SEMICOLON.

Example:

```
SCHEMA abc;
  INCLUDE 'part1.dat';
  INCLUDE 'part2.dat';
END_SCHEMA;
```

The schema named abc is composed of three separate files: the schema itself and the files named part1.dat and part2.dat.

Restrictions:
• Circular includes are prohibited. If file A includes file B, then file B must not include file A either directly or indirectly.

• The file name is operating system dependent.

A.7.5 Uses Directive

Description:

The Uses directive creates an exact logical copy of the objects listed. In effect, those objects become part of the current schema. The main use of the Uses directive is to define application subset schemas.

Syntax

265 USES-DIRECTIVE = USES NESTED-ID-LIST SEMICOLON .

Restrictions:

• A schema cannot appear as an identifier in a Uses directive.

Example:

An application subset schema defines several entities unique to it, and also brings in several other entities declared elsewhere. The Uses directive is required since none of the declarations within the application subset reference those entities either directly or indirectly.

SCHEMA MAIN;
   -- A THRU G ARE DECLARED HERE.
SCHEMA APPLICATION_SUBSET;
   USES (A, B, C, D, E, F, G);
   ENTITY H;
   END_ENTITY;
   ENTITY J;
   END_ENTITY;
   ...
   END_SCHEMA;
END_SCHEMA;

The schema named application_subset declares the entities H and J. It also needs to treat entities A through G as if they were part of the schema. When an implementation of the application_subset is developed, it needs only to account for the entities defined in the application_subset schema, and entities defined elsewhere but referenced from it. The Uses directive creates a logical declaration of entities of interest.
A.8 Types

Types are used to associate a data representation with an attribute, local variable, or formal parameter.

Syntax

\[
\text{type} = \text{base-type} | \text{aggregation-type} | \text{entity-type} | \text{defined-type} | \text{number-type} | \text{enumeration-type} | \text{select-type} | \text{generic-type}
\]

A.8.1 Base Types

Description

The base types represent the data values that can be held in an information system of some kind. Tracing a path through the schema will eventually end with one of these base types. The base types are Real Integer String and Logical.

Syntax

186 BASE-TYPE = INTEGER-TYPE | LOGICAL-TYPE | NUMBER-TYPE | REAL-TYPE | STRING-TYPE .
220 INTEGER-TYPE = INTEGER [ PRECISION-SPEC ] .
233 PRECISION-SPEC = LEFT-PAREN EXPRESSION RIGHT-PAREN .
227 LOGICAL-TYPE = LOGICAL .
231 NUMBER-TYPE = NUMBER .
237 REAL-TYPE = REAL [ PRECISION-SPEC ] .
252 STRING-TYPE = STRING [ PRECISION-SPEC ] [ VARYING ] .

A.8.1.1 Integer Type

Description

An Integer type represents a whole number. You can optionally specify the maximum number of decimal digits. If the number of decimal digits is given, there is an implicit constraint on the value. For example, the specification "INTEGER(3)" would constrain a value to -999...999 inclusive. The value of the integer number is unconstrained if no precision specification is given.
The number of decimal digits specified suggests the storage allocation needed to hold a value of that type; e.g., if a signed binary two's complement internal representation is used, the formula \( b = \frac{\text{precision}}{0.30103} \) yields the approximate number of bits needed.

Syntax

```plaintext
220 INTEGER-TYPE = INTEGER [ PRECISION-SPEC ] .
233 PRECISION-SPEC = LEFT-PAREN EXPRESSION RIGHT-PAREN .
```

Rules and Restrictions:

- `expression` must yield a numeric result that is non-zero and non-negative.
- A real result is truncated to an integer.

Example:

This example associates an integer type with the attribute named AGE_IN_YEARS. The value of the integer is totally unconstrained.

```plaintext
ENTITY PERSON;
  AGE_IN_YEARS : INTEGER;
...
END_ENTITY;
```

Operations on Integer Types:

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unary</td>
<td>+</td>
<td>no effect</td>
</tr>
<tr>
<td>unary</td>
<td>-</td>
<td>negate the operand</td>
</tr>
<tr>
<td>binary</td>
<td>+</td>
<td>forms the sum of the operands</td>
</tr>
<tr>
<td>binary</td>
<td>-</td>
<td>forms the difference of the operands</td>
</tr>
<tr>
<td>binary</td>
<td>*</td>
<td>forms the product of the operands</td>
</tr>
<tr>
<td>binary</td>
<td>/</td>
<td>converts operands to real and forms real quotient</td>
</tr>
<tr>
<td>binary</td>
<td>div</td>
<td>forms integer quotient</td>
</tr>
<tr>
<td>binary</td>
<td>mod</td>
<td>forms integer modulus</td>
</tr>
<tr>
<td>binary</td>
<td>**</td>
<td>raises op1 to the power op2</td>
</tr>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;</td>
<td>compares for less than</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;=</td>
<td>compares for less than or equal</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;</td>
<td>compares for greater than</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;=</td>
<td>compares for greater than or equal</td>
</tr>
</tbody>
</table>
A.8.1.2 Real Type

Description

A Real type represents a normalized fractional number. A real is sometimes called a float, a floating point number, or scientific notation. It has an inexact resolution. The number of significant decimal digits can optionally be specified. The precision of the real number is unconstrained if no precision specification is given.

The number of decimal digits specified suggests the storage allocation needed to hold a value of that type; e.g., if a signed binary two's complement internal representation is used, the formula \( b = \frac{\text{precision}}{0.50103} \) yields the approximate number of bits needed for the mantissa. It also suggests the maximum number of digits that are transmitted in the physical file.

Syntax

```plaintext
237 REAL-TYPE = REAL [ PRECISION-SPEC ] .
233 PRECISION-SPEC = LEFT-PAREN EXPRESSION RIGHT-PAREN .
```

Rules and Restrictions:

- `expression` must yield a numeric result that is non-zero and non-negative.
- A real result is truncated to an integer.

Example:

This example shows the effect of defining the number of decimal digits in the fraction part of a real number; i.e., it's precision.

```plaintext
LOCAL
    DISTANCE : REAL(6);
    X1, Y1, Z1 : REAL;
    X2, Y2, Z2 : REAL;
END_LOCAL:
...
X1 := 0; Y1 := 0; Z1 := 0;
X2 := 10; Y2 := 11; Z2 := 12;
...
DISTANCE := SQRT((X2-X1)**2 + (Y2-Y1)**2 + (Z2-Z1)**2);
```

DISTANCE is computed to a value of 1.9104973...e+1 but has an actual value of 1.91050e+1 since the specification calls for six digits of precision; thus, only six normalized digits of precision are retained.

Rounding is performed as follows: given \( p \) which is the precision, the digit at \( p+1 \) is examined. If that digit is five or more, one is added to the digit \( p \).
When two real numbers are compared, the one with the smaller precision is zero extended to match the longer specification. For example, given:

\[ A : \text{REAL}(3) := 1.23E0; \]
\[ B : \text{REAL}(5) := 1.2300E0; \]

the expression \((\text{tt } a=b))\) will be true.

**Operations on Real Types:**

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unary</td>
<td>+</td>
<td>no effect</td>
</tr>
<tr>
<td>unary</td>
<td>-</td>
<td>negates the operand</td>
</tr>
<tr>
<td>binary</td>
<td>+</td>
<td>forms the sum of the operands</td>
</tr>
<tr>
<td>binary</td>
<td>-</td>
<td>forms the difference of the operands</td>
</tr>
<tr>
<td>binary</td>
<td>*</td>
<td>forms the product of the operands</td>
</tr>
<tr>
<td>binary</td>
<td>/</td>
<td>forms real quotient</td>
</tr>
<tr>
<td>binary</td>
<td>div</td>
<td>converts operands to integer and forms integer quotient</td>
</tr>
<tr>
<td>binary</td>
<td>mod</td>
<td>converts operands to integer and forms integer modulus</td>
</tr>
<tr>
<td>binary</td>
<td>**</td>
<td>raises op1 to the power op2</td>
</tr>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;</td>
<td>compares for less than</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;=</td>
<td>compares for less than or equal</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;</td>
<td>compares for greater than</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;=</td>
<td>compares for greater than or equal</td>
</tr>
</tbody>
</table>

**Normalization**

A non-zero real number is normalized in the following manner. Reading from left to right, find the first significant (i.e., non-zero) digit. Then adjust the decimal point so that it lies just to the right of the significant digit. Count the number of places the decimal point moved and the direction it moved. If it moved to the right, subtract one from the exponent for each position; if it moved to the left, add to the exponent.

**Example:**

\[
\begin{align*}
0.0 & \quad 0.0 \\
0.0000001e11 & \quad 1.0e5 \\
0001780000.e0 & \quad 1.78e+7
\end{align*}
\]

**A.8.1.3 Logical Type**

**Description**
A Logical type represents a True/False value. Unknown is a third logic state which can be used when the state of a logical value is neither True nor False. The following conditions hold for logical types:

\[ \text{False} < \text{Unknown} < \text{True} \]

Syntax

```
227 LOGICAL-TYPE = LOGICAL .
```

Example:

This example associates a logical type with an attribute named FLAG. FLAG can only hold the values True and False.

```
ENTITY SOMETHING;
  FLAG : LOGICAL;
...;
END_ENTITY;
```

Operations on Logical Types:

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unary</td>
<td>not</td>
<td>reverses the state of the operand</td>
</tr>
<tr>
<td>binary</td>
<td>and</td>
<td>returns true if both operands are true</td>
</tr>
<tr>
<td>binary</td>
<td>or</td>
<td>returns true if either operand is true</td>
</tr>
<tr>
<td>binary</td>
<td>xor</td>
<td>returns true if one of the operand is true</td>
</tr>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;</td>
<td>compares for less than</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;=</td>
<td>compares for less than or equal</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;</td>
<td>compares for greater than</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;=</td>
<td>compares for greater than or equal</td>
</tr>
</tbody>
</table>

Truth Tables

<table>
<thead>
<tr>
<th>AND</th>
<th>OR</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>?</td>
<td>F</td>
</tr>
<tr>
<td>+-----+-----+-----+-----+-----+-----+-----+-----+-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>?</td>
</tr>
</tbody>
</table>
A.8.1.4 String Type

Description

A String type represents a sequence of zero or more characters. The character set and collating sequence is defined by ISO 6937/2.

Characters and symbols not defined by that standard can be represented as an "escape" sequence. The escape sequence is defined in normative Annex B, the Physical File Structure Language. An example of an escape sequence is given below.

The maximum number of characters in a string can be specified. If no such specification is given, the maximum length of the string is unlimited.

The varying property indicates that the string is allowed to grow and shrink in apparent size. The absence of the varying property indicates that the string is always exactly the specified size.

Syntax

252 STRING-TYPE = STRING [ PRECISION-SPEC ] [ VARYING ] .
233 PRECISION-SPEC = LEFT-PAREN EXPRESSION RIGHT-PAREN .

Rules and Restrictions:

- expression must yield a numeric result that is non-zero and non-negative.
- A real result is truncated to an integer.

Example:

The following four examples show the effect of the different specification elements of a string.

STRING1 : STRING;

This defines an abstract string. It cannot be implemented without making assumptions about its maximum length.

STRING2 : STRING(10);

This defines a string that is always ten characters in length; i.e., all ten positions are assumed to hold a character of some kind.
STRING3 : STRING VARYING;

This defines an abstract string. It cannot be implemented without making assumptions about its maximum length.

STRING4 : STRING(10) VARYING:

This defines a string that can hold zero to ten characters.

Escape Sequence Example:

letters := '!!greek:01, 02, 03!ABC!math5:01, 02, 03!';

The string in this example has nine characters and uses characters from three character sets. The first three come from the Greek character set, the next three come from the (default) ISO 6937/2 set, and the last three come from a math character set. The numbers (01, 02, etc.) refer to a specific character in the designated character set.

Operations on String Types:

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;</td>
<td>compares for less than</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;=</td>
<td>compares for less than or equal</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;</td>
<td>compares for greater than</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;=</td>
<td>compares for greater than or equal</td>
</tr>
<tr>
<td>binary</td>
<td>like</td>
<td>returns true if the left operand matches the template, which is the right operand</td>
</tr>
</tbody>
</table>

Components of a string variable can be addressed by a subscript. For example, the seventh character of a string called "name" could be examined by

IF NAME[7] = 'S' THEN

... String comparison

To compare two strings, compare the first (leftmost) pair of characters, then the pair at the second position, etc. until an unequal pair is found, or until all character pairs have been examined. If an unequal pair is found, the string containing the lesser character (as defined by the ISO 6937/2 collating sequence) is considered less than the other string. When all pairs are equal, the two strings are equal.
When two strings are not the same length, the shorter string is blank extended until the lengths of the two strings are the same.

Implementations that do not use the ISO 6937/2 collating sequence must yield a string comparison result that is the same as an implementation that does.

A.8.2 Aggregation Types

Description

Aggregation types are used to represent ordered or unordered collections of like things. These collections can have fixed or varying sizes. Aggregation types are called Aggregate, Array List and Set.

Syntax

222 LIMIT-SPEC = LEFT-BRACKET EXPRESSION COLON ( EXPRESSION | INFINITY ) RIGHT-BRACKET .
182 ARRAY-TYPE = ARRAY INDEX-SPEC OF [ OPTIONAL ] [ UNIQUE ] ATTRIBUTE-CAN-BE .
218 INDEX-SPEC = LEFT-BRACKET EXPRESSION COLON EXPRESSION RIGHT-BRACKET .

A.8.2.1 Aggregate Type

Description

An Aggregate type is a generalization of any aggregation type; i.e., an Array, List, or Set. An aggregate type can be used only as a formal parameter of a function or procedure, or as the result of a function.

When a procedure or function is invoked, an actual parameter is passed. That actual parameter must be either an Array, List, or Set. The operations that can be performed then depend on the type of the actual parameter.

Syntax

222 LIMIT-SPEC = LEFT-BRACKET EXPRESSION COLON ( EXPRESSION | INFINITY ) RIGHT-BRACKET .

Rules and Restrictions:

- Both expressions must yield a number result.
• A real result is truncated to an integer.
• Expression1 is the lower bound of the array.
• Expression2 is the upper bound of the array.
• The lower bound must be less than or equal to the upper bound.
• The Optional specification indicates that the array instance can be sparse; i.e., all of the elements are not always populated.
• The Unique specification indicates that each populated element of an array must be different than all other populated elements.

Example:

This example shows how a multi-dimensioned array is declared.


The first array has 10 elements of type ARRAY[11:14] OF e. There are 40 elements total in the attribute named sectors. There is no restriction on the number of dimensions of an array.

Operations on Array Types:

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
</tbody>
</table>

A.8.2.2 List Type

Description

A List type represents an ordered collection of like elements. The number of elements of a list can vary from one occurrence to another. A list specifies the minimum number of elements that must be present and the maximum number of elements that can be present. The minimum bound of a list must never be less than zero and the maximum bound can be either exact or indefinite. The number of elements in a list can vary between the minimum and maximum bounds (inclusive).

The SizeOf built-in function can be used to determine the number of elements in a list.

Syntax

222 LIMIT-SPEC = LEFT-BRACKET EXPRESSION COLON ( EXPRESSION | INFINITY ) RIGHT-BRACKET .
Rules and Restrictions:

1. Both expressions must yield a number result.
2. A real result is truncated to an integer.
3. Expression1 is the minimum number of elements that must be in a list.
4. Expression2 is the maximum number of elements that can be in a list. The alternate # specification states that the size of the list is unconstrained.
5. The lower bound must be less than or equal to the upper bound and it must be non-negative.
6. The Unique specification indicates that each element of a list must be different than all other elements. The effect of using unique in the declaration of a list is to define an ordered set.

Example:

This example defines a list of arrays. The list can contain zero to ten arrays. Each array must be different from all other arrays in the list.

ComplexList : LIST[0:10] OF UNIQUE ARRAY[1:10] OF INTEGER;

There are zero, 10, 20, ..., 100 integer values associated with the attribute named ComplexList depending on the number of list elements. Each array must be different than every other array as specified by the Unique qualifier.

Operations on List Types:

```
+-----------------+-------+-------------------------------------+
<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>in</td>
<td>returns true if the first operand (an element of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the list) is in the list</td>
</tr>
<tr>
<td>binary</td>
<td>+</td>
<td>L+E adds E to end of list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E+L adds E to front of list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L+L combines the two lists (concatenation)</td>
</tr>
</tbody>
</table>
+-----------------+-------+-------------------------------------+
```

L represents a list; E represents an element of that list.

The INSERT built-in procedure is used to insert elements into a list.
A.8.2.3 Set Type

Description

A Set type represents an unordered collection of like elements. The number of elements in a set can be constrained, but that is not necessary; it can be empty or contain any number of elements. No two elements of a set can have the same value however.

Syntax

222 LIMIT-SPEC = LEFT-BRACKET EXPRESSION COLON ( EXPRESSION | INFINITY ) RIGHT-BRACKET .

Rules and Restrictions:

- Both expressions must yield a number result.
- A real result is truncated to an integer.
- Expression1 is the minimum number of elements that must be in a set.
- Expression2 is the maximum number of elements that can be in a set. The alternate # specification states that the size of the set is unconstrained.
- The lower bound must be less than or equal to the upper bound and it must be non-negative.

Example:

This example defines an attribute as a set of points (an entity type).

ABunchOfPoints : Set Of Point;

The attribute named ABunchOfPoints can contain zero or more points. Each point instance (in the set instance) must be different than every other point in the set.

If a set requires at least one (or more) element, the specification can provide a lower bound.

ABunchOfPoints : Set [1:#] Of Point;

The attribute named ABunchOfPoints now must contain at least one point. Each point instance (in the set instance) must be different than every other point in the set.

Operations on Set Types:

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
</table>

31
### Binary Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td>compares for equality</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>compares for inequality</td>
</tr>
<tr>
<td><code>in</code></td>
<td>returns true if the first operand (an element of the set) is in the set (E in S)</td>
</tr>
<tr>
<td><code>*</code></td>
<td>S + S is union</td>
</tr>
<tr>
<td><code>-</code></td>
<td>S - S is difference</td>
</tr>
<tr>
<td><code>+</code></td>
<td>S + E adds E to S</td>
</tr>
<tr>
<td><code>-</code></td>
<td>S - E removes E from S</td>
</tr>
<tr>
<td><code>*</code></td>
<td>S * S is intersection</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>returns true when S1 is a subset of S2</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>returns true when S2 is a subset of S1</td>
</tr>
</tbody>
</table>

*S represents a set; E represents an element of that set.*

### A.8.3 Entity Type

#### Description

Any Entity declared in a schema can be used as the type of an attribute, local variable, or parameter.

#### Syntax

```plaintext
200 ENTITY-TYPE = ENTITY-ID .
```

#### Rules and Restrictions:

1. An entity type cannot be used in a defined type.

#### Example:

This example uses a Point entity as the type of an attribute.

```plaintext
ENTITY POINT;
    I, Y, Z : REAL;
END_ENTITY;

ENTITY LINE;
    PO, P1 : POINT;
END_ENTITY;
```

The line entity has two attributes named PO and P1. The type of each of those attributes is [entity] Point. In this case the value of each Point is mandatory. Thus, a line cannot exist without those two points.

#### Relationships in Express
Every attribute establishes a relationship between an entity and some object. That object can
be a terminal or a non-terminal object. A terminal object is anything such as an integer number
that we think of as irreducible. The base types of Express are terminal objects. An entity is a
non-terminal object. The nature of the relationship does not depend on whether the object is
terminal or non-terminal.

Given an entity E an attribute A and the type of the attribute T a relationship can be dia-
grammed as:

\[ E . A \rightarrow [m:n] \rightarrow [p:q] \rightarrow T \]

The relationship can be explained in English as: "T plays the role of A in the definition of E ". For example, for \( E = \text{Line} \ A = \text{Start} \) and \( T = \text{Point} \) the interpretation is: "Point plays the role of Start in the definition of Line ".

\( m \) and \( n \) define the number of \( T \)'s needed for each \( E.A \); \( p \) and \( q \) define the number of \( E \)'s needed for each \( T \).

The value of \( m \) and \( n \) is specified by array, list, or set or the absence of one of those aggregates.

A specification such as

\[ A : B ; \]

means that \( m=n=1 \) while the specification

\[ A : \text{ARRAY}[1:90] \text{OF} B ; \]

translates to \( m=1:n=90 \).

The value of \( p \) and \( q \) depends on whether the relationship is internal or external. An internal
relationship always yields \( p=q=1 \); i.e., for every \( T \) there must be exactly one \( E \). An external
relationship always yields \( p=0:q=\# \); i.e., for every \( T \) there can be zero or more \( E \)'s. Since
dynamic means either internal or external, the values of \( p \) and \( q \) depend on whether \( T \) is
actually internal or external in an entity instance.

In a system of instances (as in a database) we may find that a constraint has to be put on \( p \)
and \( q \). In other words, \( p=0:q=\# \) is unacceptable. The cardinality relational operator can be
used to state that constraint. The cardinality relational operator looks like

\[ T \{ \exists m [ p' : q' ] \} E.A \]

which means that the actual number of \( E.A \)'s for a given \( T (k) \) must satisfy

\[ \{ p' <= k <= q' \} . \]

We have not redefined \( p \) and \( q \), but have constrained \( k \) in some context of use.

Operations on Entity Types:

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>:=</td>
<td>compares for instance equality</td>
</tr>
<tr>
<td>binary</td>
<td>:&lt;&gt;</td>
<td>compares for instance inequality</td>
</tr>
</tbody>
</table>
A.8.4 Defined Type

Description

A Defined type is a user extension to the set of built-in types. Defined types must be declared in a Type block. Once declared, a defined type can be used as any other type.

Enumeration and Select types can be defined only as a defined type.

Syntax

```
194 DEFINED-TYPE = DEFINED-TYPE-ID .
```

Rules and Restrictions:

1. A defined type must be declared in a Type...End_Type block.
2. An Entity type can not be used as a defined type.

Example:

Several types are defined below to indicate the units of measure associated with certain kinds of attributes.

```
TYPE
  AREA = REAL;
  MAGNITUDE = REAL;
  VOLUME = REAL;
  MASS = REAL;
... END_TYPE;
...
ENTITY PART;
...
  WEIGHT : MASS;
END_ENTITY;
```

The WEIGHT attribute is represented as a real number, but the use of the defined type, MASS, helps to clarify the meaning and context of the real number; it means mass, not volume or area for instance.

Operations on Defined Types:

The operations applicable to defined types depends on the base type; e.g., if the base type is real, only the operation set for real types can be used. Note however that assignment of dissimilar user defined types is restricted; e.g., if VA is a variable of type A and VB is a variable of type B then the statement VA := VB; is illegal.
A.8.5 Number Type

Description

A Number is an abstraction of Real or Integer types. Generally speaking, a number type is used in an "abstract" sense when an exact representation does not matter, or as the specification of a generic number parameter of a function or procedure.

Syntax

```
231 NUMBER-TYPE = NUMBER .
```

Example:

The following example shows a generic numeric addition function that accepts any number as actual parameters and returns the appropriate kind of numeric result. An integer is returned if both inputs are integers; a real result is returned for mixed arithmetic. This function does no error checking.

```
FUNCTION ADD(A,B:NUMBER):NUMBER;
    LOCAL
        RI : INTEGER; -- INTEGER RESULT
        RR : REAL;     -- REAL RESULT
    END_LOCAL;
    CASE TYPEOF(A) OF
        INTEGER : CASE TYPEOF(B) OF
            INTEGER : BEGIN; RI := A+B; RETURN(RI); END;
            REAL   : BEGIN; RR := A+B; RETURN(RR); END;
            END_CASE;
        REAL    : CASE TYPEOF(B) OF
            INTEGER : BEGIN; RR := A+B; RETURN(RR); END;
            REAL   : BEGIN; RR := A+B; RETURN(RR); END;
            END_CASE;
    END_CASE;
END_FUNCTION;
```

Operations on Number Types:

```
+-----------------+-----------------+-----------------|
| form | op | description |
+-----------------+-----------------+-----------------|
| unary | + | no effect |
| unary | - | negates the operand |
| binary | + | forms the sum of the operands |
| binary | - | forms the difference of the operands |
| binary | * | forms the product of the operands |
```
A.8.6 Enumeration Type

Description

An Enumeration is an ordered set of values represented by names. Note that the enumeration type can only be used as a defined type.

Each enumeration item exists at the same scope as the type of the enumeration itself. Thus, given the example below, the identifiers colors, red, blue, and green all exist at the same lexical scope.

Syntax

201 ENUMERATION-TYPE = ENUMERATION OF NESTED-ID-LIST .

Rules and Restrictions:

1. Each identifier declared as an enumerated item becomes a unique identifier in the current scope.

2. Each enumerated item has a value that is dependent on its position in the list. The value of the first item is less than the second; the second is less than the third, etc. The specific value is not relevant.

Example:

This example shows an enumeration of colors.

COLORS = ENUMERATION OF (RED, BLUE, GREEN);

(* RED < BLUE, BLUE < GREEN *)
Operations on Enumeration Types

<table>
<thead>
<tr>
<th>form</th>
<th>op</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>=</td>
<td>compares for equality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;&gt;</td>
<td>compares for inequality</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;</td>
<td>compares for less than</td>
</tr>
<tr>
<td>binary</td>
<td>&lt;=</td>
<td>compares for less than or equal</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;</td>
<td>compares for greater than</td>
</tr>
<tr>
<td>binary</td>
<td>&gt;=</td>
<td>compares for greater than or equal</td>
</tr>
</tbody>
</table>

A.8.7 Select

Type

Description

A Select type is used when a choice must be made among a number of possible entity types.

Syntax

245 SELECT-TYPE = SELECT NESTED-ID-LIST .

Rules and Restrictions

1. Each identifier in the selection list must be an entity identifier or a select type.

Example

A choice must be made among several dissimilar things in a given context.

TYPE

    ArbitraryChoice = SELECT(NAIL, SCREW, GLUE);
END_TYPE

    AttachmentMethod : ArbitraryChoice;

An AttachmentMethod can be any of the entities named NAIL, SCREW, or GLUE.

Operations on Select Types

No operations are defined for the select type.

A.8.8 Generic Type

Description
A Generic type is used to represent any type. The purpose of the generic type is to enable the writing of generic functions and procedures.

Syntax

213 GENERIC-TYPE = GENERIC \}

Rules and Restrictions

1. A generic type can only be used as a parameter type in a function or procedure formal parameter list or as the result type of a function.

Example:

This example shows a generic function that adds numbers or vectors. It is an elaboration of the example given above in 7.1.1.

FUNCTION ADD(A,B:GENERIC):GENERIC;
  LOCAL
   RI : INTEGER; -- INTEGER RESULT
   RR : REAL; -- REAL RESULT
   RV : VECTOR; -- VECTOR RESULT
  END_LOCAL;
  CASE TYPEOF(A) OF
   INTEGER : CASE TYPEOF(B) OF
      INTEGER : BEGIN; RI := A+B; RETURN(RI); END;
      REAL : BEGIN; RR := A+B; RETURN(RR); END;
   END_CASE;
   REAL : CASE TYPEOF(B) OF
      INTEGER : BEGIN; RR := A+B; RETURN(RR); END;
      REAL : BEGIN; RR := A+B; RETURN(RR); END;
   END_CASE;
   VECTOR : IF TYPEOF(B)=VECTOR THEN
      BV.I := A.I+B.I;
      BV.J := A.J+B.J;
      BV.K := A.K+B.K;
      RETURN(RV);
   END_IF;
  END_CASE;
  END_FUNCTION;

Operations on Generic Types:

No operations are defined for generic types. You must first determine the actual type (by the TipoOf built-in function) and then use the appropriate operation set for that type.
A.9 Built-in Constants

The built-in constants are assumed to have exact values, even though an exact value may not exist in the real world; e.g., pi is assumed to be exact.

A.9.1 Infinity

The infinity symbol (⊗) stands for an indeterminate upper bound of a LIST or SET.

A.9.2 False

False is the opposite of True; see below.

A.9.3 Pi

Pi is the number 3.14159.... It is assumed that this number is exact.

A.9.4 True

True False and Unknown together make up a ternary logic system in which False is NOT True and Unknown is neither True nor False.

A.9.5 Unknown

Unknown is a logic state which indicates that the value is neither True nor False.

A.9.6 Built-in Functions

All functions (and mathematical operations in general) are assumed to yield exact results. Functions that operate on angles express angular measurement in terms of radians.

A.9.7 Abs - Arithmetic Function

FUNCTION ABS ( V:NUMBER ) : NUMBER;

The Abs function finds the absolute value of V and returns the result (integer or real depending on the type of the input) to the point of invocation.

Example:

\[ \text{ABS} (-10) \rightarrow 10 \]

A.9.8 ACos - Arithmetic Function

FUNCTION ACOS ( V:NUMBER ) : REAL;

The ACos function finds the arc-cosine of V and returns a real in radians to the point of invocation.

Example:
A.9.9 ASin - Arithmetic Function

FUNCTION ASIN ( V:NUMBER ) : REAL;

The ASin function finds the arc-sine of V and returns a real result in radians to the point of invocation.

Example:

\[ \text{ASIN} (0.3) \rightarrow 1.266103\ldots \]

A.9.10 ATan - Arithmetic Function

FUNCTION ATAN ( V1:NUMBER ; V2:NUMBER ) : REAL;

The ATan function finds the arctangent of V and returns a real result in radians to the point of invocation.

\[ \text{ATAN}(v1, v2) \]

The argument v1 must always be specified; the argument v2 is optional. If v2 is omitted, v1 represents the value whose arctangent is to be found. If v2 is specified, the value whose arctangent is to be found is taken to be the expression v1/v2 and in the case where v2 is zero the result is \(\pi/2\) or \(-\pi/2\) depending on the sign of v1.

Example:

\[ \text{ATAN}(0) \rightarrow 0.0 \]
\[ \text{ATAN}(x, y) \]
\[ \text{ATAN}(-5.5, 3.0) \rightarrow -1.071449\ldots \]

A.9.11 Cos - Arithmetic Function

FUNCTION COS ( V:NUMBER ) : REAL;

The Cos function finds the cosine of V expressed in radians and returns a real result to the point of invocation.

Example:

\[ \text{COS}(0.5) \rightarrow 0.77582\ldots \times 10^{-1} \]

A.9.12 Exists - General Function

FUNCTION EXISTS ( V ) : LOGICAL;

The Exists function accepts any variable and returns True if a value exists for that parameter, or False when no value exists for it. Note that only attributes declared with the Optional keyword can ever have a missing value.

Example:
if EXIST (a) then

A.9.13 Exp - Arithmetic Function

FUNCTION EXP (V:NUMBER): REAL;

The EXP function raises e (the base of the natural logarithm system) to the power V and returns a real to the point of invocation.

Example:

EXP (10) --> 2.202646. E+4

A.9.14 HiBound - Arithmetic Function

FUNCTION HIBOUND (V): INTEGER;

The HiBound function returns the actual upper bound of an Array, List, or Set. The result depends on the type of the actual parameter V. If the actual parameter is:

<table>
<thead>
<tr>
<th>Actual Parameter Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>the declared upper bound of the array</td>
</tr>
<tr>
<td>List</td>
<td>the number of elements actually in the list.</td>
</tr>
<tr>
<td>Set</td>
<td>the number of elements actually in the set.</td>
</tr>
<tr>
<td>any non-aggregate</td>
<td>1 (one).</td>
</tr>
</tbody>
</table>

Example:

LOCAL
A: ARRAY[-3:19] OF SET OF LIST[0:#] OF INTEGER;
H1, H2, H3: INTEGER;
END_LOCAL;

A[-3][1][1] := 2;
H1 := HIBOUND(A); -- =19 (UPPER BOUND OF ARRAY)
H2 := HIBOUND(A[-3]); -- =1 (SIZE OF SET)
H3 := HIBOUND(A[-3][1]); -- =1 (SIZE OF LIST)

A.9.15 LoBound - Arithmetic Function

FUNCTION LOBOUND (V): INTEGER;

The LoBound function returns the actual lower bound of an Array, List, or Set. The result depends on the type of the actual parameter V. If the actual parameter is:
<table>
<thead>
<tr>
<th>Actual Parameter Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>the declared lower bound of the array</td>
</tr>
<tr>
<td>List</td>
<td>0 (zero) if the list is empty; otherwise 1 (one).</td>
</tr>
<tr>
<td>Set</td>
<td>0 (zero) if the set is empty; otherwise 1 (one).</td>
</tr>
<tr>
<td>any non-aggregate</td>
<td>1 (one).</td>
</tr>
</tbody>
</table>

Example:

```pascal
LOCAL
  H1, H2, H3 : INTEGER;
END_LOCAL;

  H1 := LOBOUND(A);       -- =-3 (LOWER BOUND OF ARRAY)
  H2 := LOBOUND(A[-3]);   -- =0 (SIZE OF SET)
  H3 := LOBOUND(A[-3][1]); -- =0 (SIZE OF LIST)
```

A.9.16 Log - Arithmetic Function

```pascal
FUNCTION LOG ( V:NUMBER ) : REAL;
```

The Log function finds the natural logarithm of \( V \) and returns a real result to the point of invocation.

Example:

```pascal
LOG ( 4.5 ) --> 1.504077...EO
```

A.9.17 Log2 - Arithmetic Function

```pascal
FUNCTION LOG2 ( V:NUMBER ) : REAL;
```

The Log2 function finds the base 2 logarithm of \( V \) and returns a real result to the point of invocation.

Example:

```pascal
Log2 ( 8 ) --> 3
```

A.9.18 Log10 - Arithmetic Function

```pascal
FUNCTION LOG10 ( V:NUMBER ) : REAL;
```

The Log10 function finds the base ten logarithm of \( V \) and returns a real to the point of invocation.

Example:

```pascal
LOG10 ( 10 ) --> 1.00...EO
```
A.9.19 Odd - Arithmetic Function

FUNCTION ODD ( V:INTEGER ) : LOGICAL;

The Odd function accepts any integer value V and returns True if the number is odd; i.e., -3, -1, 1, 3, 5 ...
Example:

ODD ( 121 ) --> TRUE

A.9.20 Sin - Arithmetic Function

FUNCTION SIN ( V:NUMBER ) : REAL;

The Sin function finds the sine of V expressed in radians and returns a real result to the point of invocation.
Example:

SIN ( pi ) --> 0.0

A.9.21 SizeOf - Aggregate Function

FUNCTION SIZEOF ( V:AGGREGATE ) : INTEGER;

The SizeOf function accepts any aggregate type and returns the number of elements in it.

The returned value for an Array will be its declared number of elements.

The returned value for a List will be the number of occupied elements.

The returned value for a Set will be the number of actual elements.
Example:

LOCAL
N : NUMBER;
X : LIST[1:#] OF Y;
...
N = SIZEOF ( X ) ;

A.9.22 Sqrt - Arithmetic Function

FUNCTION SQRT ( V:NUMBER ) : REAL;

The Sqrt function finds the square root of the non-negative numeric value V and returns a real result to the point of invocation.
Example:

SQRT ( 121 ) --> 11
A.9.23  Tan - Arithmetic Function

FUNCTION TAN ( V:NUMBER ) : REAL;

The Tan function finds the tangent of V and returns a real to the point of invocation.
Example:

TAN ( 0.0 ) --> 0.0

A.9.24  TypeOf - General Function

FUNCTION TYPEOF ( V ) : SET OF TYPE;

The TypeOf function accepts any variable V and returns a set containing its type. When V is an entity subtype the set contains the type of the entity itself and each of its supertype entity types, etc. until there are no more supertype entity types.
Example:

if TYPEOF ( a ) = point THEN

...

A.9.25  Value - Arithmetic Function

FUNCTION VALUE ( V:STRING ) : NUMBER;

The Value function converts a string variable or string literal to its numeric equivalent and returns the result to the point of invocation. The type of the result matches the type of the input string. If the string cannot be converted to a number, the returned value is zero.
Example:

VALUE ( '1.234' ) --> 1.234 (REAL)
VALUE ( '20' ) --> 20 (INTEGER)

A.10  Built-in Procedures

A.10.1  Insert - List Procedure

PROCEDURE INSERT ( VAR L:LIST; E; P:NUMBER ) ;

The Insert procedure is used to insert an element compatible with a list variable into that list variable. L is a list, E is an element compatible with the type of L and P is a number indicating where E is inserted into the list. The value of P must satisfy:

{ 0 <= P <= SizeOf(L) }

E is inserted after the list element indicated by P. For example, when P=0 E becomes the first element in the list.
A.10.2 Nullify - General Procedure

PROCEDURE NULLIFY ( VAR V );

The Nullify procedure accepts a variable of any type and nullifies its value. Nullification means causing a value associated with an attribute to cease to exist. That is, the attribute no longer has a value after nullification. Attempts to nullify an attribute not declared as optional has no effect on its value.

A.10.3 Violation - General Procedure

PROCEDURE VIOLATION ;

The Violation procedure raises an exception. This procedure can be invoked inside a rule to indicate that some global constraint has been violated. Other than having the ability to indicate that an exceptional condition exists, Express provides no mechanism for recovering from it.
A.11 Statements and Blocks

A statement is a sequence of tokens that conform to the grammar of Express. Most statements are terminated by the semicolon character. There are a few exceptions to this rule. For example, the TYPE statement which starts a type definition block is not terminated with a semicolon.

Certain sequences of statements are called blocks. All of the statements that make up a block are treated logically as a single statement. There are two kinds of blocks: declaration blocks and execution blocks. The declaration blocks are the Schema Entity Map Type and Local blocks. Functions, procedures, and rules are called execution blocks.

The outermost block must be a Schema block. A schema contains all declarations including possibly other schemas.

Some blocks establish a scope over which identifiers must be unique. Every identifier declaration within those blocks must be unique. The type block is a pseudo-block; it does not spawn a new identifier scope. For example, the names of the attributes of an entity are unique to the entity block in which they are declared. The same attribute identifier could be used in another entity since each entity establishes its own scope.

Examples

```express
SCHEMA example; -----------------------------

ENTITY entity1; ------
a : integer;  | this is a block
b : integer;  | inside a block
END_ENTITY;  ------

ENTITY entity2; ------
a : entity1;  | this is a block
b : integer;  | inside a block
END_ENTITY;  ------

END_SCHEMA; -----------------------------
```

A.11.1 Schema

Description

A Schema contains all of the declarations needed to describe some problem area. A schema can declare types, rules, entities, functions, procedures, and even other schemas. It can also contain directives and remarks as needed.

In general, a declaration does not have to precede references to it. Position is important for two directives. A Define directive must appear before a reference to the replacement symbol it declares. Likewise, an Assume directive must be given prior to any references that would be resolved in one of the assumed schemas.
Syntax

244 SCHEMA-DECL = SCHEMA SCHEMA-ID SEMICOLON { BLOCK-MEMBER }
                END-SCHEMA SEMICOLON .
187 BLOCK-MEMBER = ENTITY-DECL | MAP-DECL | FUNCTION-DECL | PROCEDURE-DECL
                    | RULE-DECL | SCHEMA-DECL | TYPE-DECL | DIRECTIVE .
197 DIRECTIVE = ASSUME-DIRECTIVE | DEFINE-DIRECTIVE | EXPORT-DIRECTIVE
                | INCLUDE-DIRECTIVE | USES-DIRECTIVE .
184 ASSUME-DIRECTIVE = ASSUME LEFT-PAREN SCHEMA-ID { COMMA SCHEMA-ID }
                    RIGHT-PAREN SEMICOLON .
193 DEFINE-DIRECTIVE = DEFINE DEFINED-TYPE-ID STRING-LITERAL SEMICOLON .
205 EXPORT-DIRECTIVE = EXPORT ( NESTED-ID-LIST | EVERYTHING
                    | EVERYTHING EXCEPT NESTED-ID-LIST ) SEMICOLON .
216 INCLUDE-DIRECTIVE = INCLUDE STRING-LITERAL SEMICOLON .
265 USES-DIRECTIVE = USES NESTED-ID-LIST SEMICOLON .

A.11.2 Type Declarations

Description

Any user defined types must be declared in a Type block. Each defined type is a unique type
that is not assignment compatible with other built-in or user defined types. This is true even
if the underlying types are the same.

Syntax

261 TYPE-DECL = TYPE TYPE-ITEM { TYPE-ITEM } END-TYPE SEMICOLON .

Examples

    TYPE
    MASS = REAL ;
    VOLUME = REAL ;
    END_TYPE ;

Whenever MASS or VOLUME is used as a data type, a REAL is used to store the value of a
variable. MASS and VOLUME are incompatible types however. That is, it is illegal to assign
a variable of type MASS to a variable of type VOLUME.

A.11.3 Entity

Description

A Entity declaration creates an entity type. The entity type represents some thing, concept,
event, etc. that is important to a schema. An entity instance is an occurrence of an entity type
as in a database. The word entity used alone means entity type. Whenever an entity occurrence is intended the words entity instance will be used.

An entity is defined in terms of a set of attributes. Explicit attributes represent the data one would expect to find in a database. Derived attributes represent information that can be computed as needed from other attributes. Derived attributes are declared following the Derive keyword.

Uniqueness constraints are specified following the Unique keyword. Uniqueness can be specified for individual attributes or combinations of attributes.

It is often necessary to define a number of constraints that apply to the explicit attributes, either individually or in combination with one another. These constraints are defined following the Where keyword.

The combination of explicit attributes, derived attributes, and constraints is often sufficient to fully define the intended characteristics of an entity. On occasion however, a more complex constraint is needed to fully define the behavior of an entity. See the section titled Rule for more information on this subject.

Syntax

```
199 ENTITY-DECL = ENTITY ENTITY-ID [ SUBSUPER-DECL ] SEMICOLON
       [ EXPLICIT-ATTRIBUTE-DECL ] [ DERIVED-ATTRIBUTE-DECL ]
       [ UNIQUE-RULE ] [ WHERE-RULE ] END-ENTITY SEMICOLON.
204 EXPLICIT-ATTRIBUTE-DECL = EXPLICIT-ATTRIBUTE { EXPLICIT-ATTRIBUTE }.
203 EXPLICIT-ATTRIBUTE = ID-LIST COLON [ OPTIONAL ] [ EXTERNAL | INTERNAL | DYNAMIC ] ATTRIBUTE-CAN-BE SEMICOLON.
196 DERIVED-ATTRIBUTE-DECL = DERIVE DERIVED-ATTRIBUTE { DERIVED-ATTRIBUTE }.
195 DERIVED-ATTRIBUTE = ATTRIBUTE-ID COLON RESULT-CAN-BE INITIALIZER SEMICOLON.
219 INITIALIZER = ASSIGNMENT-OPERATOR EXPRESSION.
263 UNIQUE-RULE = UNIQUE { ID-LIST SEMICOLON }.
266 WHERE-RULE = WHERE { EXPRESSION SEMICOLON }.
```

A.11.3.1 Supertypes and Subtypes

Description

An Entity SubType is a kind or variety of the entity that is its supertype. All of the subtype entities that share the same supertype have certain characteristics in common. These common characteristics may be material; i.e., the attributes of the supertype entity are implicitly part of each subtype entity. In some cases entities that share a common supertype may share no attributes, but are still constructed as subtypes because they are widely understood to be related entities.
Syntax

\[ 254 \text{SUBSUPER-DECL} = \text{SUBTYPE-DECL} [ \text{SUPERTYPE-DECL} ] | \text{SUPERTYPE-DECL} [ \text{SUBTYPE-DECL} ] \]

An Entity SuperType must mention those entities that are its subtypes. Likewise, an entity that is a subtype must mention each of its supertypes. The relationship between a supertype entity and its subtypes can be inclusion (and) or exclusion (or or xor). If a subtype entity has more than one supertype entity, the relationship is always inclusion (this means that the existence of a subtype depends on the existence of all of its supertypes).

Examples

Assume entities named A, B, C, D, E, and F. Entities B, C, and D are subtypes of entity A. Entities D and F are subtypes of entity E. If we assume exclusion in all cases, the entity headers for each entity would look like:

\[
\begin{align*}
\text{ENTITY A SUPERTYPE OF (B xor C xor D);} \\
\text{ENTITY B SUBTYPE OF (A);} \\
\text{ENTITY C SUBTYPE OF (A);} \\
\text{ENTITY D SUBTYPE OF (A, E);} \\
\text{ENTITY E SUPERTYPE OF (D xor F);} \\
\text{ENTITY F SUBTYPE OF (E);}
\end{align*}
\]

The dependency relationships that can exist for this configuration are:

<table>
<thead>
<tr>
<th>Given an instance of:</th>
<th>There must also exist:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B xor C xor D</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>A and E</td>
</tr>
<tr>
<td>E</td>
<td>D xor F</td>
</tr>
<tr>
<td>F</td>
<td>E</td>
</tr>
</tbody>
</table>

An instance of a subtype entity demands the existence of each of its supertypes.

If a supertype entity has an attribute, that attribute is implicitly known to each of its subtypes. Thus, if the entity A above had an attribute called a, the entities B, C, and D also have the very same a attribute. This means that entity E cannot also have an attribute called a since entity D would have no way of distinguishing between the two attributes.

The form of the supertype declaration can specify both inclusion and exclusion. For example, the entity header:
ENTITY X SUPERTYPE OF ((P xor Q) and R);

says that entity X is at the same time a supertype of either entities P and R, or Q and R.

You can also indicate that a supertype entity might have no subtype as in:

ENTITY X SUPERTYPE OF (A xor B xor NULL);

Meaning that entity X is a supertype of A or B or nothing at all. Note that the and operator cannot be used with NULL.

A.11.3.2 Attribute

An entity is defined in terms of a set of attributes, which are the entity’s essential traits, qualities, or properties. Attributes can be explicit data or the value can be derived.

A.11.3.2.1 Explicit Attribute

Description

An explicit attribute represents actual data that one might expect to find in a database implementation.

Syntax

204 EXPLICIT-ATTRIBUTE-DECL = EXPLICIT-ATTRIBUTE { EXPLICIT-ATTRIBUTE } .
203 EXPLICIT-ATTRIBUTE = ID-LIST COLON [ OPTIONAL ] [ EXTERNAL
| INTERNAL | DYNAMIC ] ATTRIBUTE-CAN-BE SEMICOLON .

Several attributes that are alike in all respects can be declared together as in:

ENTITY point :
   x, y, z : real ;
END_ENTITY ;

which is equivalent to:

ENTITY point :
   x : real ;
   y : real ;
   z : real ;
END_ENTITY ;
Optional indicates that an attribute does not have to have a value.

An explicit attribute can be both Optional and Unique at the same time (see "Uniqueness Rule" below). When an attribute is declared as both Optional and Unique and the value in an instance is missing, that value is exempt from the uniqueness test. If neither of these keywords are given, the attribute value is mandatory (not optional) and not unique.

External, Internal, and Dynamic can be used alone or in combination with the Optional or Unique keywords. These keywords specify the kind of ownership the entity has over attribute values. If none of these keywords are used, Dynamic is assumed.

External says that the attribute value exists outside the scope of the entity, and potentially can be shared by several entities.

Internal says that the attribute value exists within the scope of the entity, and cannot be shared by any other entity.

Dynamic says that the attribute can exist within an entity instance sometimes and outside it at other times.

A.11.3.2.2 Derived Attribute

Description

A derived attribute represents information that is computed in some manner. Derived attributes must be declared following the Derive keyword. Each derived attribute must be specified individually. The type of the attribute follows the colon as in an explicit attribute declaration; then the derivation is given as an expression. All of the parameters used in the expression must be literals or explicit or derived attributes of the entity being declared.

Syntax

196 DERIVED-ATTRIBUTE-DECL = DERIVE DERIVED-ATTRIBUTE { DERIVED-ATTRIBUTE } .
195 DERIVED-ATTRIBUTE = ATTRIBUTE-ID COLON RESULT-CAN-BE INITIALIZER SEMICOLON .
219 INITIALIZER = ASSIGNMENT-OPERATOR EXPRESSION .

Examples

In the following example a circle is defined by three points through which it passes. These three points represent the data by which the circle is defined. In addition to these data there is a need to account for important traits such as the radius, or the axis. This is accomplished by defining them as derived attributes, giving the value as an expression.

FUNCTION f1(p1,p2,p3:point):real :
 (* Compute RADIUS of circle given three points on circumference *)
END_FUNCTION

FUNCTION f2(p1,p2,p3:point):vector ;
/* Compute AXIS of circle given three points on circumference */
END_FUNCTION ;

ENTITY circle ;
  p1, p2, p3 : point ;
DERIVE
  radius : real := f1(p1,p2,p3) ;
  axis : vector := f2(p1,p2,p3) ;
  area : real := pi*radius**2 ;
END_ENTITY ;

The three defining points (P1, P2, and P3) are explicit attributes. Radius, Axis, and Area are derived information. The values of these derived attributes are computed by the expression following the assignment operator. The values of Radius and Axis are obtained by means of a function call; the value of Area is computed in-line. Note that the type given after the colon must be assignment compatible with the result produced by the expression.

A.11.3.3 Uniqueness Rule

Description

A uniqueness constraint can be given for individual explicit attributes or combinations of them by means of a Unique rule. The uniqueness rules follow the Unique keyword. Each rule lists the attribute names whose values must be unique. When such a list has two or more attribute names, the values are jointly unique.

If an entity had three attributes called A, B, and C, we could have:

UNIQUE
  A;
  B;
  C;

which means that the uniqueness constraint applies to each of the attributes individually. Or we could have:

UNIQUE
  A.B;
  B;
  C;

This means that the uniqueness constraint applies jointly to A and B together, and also to B and C individually.

Examples
A person_name entity might look like:

```plaintext
ENTITY PERSON_NAME;
LAST   : STRING;
FIRST  : STRING;
MIDDLE : STRING;
NICKNAME : STRING;
END_ENTITY;
```

and it might be used as:

```plaintext
ENTITY EMPLOYEE;
BADGE   : NUMBER;
NAME    : PERSON_NAME;
...
UNIQUE  
BADGE, NAME;
...
END_ENTITY;
```

and as:

```plaintext
ENTITY ROSTER;
NAME     : LIST[0:] OF PERSON_NAME;
...
END_ENTITY;
```

The uniqueness constraint is not stated within the person_name entity since it is not clear that names always have to be different. The roster entity does not insist that names are different. In the case of an employee, however, there is a requirement for unique identification. The example shows that BADGE and NAME together have to be unique. This allows the value of BADGE in one employee entity instance to be the same as another instance; likewise for the value of NAME. The two values taken together have to be unique however.

Uniqueness spans some collection of entity instances. That collection could reasonably be called a database. So, within a given database, the attribute, or set of attributes, or combination of attributes must be distinctly different throughout.

### A.11.3.4 Local Rule

**Description**

Local rules are used to specify conditions that constrain the value of individual attributes or a combinations of attributes. All local rules must follow the Where keyword.

**Syntax**
Each expression must yield a logical (True or False) result. The variables used in the expressions must be the names of attributes that are declared for this entity.

For example, a unit_vector requires that the length of the vector be exactly one. This constraint can be specified by:

```plaintext
ENTITY unit_vector ;
    a, b, c : real ;
WHERE
    a**2 + b**2 + c**2 = 1.0 ;
END_ENTITY ;
```

Local rules deal with optional attributes sometimes. That part of a rule which contains an optional attribute is treated as if it did not exist when the value of that attribute does not exist. Modifying the previous example:

```plaintext
ENTITY unit_vector ;
    a, b : real ;
    c : optional real ;
WHERE
    a**2 + b**2 + c**2 = 1.0 ;
END_ENTITY ;
```

When `c` has a value the rule expression is evaluated normally, but when `c` has no value the subexpression `+ c**2` has no evaluation.

### A.11.4 Map Declaration

**Description**

A Map is a declaration that modifies an entity for some purpose. The effect of declaring a Map is to declare an entity. Mapping can give a different name to an entity, alter the type of an attribute (if the new type is compatible with the old type), introduce new subtypes and supertypes, and add local rules.

**Syntax**

228 MAP-DECL = MAP MAP-ID FROM ENTITY-ID [ SUBSUPER-DECL ]
    SEMICOLON { RETYPE-DECL } [ UNIQUE-RULE ]
    [ WHERE-RULE ] END-MAP SEMICOLON .
263 UNIQUE-RULE = UNIQUE { ID-LIST SEMICOLON } .
266 WHERE-RULE = WHERE { EXPRESSION SEMICOLON } .
Examples

A vertex entity is defined as:

```plaintext
ENTITY VERTEX;
    PT : OPTIONAL POINT;
END_ENTITY;
```

A specific application of the vertex entity (i.e., another schema) requires that the point component of vertex is mandatory. This is accomplished by:

```plaintext
MAP VERTEX FROM VERTEX;
    WHERE
        (_EXISTS(PT));
    END_MAP;
```

This example assumes that there is an entity named vertex visible in the scope of the mapped entity (also called vertex). The attributes of the mapped vertex do not have to be written as they (in this case) are automatically part of the mapped entity. The local rule states that the attribute named pt must not be null, thus making it mandatory.

Another application wishes to call an entity defined some place else by a different name. The definition is in all other respects the same. This is done by:

```plaintext
MAP NODE FROM POINT;
    END_MAP;
```

which simply allows an application to call Point by a more familiar name.

Even though the declaration is called Map an Entity declaration is created in the current schema block. Using a map is preferable to duplicating an entity (with appropriate changes) since mapping establishes the logical connection between the two entities.

A.11.5 Rule

Description

A Where clause in an entity declaration provides a set of constraints that apply within a given entity instance. The Rule permits the definition of constraints on the model as a whole.

Syntax

```plaintext
243 RULE-DECL = RULE RULE-ID FOR NESTED-ID-LIST SEMICOLON
    ACTION-BODY END-RULE SEMICOLON
```

The rule header names the rule and specifies the entities affected by the rule. This statement is similar to a formal parameter list except the "parameters" are themselves entity types. Each of the entity types is regarded as a set of instances of that type; i.e., the entity set that makes
up a model.

The body of the rule is much like the body of a function or procedure. The chief difference is that the "end state" of a rule is to indicate whether or not some global constraint is satisfied. The rule is satisfied if upon termination, the violation built-in procedure never was executed.

Examples

The following rule declares that for each point in the first octant, there must be a point in the seventh octant.

```
RULE POINT_MATCH FOR (POINT);
LOCAL
  K1, K2 : INTEGER;
END_LOCAL;
K1 := 0;
K2 := 0;
REPEAT FOR EACH POINT IN MODEL;
  IF ((POINT.X>0) AND (POINT.Y>0) AND (POINT.Z>0)) THEN
    K1 := K1+1;
  END_IF;
  IF ((POINT.X<0) AND (POINT.Y<0) AND (POINT.Z<0)) THEN
    K2 := K2+1;
  END_IF;
END_REPEAT;
IF (K1<>K2) THEN
  VIOUTION;
END_IF;
END_RULE;
```

A.11.6 Algorithms

An algorithm is a sequence of statements that produces some desired end state. The two kinds of algorithms that can be specified are procedures and functions (a rule is also an algorithm, but is discussed separately).

Formal parameters define the input to the algorithm. When the algorithm is called, actual parameters provide actual values. The actual parameters must agree in type, order, and number with the formal parameters.

Declarations local to the algorithm are given following the header. These declarations can be types, local variables, other algorithms, etc. as needed.

The body of the algorithm follows local declarations.

A.11.6.1 Parameters

Description
A Function or Procedure can have formal parameters. Each formal parameter has a name and a type. The name is treated as an identifier that is unique within the scope of the function or procedure. A formal parameter can also be declared as var (variable), which means that if the parameter is changed within the function the change is apparent to the point of invocation. Parameters not declared as var can be changed also, but the change will not be apparent when control is returned to the caller.

Syntax


Examples

FUNCTION dist(p1,p2:point):real ;
PROCEDURE midpt(p1,p2:point; var result:point) ;

A.11.6.2 Local Variables

Description

A Local variable has the same scope as a formal parameter and exists only within the algorithm in which it is declared. A local variable can be assigned a value or can participate in an expression.

Syntax

226 LOCAL-DECL = LOCAL { LOCAL-BODY-ITEM } END-LOCAL SEMICOLON .
181 AGGREGATION-TYPE = AGGREGATE-TYPE | ARRAY-TYPE | LIST-TYPE | SET-TYPE .
222 LIMIT-SPEC = LEFT-BRACKET EXPRESSION COLON ( EXPRESSION | INFINITY ) RIGHT-BRACKET .
182 ARRAY-TYPE = ARRAY INDEX-SPEC OF [ OPTIONAL ] [ UNIQUE ] ATTRIBUTE-CAN-BE .
218 INDEX-SPEC = LEFT-BRACKET EXPRESSION COLON EXPRESSION RIGHT-BRACKET .
186 BASE-TYPE = INTEGER-TYPE | LOGICAL-TYPE | NUMBER-TYPE | REAL-TYPE | STRING-TYPE .
220 INTEGER-TYPE = INTEGER [ PRECISION-SPEC ] .
233 PRECISION-SPEC = LEFT-PAREN EXPRESSION RIGHT-PAREN .
227 LOGICAL-TYPE = LOGICAL .
231 NUMBER-TYPE = NUMBER .
When an algorithm is activated, numbers are initialized to zero, strings are empty, logicals are False and lists and sets are empty unless an initializer is explicitly given. The initializer value is given to the local variable upon entry to the algorithm.

A.11.6.3 Procedure

Description

A Procedure is an algorithm that receives parameters from the point of invocation, operates on those parameters in some manner to produce the desired end state. The value of those parameters might be changed. A given parameter can be altered only if the Var attribute is present for that parameter.

Syntax

235 PROCEDURE-DECL = PROCEDURE PROCEDURE-ID [ FORMAL-PARAMETER-LIST ]
  SEMICOLON ACTION-BODY END-PROCEDURE SEMICOLON .

A.11.6.4 Function

Description

A Function is an algorithm that operates on parameters and produces a specific result type. The invocation of a function is equivalent to having a variable of that result type at the point of invocation.

Syntax

211 FUNCTION-DECL = FUNCTION FUNCTION-ID [ FORMAL-PARAMETER-LIST ]
  COLON RESULT-CAN-BE SEMICOLON ACTION-BODY END-FUNCTION SEMICOLON .

function-id is the name of the function. formal-parameter-list is zero or more parameters supplying values for the function to operate on. result-can-be is the result produced by the function.

A Return statement is used to pass the function result to the point of invocation. Return also terminates the function. It is illegal to terminate a function except by a return statement with an argument.
A.11.6.5 Executable Statements

The executable statements are Assignment Case Compound Escape If Procedure call Repeat Return Skip and With.

Executable statements can appear only within an execution block.

An executable statement can be just a semicolon, in which case it is called a null statement.

Examples

IF A = 13 THEN
    ; -- this is a null statement.
ELSE
    B := 5 ;
END_IF ;

Syntax

251  STATEMENT = ASSIGNMENT-STATEMENT | CASE-STATEMENT
     | COMPOUND-STATEMENT | DUMMY-STATEMENT | ESCAPE-STATEMENT
     | IF-STATEMENT | PROCEDURE-CALL-STATEMENT | REPEAT-STATEMENT
     | RETURN-STATEMENT | SKIP-STATEMENT | WITH-STATEMENT .
183  ASSIGNMENT-STATEMENT = ID ASSIGNMENT-OPERATOR EXPRESSION SEMICOLON .
191  CASE-STATEMENT = CASE EXPRESSION OF CASE-BLOCK END-CASE SEMICOLON .
189  CASE-BLOCK = CASE-ACTION { CASE-ACTION } [ CASE-OTHERWISE ] .
188  CASE-ACTION = EXPRESSION { COMMA EXPRESSION } COLON STATEMENT .
190  CASE-OTHERWISE = OTHERWISE COLON STATEMENT .
192  COMPOUND-STATEMENT = BEGIN { STATEMENT } END SEMICOLON .
198  DUMMY-STATEMENT = SEMICOLON .
202  ESCAPE-STATEMENT = ESCAPE SEMICOLON .
215  IF-STATEMENT = IF EXPRESSION THEN STATEMENT { STATEMENT }
     [ ELSE STATEMENT { STATEMENT } ] END-IF .
239  REPEAT-STATEMENT = REPEAT { REPEAT-CONTROL } SEMICOLON
     { STATEMENT } END-REPEAT SEMICOLON .
241  RETURN-STATEMENT = RETURN [ LEFT-PAREN EXPRESSION RIGHT-PAREN ]
     SEMICOLON
250  SKIP-STATEMENT = SKIP SEMICOLON .
268  WITH-STATEMENT = WITH ID STATEMENT .

A.11.6.5.1 Assignment Statement

Description
The assignment statement is used to assign a value to a variable.

**Syntax**

```
183 ASSIGNMENT-STATEMENT = ID ASSIGNMENT-OPERATOR EXPRESSION SEMICOLON .
```

**Examples**

```
LOCAL
    A, B, C : INTEGER ;
END_LOCAL ;

A := 1 ;
B := 2 ;
C := A+B ;
```

### A.11.6.5.2 Assignment Compatibility

An expression can be assigned to a variable only when the type of the expression is compatible with the type of the variable. Types can be compatible without being identical. Types are assignment compatible when:

1. The types are the same. Different user defined types are never the same even if their base types are the same.
2. One type is a subtype of the other type; e.g., integers and reals are subtypes of number, and integer is a subtype of real.
3. Both types are strings.
4. Both types are arrays where the bounds and the base types are identical.
5. Both types are lists with identical base types.
6. Both types are sets with identical base types.

### A.11.6.5.3 Case Statement

**Description**

The Case statement causes the selection and execution of many possible statements depending on the value of a selector expression. The case statement consists of an expression, which is the case selector, and a list of alternative actions, each one preceded by a case label. The type of the case label must agree with the type of the case selector. The statement having the case label that matches the value of the case selector is executed. If none of the case labels match the value of the case selector then:

1. if the Otherwise clause is present the statement associated with otherwise is executed,
2. if the Otherwise clause is not present no statement (in the CASE) is executed.

Syntax

\[
\text{CASE-STATEMENT} = \text{CASE EXPRESSION OF CASE-BLOCK END-CASE SEMICOLON}.
\]

Examples

```plaintext
LOCAL
   a : integer ;
   x : real ;
END_LOCAL ;

a := 3 ;
x := 34.97 ;
CASE a OF
   1 : x := \sin(x) ;
   2 : x := \exp(x) ;
   3 : x := \sqrt(x) ; -- This is executed!
   4 : x := \log(x) ;
END_CASE ;
```

Restrictions:

- The type of the expression following case must agree with the expression that is part of the case-action.
- Only one otherwise case-action can be given. It must be the last case-action if it appears at all.
- Each case label must be different. However, it is possible that two case label expressions will evaluate the same. In that event, the first reading from top to bottom will be executed.

A.11.6.5.4 Compound Statement

Description

The compound statement is a sequence of statements delimited by Begin and End. A compound statement is treated as a single logical statement.

Syntax

\[
\text{COMPOUND-STATEMENT} = \text{BEGIN \{} \text{STATEMENT} \} \text{ END SEMICOLON}.
\]
Examples

BEGIN
  A = A+1;
  IF A > 100 THEN
    A := 0;
  END_IF;
END;

A.11.6.5.5 Escape Statement

Description

Escape causes an immediate transfer to the statement following the end of the block in which it appears. Note that this is the only way an indefinite repeat can be terminated.

Syntax

202 ESCAPE-STATEMENT = ESCAPE SEMICOLON.

Examples

... REPEAT UNTIL (A=1);
... IF (A<0) THEN
  ESCAPE: -- When executed, control passes to the statement
  END_IF: after END_REPEAT.
...
END_REPEAT;
...

A.11.6.5.6 If Then Else Statement

Description

The If statement provides for the conditional execution of a statement according to the value of a logical (true or false) expression. The statement following Then is executed when the logical expression evaluates to True. When the expression evaluates to False the statement associated with Else is executed.

The Else clause is optional however. If an else clause is not present, and the result of the logical expression is False then the if statement has no effect.

Syntax
215 IF-STATEMENT = IF EXPRESSION THEN STATEMENT (STATEMENT) [ELSE STATEMENT (STATEMENT)] END-IF.

Examples

LOCAL
    A: INTEGER;
END_LOCAL;

A = 5;
IF A < 10 THEN
    A = A+1; -- THIS STATEMENT WILL BE EXECUTED.
ELSE
    A = A-1;
END_IF;

A.11.6.5.7 Procedure Call Statement

Description

The Procedure call statement activates a procedure. The statement consists of a procedure identifier followed by an actual parameter list (which can be empty). The number and type of the actual parameters must agree with the formal parameters defined for that procedure.

Syntax

234 PROCEDURE-CALL-STATEMENT = PROCEDURE-ID [ACTUAL-PARAMETER-LIST] SEMICOLON.

Examples

The following procedure header defines two formal parameters; A, which is an integer variable, and B, which is a real. The actual parameters used with any call to this procedure must agree in number and type.

PROCEDURE XXX(VAR A:INTEGER; B:REAL);

The following procedure calls show both valid and invalid cases.

XXX(1.2); (* Invalid since the first actual parameter is not a variable *)
XXX(a, 37); (* This is a valid procedure call (assuming that a is typed as an integer). The second actual parameter, although not a real, is valid since a type conversion takes place automatically *)
XXX(a); (* This call is not valid since the correct number of
actual parameters is not given *)

III(a.2.0.3); (* Invalid, too many actual parameters *)

A.11.6.5.8 Repeat Statement

Description

The Repeat block is used to repeat a sequence of statements. The controlling conditions are finite iteration, while a condition is true, until a condition is true, and indefinite repetition. These controls can be used in combination to specify the conditions that terminate the repeat block.

Syntax

\[ \text{239 REPEAT-STATEMENT} = \text{REPEAT} \{ \text{REPEAT-CONTROL} \} \text{ SEMICOLON} \{ \text{STATEMENT} \} \text{ END-REPEAT SEMICOLON} . \]

Note – The repeat controls can be given in any order, but the same kind of control cannot be given more than once.

A.11.6.5.9 Until Control

Description

Causes the block to repeat Until the expression is true. The expression will be evaluated after each iteration. At least one iteration will always be executed.

Syntax

\[ \text{264 UNTIL-CONTROL} = \text{UNTIL EXPRESSION} . \]

A.11.6.5.10 While Control

Description

Causes the block to repeat While the expression is true. The expression will be evaluated before each iteration. If the expression is false initially, the block will not be executed.

Syntax

\[ \text{267 WHILE-CONTROL} = \text{WHILE EXPRESSION} . \]
A.11.6.5.11 Increment Control

Description

Causes the block to repeat while the active variable lies in the given interval. The active variable is initialized as the first bounding condition. After each iteration the active variable is incremented (or decremented) as specified. Before each iteration, the test:

{ expression1 <= active-var <= expression2 }

is evaluated. The block is executed only when this test yields true. expression3, if present, updates the value of the active variable after each iteration. If expression3 is omitted, the active variable is updated by one (1).

Syntax

217 INCREMENT-CONTROL = LOCAL-ID ASSIGNMENT-OPERATOR EXPRESSION TO EXPRESSION [ BY EXPRESSION ] .

A.11.6.5.12 Return Statement

Description

The Return statement terminates the execution of a function or procedure. A function return must include an expression, which is the result returned to the point of invocation. The expression must be the same type as the declared type of the function. A procedure return must not include an expression.

Syntax

241 RETURN-STATEMENT = RETURN [ LEFT-PAREN EXPRESSION RIGHT-PAREN ] SEMICOLON

Examples

RETURN(50) ;       (* from a function *)
RETURN(work_point) ;
RETURN ;           (* from a procedure *)

A.11.6.5.13 Skip Statement

Description

Skip causes an immediate transfer to the end of the block in which it appears. If the block is a repeat block, the terminal conditions are tested. The block will be repeated or terminated based on evaluation of the repeat controls.

Syntax
Examples

```plaintext
REPEAT UNTIL (A=1);
...
IF (A<0) THEN
  SKIP;  (* When executed, control passes to the END_REPEAT
  statement where the terminal condition is
  evaluated. *)
END_IF;
...
END_REPEAT;
...
```

A.11.6.5.14 With Statement

Description

The With statement provides a partial variable qualification that simplifies the writing of separate components of a complex object, i.e., an entity. The with statement starts with the keyword With followed by a partial qualification, followed by a statement.

Syntax

```plaintext
268 WITH-STATEMENT = WITH ID STATEMENT .
```

Examples

```plaintext
local
  a : point ;  (* point has attributes x, y, and z *)
  b : number ;
end_local ;
  WITH a
  begin ;
  b := x + y + z ;  (* same as: a.x + a.y + a.z *)
end ;
```
A.12 Expression

Description

Expressions are made up of operators and operands. Operands must be compatible with operators. Binary operators require two operands and the operator appears between those operands. A unary operator requires only one operand, which is given before the operand.

There are three kinds of expressions.

- Arithmetic expressions accept numeric operands and produce a numeric result.
- Relational expressions accept various kinds of operands and produce a logical (true or false) result.
- String expressions accept string operands and produce a string result.

Operators have computational precedence as defined below. Evaluation proceeds from left to right with the highest precedence being evaluated first.

<table>
<thead>
<tr>
<th>Operator Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Unary operators</td>
</tr>
<tr>
<td>2 Exponentiation</td>
</tr>
<tr>
<td>3 Multiplication and Division</td>
</tr>
<tr>
<td>4 Addition and Subtraction</td>
</tr>
<tr>
<td>5 Relational</td>
</tr>
</tbody>
</table>

An operand between two operators of different precedence is bound to the operator with the higher precedence. e.g., -10*20 or (-10)*20.

An operand between two operators of the same precedence is bound to the one on the left. e.g., 10/20*30 or (10/20)*30

Expression enclosed by parentheses are evaluated before being treated as a single operand.

Syntax

259 TERM = FACTOR { MULTIPLICATION-LIKE-OPERATIONS FACTOR } .
207 FACTOR = SIMPLE-FACTOR [ EXPONENTIATION-OPERATOR SIMPLE-FACTOR ] .
249 SIMPLE-FACTOR = LITERAL | FUNCTION-CALL | ID | SET-CONSTRUCT | LEFT-PAREN EXPRESSION RIGHT-PAREN | INTERVAL | NOT-OPERATOR SIMPLE-FACTOR .
A.12.1 Arithmetic Expressions

Description

The arithmetic operations are add (+), subtract (−), divide (/), integer divide (DIV), multiply (•), exponentiation (**), and modulo (MOD). Both operands must be numbers and the result is a number.

The type of the result of these operations, based on the type of the operands, is:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operand-Types</th>
<th>Result-Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Add</td>
<td>real</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>real</td>
</tr>
<tr>
<td>- Subtract</td>
<td>real</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>real</td>
</tr>
<tr>
<td>* Multiply</td>
<td>real</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>real</td>
</tr>
<tr>
<td>/ Divide</td>
<td>real</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>integer</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>real</td>
</tr>
<tr>
<td>DIV</td>
<td>Integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>integer</td>
</tr>
<tr>
<td>MOD</td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>(remainder)</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>** Exponentiation</td>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>real</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>integer, real</td>
<td>real</td>
</tr>
<tr>
<td></td>
<td>real, integer</td>
<td>real</td>
</tr>
</tbody>
</table>

* Real operands are first truncated to integers.
A.12.2 Relational Expressions

Description

The value relational operators are equal (\(=\)), not equal (\(<>\)), greater than (\(>\)), less than (\(<\)), greater than or equal (\(\geq\)), less than or equal (\(\leq\)), set membership (\(\text{In}\)), and string match (\(\text{like}\)).

The instance relational operators are instance equal (\(::\)) and instance not equal (\(::<>\)).

The cardinality relational operator has the form [expression: expression].

The two operands must be compatible; e.g., a string cannot be compared to a number in this manner.

The result of a relational expression is a logical value true or false. The not operator may precede the operation to reverse the result.

Examples

\[
\begin{align*}
27 & < 84 \quad (* \text{true} *) \\
\text{not} (27 < 84) & (* \text{false} *)
\end{align*}
\]

A.12.3 Interval Expressions

Description

An interval expression tests whether or not a value falls within a given interval yielding a True or False result.

Syntax

\[
\begin{align*}
221 \text{ INTERVAL} & = \text{LEFT-CURL} \text{ EXPRESSION} \text{ RELATIONAL-OPERATOR} \text{ EXPRESSION} \\
& \text{ RELATIONAL-OPERATOR} \text{ EXPRESSION} \text{ RIGHT-CURL} .
\end{align*}
\]

Examples

\[
\text{IF} \{ 10 \leq A < \text{MAXVAL} \} \text{ THEN}
\]

69
A.12.4 String Expressions

Description

The only string operation is string concatenation represented by the `||` symbol. Both operands must be strings and the result of a concatenation is also a string.

Examples

```plaintext
name := last_name || ' ' || first_name ;
```

A.12.5 Function Call

Description

The function call activates a function. It consists of a function identifier possibly followed by an actual parameter list. The number and type of the actual parameters must agree with the formal parameters defined for that function.

A function call can be qualified when the function returns some kind of aggregate type. "dot" qualification allows you to access individual attributes when the return type is an entity type. Subscript notation can be used when the return type is an array, list, or set type.

Examples

```plaintext
ENTITY POINT;
   X, Y, Z : NUMBER;
END_ENTITY;

FUNCTION MIDPOINT_OF_LINE(L:LINE):POINT;
   ...;
END_FUNCTION;

IF MIDPOINT_OF_LINE(L506).X=0.0 THEN
   ...;
END_IF;
```

Syntax

```plaintext
FUNCTION-CALL = FUNCTION-ID [ ACTUAL-PARAMETER-LIST ] { QUALIFICATION } .
QUALIFICATION = { SUBSCRIPT } [ DOT ID ] .
```

Examples

```plaintext
a := f1(1,2,3) ;
```
A.12.6 Set Construct

The set construct is used to establish a set value.

Syntax

\[ \text{SET-CONSTRUCT} = \text{LEFT-BRACKET} \{ \text{EXPRESSION} \{ \text{COMMA} \text{EXPRESSION} \} \} \text{RIGHT-BRACKET} . \]

Examples

Given the declaration:

\[ a : \text{set of integer}; \]

a value can be assigned as:

\[ a := [ 1, 3, 6, 98, -12 ]; \]
A.13 Express Specification

The formal definition of the *Express Language* is given in this section. This specification is organized as follows:

- Syntax of the specification
- Cross reference for rules
- Rules

A.13.1 The Syntax of the Specification

Regular expressions define the tokens of the *Express* language and the statements that can legally be constructed from those tokens. Tokens are composed of characters; statements are composed of tokens. Tokens can be constants or variables. For example, keywords and punctuation are token constants and identifiers and literals are token variables.

The general form of a regular expression is:

```
name = expression .
```

where

- `name` is the symbolic name given to expression where the name is made up entirely of letters, digits, or the dash (-) character.
- `=` is the assignment of a regular expression to a name
- `expression` is some valid regular expression composed from the following elements:
  - `name` is a token or statement defined somewhere in the specification according to the definition given above.
  - `'x'` is the character x or X (case insensitive). For example, 'xy' is the string xy, xY, Xy, or XY (case insensitive).
  - `'''` is the character ' 
  - `|` is the choice operator, e.g., 'x'|'y' means either x or y
  - `[element]` means element occurs zero or one times
  - `{element}` means element occurs zero or many times
  - `(element)` groups the element(s) enclosed by parentheses. e.g., 'x'('y'|'z') means xy or xz.
  - `.` the period terminates a production.

The following are system dependent characters or character sets. For example, `\t` is the Tab character, which may or may not have meaning to a particular system.

- `\a` is any printable ASCII character.
- `\n` is the newline character
- `\q` is the quote (') character.
- `\t` is the tab character
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>1</td>
</tr>
<tr>
<td>ACOS</td>
<td>2</td>
</tr>
<tr>
<td>ACTION-BODY</td>
<td>178</td>
</tr>
<tr>
<td>ACTUAL-PARAMETER-LIST</td>
<td>179</td>
</tr>
<tr>
<td>ADDITION-LIKE-OPERATIONS</td>
<td>134</td>
</tr>
<tr>
<td>AGGREGATE</td>
<td>3</td>
</tr>
<tr>
<td>AGGREGATE-TYPE</td>
<td>180</td>
</tr>
<tr>
<td>AGGREGATION-TYPE</td>
<td>181</td>
</tr>
<tr>
<td>AND</td>
<td>4</td>
</tr>
<tr>
<td>ARRAY</td>
<td>5</td>
</tr>
<tr>
<td>ARRAY-TYPE</td>
<td>182</td>
</tr>
<tr>
<td>AS</td>
<td>6</td>
</tr>
<tr>
<td>ASIN</td>
<td>7</td>
</tr>
<tr>
<td>ASSIGNMENT-OPERATOR</td>
<td>8</td>
</tr>
<tr>
<td>ASSIGNMENT-STATEMENT</td>
<td>183</td>
</tr>
<tr>
<td>ASSUME</td>
<td>9</td>
</tr>
<tr>
<td>ASSUME-DIRECTIVE</td>
<td>184</td>
</tr>
<tr>
<td>ATAN</td>
<td>10</td>
</tr>
<tr>
<td>ATTRIBUTE-CAN-BE</td>
<td>185</td>
</tr>
<tr>
<td>ATTRIBUTE-ID</td>
<td>136</td>
</tr>
<tr>
<td>BASE-TYPE</td>
<td>186</td>
</tr>
<tr>
<td>BEGIN</td>
<td>11</td>
</tr>
<tr>
<td>BLOCK-MEMBER</td>
<td>187</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>BUILT-IN-PROCEDURE-NAME</td>
<td>138</td>
</tr>
<tr>
<td>BY</td>
<td>12</td>
</tr>
<tr>
<td>CARDINALITY-OPERATOR</td>
<td>139</td>
</tr>
<tr>
<td>CASE</td>
<td>13</td>
</tr>
<tr>
<td>CASE-ACTION</td>
<td>188</td>
</tr>
<tr>
<td>CASE-BLOCK</td>
<td>189</td>
</tr>
<tr>
<td>CASE-OTHERWISE</td>
<td>190</td>
</tr>
<tr>
<td>CASE-STATEMENT</td>
<td>191</td>
</tr>
<tr>
<td>COLON</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Entity Type</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
</tr>
<tr>
<td>ENTITY-TYPE</td>
<td>200</td>
</tr>
<tr>
<td>ATTRIBUTE-CAN-BE</td>
<td>185</td>
</tr>
<tr>
<td>LOCAL-CAN-BE</td>
<td>226</td>
</tr>
<tr>
<td>PARAMETER-CAN-BE</td>
<td>232</td>
</tr>
<tr>
<td>RESULT-CAN-BE</td>
<td>240</td>
</tr>
<tr>
<td>ENUMERATION</td>
<td>37</td>
</tr>
<tr>
<td>ENUMERATION-TYPE</td>
<td>201</td>
</tr>
<tr>
<td>TYPE-CAN-BE</td>
<td>260</td>
</tr>
<tr>
<td>EQUAL</td>
<td>38</td>
</tr>
<tr>
<td>EQUAL-OPERATOR</td>
<td>145</td>
</tr>
<tr>
<td>RELATIONAL-OPERATOR</td>
<td>170</td>
</tr>
<tr>
<td>ESPAPE</td>
<td>39</td>
</tr>
<tr>
<td>ESCAPE-STATEMENT</td>
<td>202</td>
</tr>
<tr>
<td>STATEMENT</td>
<td>251</td>
</tr>
<tr>
<td>EVERYTHING</td>
<td>40</td>
</tr>
<tr>
<td>EXPORT-DIRECTIVE</td>
<td>205</td>
</tr>
<tr>
<td>EXCEPT</td>
<td>41</td>
</tr>
<tr>
<td>EXPORT-DIRECTIVE</td>
<td>205</td>
</tr>
<tr>
<td>EXCLUSIVE-OR-OPERATOR</td>
<td>42</td>
</tr>
<tr>
<td>OR-OPERATOR</td>
<td>166</td>
</tr>
<tr>
<td>EXISTS</td>
<td>43</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>EXP</td>
<td>44</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>EXPLICIT-ATTRIBUTE-DECL</td>
<td>204</td>
</tr>
<tr>
<td>ENTITY-DECL</td>
<td>199</td>
</tr>
<tr>
<td>EXPOSENTIATION-OPERATOR</td>
<td>45</td>
</tr>
<tr>
<td>FACTOR</td>
<td>207</td>
</tr>
<tr>
<td>EXPORT-DIRECTIVE</td>
<td>205</td>
</tr>
<tr>
<td>DIRECTIVE</td>
<td>197</td>
</tr>
<tr>
<td>EXPRESSION</td>
<td>206</td>
</tr>
<tr>
<td>ACTUAL-PARAME-B LIST</td>
<td>179</td>
</tr>
<tr>
<td>FACTOR</td>
<td>207</td>
</tr>
<tr>
<td>FALSE</td>
<td>48</td>
</tr>
<tr>
<td>CONSTANT</td>
<td>140</td>
</tr>
<tr>
<td>LOGICAL-LITERAL</td>
<td>160</td>
</tr>
<tr>
<td>FORMAL-PARAMETER</td>
<td>208</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>FROM</td>
<td>51</td>
</tr>
<tr>
<td>MAP-DECL</td>
<td>228</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>62</td>
</tr>
<tr>
<td>FUNCTION-DECL</td>
<td>211</td>
</tr>
<tr>
<td>FUNCTION-ID</td>
<td>147</td>
</tr>
<tr>
<td>FUNCTION-CALL</td>
<td>210</td>
</tr>
<tr>
<td>SIMPLE-FACTO</td>
<td>249</td>
</tr>
<tr>
<td>BLOCK-MEMBER</td>
<td>187</td>
</tr>
<tr>
<td>GENERAL-AGGREGATION</td>
<td>212</td>
</tr>
<tr>
<td>PARAMETER-CAN-BE</td>
<td>232</td>
</tr>
<tr>
<td>GENERIC</td>
<td>53</td>
</tr>
<tr>
<td>GENERIC-TYPE</td>
<td>213</td>
</tr>
<tr>
<td>RESULT-CAN-BE</td>
<td>240</td>
</tr>
<tr>
<td>GREATER-EQUAL-OPERATOR</td>
<td>54</td>
</tr>
<tr>
<td>RELATIONAL-OPERATOR</td>
<td>170</td>
</tr>
<tr>
<td>GREATER-TAN-OPERATOR</td>
<td>55</td>
</tr>
<tr>
<td>RELATIONAL-OPERATOR</td>
<td>170</td>
</tr>
<tr>
<td>HIBOUND</td>
<td>56</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>ID</td>
<td>148</td>
</tr>
<tr>
<td>ID-INDEXED</td>
<td>150</td>
</tr>
<tr>
<td>LOCAL-ID</td>
<td>159</td>
</tr>
</tbody>
</table>

Assignment Statement (183)
Case-Action (188)
Case-Statement (191)
If-Statement (215)
Increment-Control (217)
Index-Spec (218)
Initializer (219)
Interval (221)
Limit-Spec (222)
Precision-Spec (233)
Return-Statement (241)
Set-Construct (246)
Simple-Factor (249)
Subscript (253)
Until-Control (264)
Where-Rule (266)
While-Control (267)

Assignment Statement (183)
Case-Action (188)
Case-Statement (191)
If-Statement (215)
Increment-Control (217)
Index-Spec (218)
Initializer (219)
Interval (221)
Limit-Spec (222)
Precision-Spec (233)
Return-Statement (241)
Set-Construct (246)
Simple-Factor (249)
Subscript (253)
Until-Control (264)
Where-Rule (266)
While-Control (267)
<table>
<thead>
<tr>
<th>Token</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID-CHAR</td>
<td>149</td>
</tr>
<tr>
<td>ID-INDEXED</td>
<td>150</td>
</tr>
<tr>
<td>ID-LIST</td>
<td>214</td>
</tr>
<tr>
<td>ID-QUALIFIED</td>
<td>161</td>
</tr>
<tr>
<td>ID-SIMPLE</td>
<td>152</td>
</tr>
<tr>
<td>IF</td>
<td>57</td>
</tr>
<tr>
<td>IF-STATEMENT</td>
<td>215</td>
</tr>
<tr>
<td>IN</td>
<td>58</td>
</tr>
<tr>
<td>IN-OPERATOR</td>
<td>153</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>59</td>
</tr>
<tr>
<td>INCLUDE-DIRECTIVE</td>
<td>216</td>
</tr>
<tr>
<td>INCLUSIVE-OR-OPERATOR</td>
<td>60</td>
</tr>
<tr>
<td>INCREMENT-CONTROL</td>
<td>217</td>
</tr>
<tr>
<td>INDEX-SPEC</td>
<td>218</td>
</tr>
<tr>
<td>INFINITY</td>
<td>61</td>
</tr>
<tr>
<td>_INITIALIZER</td>
<td>219</td>
</tr>
<tr>
<td>INSERT</td>
<td>62</td>
</tr>
<tr>
<td>INSTANCE-EQUAL-OPERATOR</td>
<td>63</td>
</tr>
<tr>
<td>INSTANCE-NOTEQUAL-OPERATOR</td>
<td>64</td>
</tr>
<tr>
<td>INTEGER</td>
<td>65</td>
</tr>
<tr>
<td>INTEGER-DIVIDE-OPERATOR</td>
<td>164</td>
</tr>
<tr>
<td>INTEGER-LITERAL</td>
<td>155</td>
</tr>
<tr>
<td>INTEGER-TYPE</td>
<td>220</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>66</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>221</td>
</tr>
<tr>
<td>LEFT-BRACKET</td>
<td>67</td>
</tr>
<tr>
<td>LEFT-CURL</td>
<td>68</td>
</tr>
<tr>
<td>LEFT-PAREN</td>
<td>69</td>
</tr>
<tr>
<td>LESS-EQUAL-OPERATOR</td>
<td>70</td>
</tr>
<tr>
<td>LESS-THAN-OPERATOR</td>
<td>71</td>
</tr>
</tbody>
</table>

- QUALIFICATION(236) SIMPLE-FACTOR(249)
- WITH-STATEMENT(268)
- ID-CHAR(149) ID-SIMPLE(152)
- ID-INDEXED(150) ID-QUALIFIED(151)
- ID-LIST(214) EXPLICIT-ATTRIBUTE(203)
- FORMAL-PARAMETER(208)
- LOCAL-BODY-ITEM(224)
- NESTED-ID-LIST(230) RETYPE-DECL(242)
- UNIQUE-RULE(263)
- ID-QUALIFIED(151) ID(148)
- ID-SIMPLE(152) ATTRIBUTE-ID(136) DEFINED-TYPE-ID(141)
- ENTITY-ID(145) FUNCTION-ID(147)
- MAP-ID(161) PROCEDURE-ID(168)
- RULE-ID(173) SCHEMA-ID(174)
- ID-LIST(214) SUPER-TYPE-FACTOR(258)
- IF(57) IF-STATEMENT(215)
- IN(58) IN-OPERATOR(153)
- IN-OPERATOR(153) RELATIONAL-OPERATOR-EXTENDED(171)
- INCLUDE(59) INCLUDE-DIRECTIVE(216)
- RELATIONAL-OPERATOR(170)
- OR-OPERATOR(166)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
- RELATIONAL-OPERATOR(170)
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LETTER</td>
<td>156</td>
</tr>
<tr>
<td>LIKE</td>
<td>72</td>
</tr>
<tr>
<td>LIKE-OPERATOR</td>
<td>157</td>
</tr>
<tr>
<td>LIMIT-SPEC</td>
<td>222</td>
</tr>
<tr>
<td>LIST</td>
<td>73</td>
</tr>
<tr>
<td>LIST-TYPE</td>
<td>223</td>
</tr>
<tr>
<td>LITERAL</td>
<td>158</td>
</tr>
<tr>
<td>LOBOUND</td>
<td>74</td>
</tr>
<tr>
<td>LOCAL</td>
<td>75</td>
</tr>
<tr>
<td>LOCAL-BODY-ITEM</td>
<td>224</td>
</tr>
<tr>
<td>LOCAL-CAN-BE</td>
<td>225</td>
</tr>
<tr>
<td>LOCAL-DECL</td>
<td>226</td>
</tr>
<tr>
<td>LOCAL-ID</td>
<td>159</td>
</tr>
<tr>
<td>LOG</td>
<td>76</td>
</tr>
<tr>
<td>LOG10</td>
<td>77</td>
</tr>
<tr>
<td>LOG2</td>
<td>78</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>79</td>
</tr>
<tr>
<td>LOGICAL-LITERAL</td>
<td>160</td>
</tr>
<tr>
<td>LOGICAL-TYPE</td>
<td>227</td>
</tr>
<tr>
<td>MAP</td>
<td>80</td>
</tr>
<tr>
<td>MAP-DECL</td>
<td>228</td>
</tr>
<tr>
<td>MAP-ID</td>
<td>161</td>
</tr>
<tr>
<td>MAP-TYPE</td>
<td>229</td>
</tr>
<tr>
<td>MINUS</td>
<td>81</td>
</tr>
<tr>
<td>MINUS-OPERATOR</td>
<td>162</td>
</tr>
<tr>
<td>MOD</td>
<td>82</td>
</tr>
<tr>
<td>MODULO-OPERATOR</td>
<td>163</td>
</tr>
<tr>
<td>MULTIPLICATION-LIKE-OPERATIONS</td>
<td>164</td>
</tr>
<tr>
<td>MULTIPLY-OPERATOR</td>
<td>83</td>
</tr>
<tr>
<td>NESTED-ID-LIST</td>
<td>230</td>
</tr>
<tr>
<td>NOT</td>
<td>84</td>
</tr>
<tr>
<td>NOT-EQUAL-OPERATOR</td>
<td>85</td>
</tr>
<tr>
<td>NOT-OPERATOR</td>
<td>165</td>
</tr>
<tr>
<td>NULL</td>
<td>86</td>
</tr>
<tr>
<td>NULLIFY</td>
<td>87</td>
</tr>
<tr>
<td>NUMBER</td>
<td>88</td>
</tr>
<tr>
<td>NUMBER-TYPE</td>
<td>231</td>
</tr>
<tr>
<td>ODD</td>
<td>89</td>
</tr>
<tr>
<td>OF</td>
<td>90</td>
</tr>
<tr>
<td>ID-CHAR</td>
<td>149</td>
</tr>
<tr>
<td>ID-SIMPLE</td>
<td>152</td>
</tr>
<tr>
<td>LIKE-OPERATOR</td>
<td>157</td>
</tr>
<tr>
<td>RELATIONAL-OPERATOR-EXTENDED</td>
<td>171</td>
</tr>
<tr>
<td>CARDINALITY-OPERATOR</td>
<td>139</td>
</tr>
<tr>
<td>AGGREGATE-TYPE</td>
<td>180</td>
</tr>
<tr>
<td>LIST-TYPE</td>
<td>223</td>
</tr>
<tr>
<td>SET-TYPE</td>
<td>247</td>
</tr>
<tr>
<td>GENERAL-AGGREGATION</td>
<td>212</td>
</tr>
<tr>
<td>LIST-TYPE</td>
<td>223</td>
</tr>
<tr>
<td>AGGREGATION-TYPE</td>
<td>181</td>
</tr>
<tr>
<td>SIMPLE-FACTOR</td>
<td>249</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>LOCAL-DECL</td>
<td>226</td>
</tr>
<tr>
<td>LOCAL-BODY-ITEM</td>
<td>224</td>
</tr>
<tr>
<td>LOCAL-CAN-BE</td>
<td>225</td>
</tr>
<tr>
<td>LOCAL-DECL</td>
<td>226</td>
</tr>
<tr>
<td>LOCAL-ID</td>
<td>159</td>
</tr>
<tr>
<td>INCREMENT-CONTROL</td>
<td>217</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>LOG2</td>
<td>78</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>79</td>
</tr>
<tr>
<td>LOGICAL-LITERAL</td>
<td>160</td>
</tr>
<tr>
<td>LOGICAL-TYPE</td>
<td>227</td>
</tr>
<tr>
<td>BASE-TYPE</td>
<td>186</td>
</tr>
<tr>
<td>MAP-DECL</td>
<td>228</td>
</tr>
<tr>
<td>BLOCK-MEMBER</td>
<td>187</td>
</tr>
<tr>
<td>MAP-ID</td>
<td>161</td>
</tr>
<tr>
<td>MAP-TYPE</td>
<td>229</td>
</tr>
<tr>
<td>ATTRIBUTE-CAN-BE</td>
<td>185</td>
</tr>
<tr>
<td>PARAMETER-CAN-BE</td>
<td>232</td>
</tr>
<tr>
<td>RESULT-CAN-BE</td>
<td>240</td>
</tr>
<tr>
<td>MINUS-OPERATOR</td>
<td>162</td>
</tr>
<tr>
<td>SIGN</td>
<td>175</td>
</tr>
<tr>
<td>ADDITION-LIKE-OPERATIONS</td>
<td>134</td>
</tr>
<tr>
<td>MODULO-OPERATOR</td>
<td>163</td>
</tr>
<tr>
<td>MULTIPLICATION-LIKE-OPERATIONS</td>
<td>164</td>
</tr>
<tr>
<td>EXPORT-DIRECTIVE</td>
<td>205</td>
</tr>
<tr>
<td>RULE-DECL</td>
<td>243</td>
</tr>
<tr>
<td>SELECT-TYPE</td>
<td>245</td>
</tr>
<tr>
<td>SUBTYPE-DECL</td>
<td>255</td>
</tr>
<tr>
<td>USES-DIRECTIVE</td>
<td>265</td>
</tr>
<tr>
<td>NOT-OPERATOR</td>
<td>165</td>
</tr>
<tr>
<td>RELATIONAL-OPERATOR</td>
<td>170</td>
</tr>
<tr>
<td>SIMPLE-FACTOR</td>
<td>249</td>
</tr>
<tr>
<td>SUPERTYPE-FACTOR</td>
<td>258</td>
</tr>
<tr>
<td>BUILT-IN-PROCEDURE-NAME</td>
<td>138</td>
</tr>
<tr>
<td>NUMBER-TYPE</td>
<td>231</td>
</tr>
<tr>
<td>BASE-TYPE</td>
<td>186</td>
</tr>
<tr>
<td>TYPE-CAN-BE</td>
<td>260</td>
</tr>
<tr>
<td>BUILT-IN-FUNCTION-NAME</td>
<td>137</td>
</tr>
<tr>
<td>AGGREGATE-TYPE</td>
<td>180</td>
</tr>
<tr>
<td>ARRAY-TYPE</td>
<td>182</td>
</tr>
<tr>
<td>CASE-STATEMENT</td>
<td>191</td>
</tr>
<tr>
<td>ENUMERATION-TYPE</td>
<td>201</td>
</tr>
<tr>
<td>GENERAL-AGGREGATION</td>
<td>212</td>
</tr>
<tr>
<td>LIST-TYPE</td>
<td>223</td>
</tr>
<tr>
<td>SET-TYPE</td>
<td>247</td>
</tr>
<tr>
<td>SUBTYPE-DECL</td>
<td>255</td>
</tr>
<tr>
<td>Optional</td>
<td>91</td>
</tr>
<tr>
<td>OR-Operator</td>
<td>166</td>
</tr>
<tr>
<td>Supertype-Decl</td>
<td>256</td>
</tr>
<tr>
<td>Otherwise</td>
<td>92</td>
</tr>
<tr>
<td>Parameter-can-be</td>
<td>232</td>
</tr>
<tr>
<td>General-aggregation(212)</td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td>93</td>
</tr>
<tr>
<td>Plus</td>
<td>94</td>
</tr>
<tr>
<td>Plus-operator</td>
<td>167</td>
</tr>
<tr>
<td>Precision-spec</td>
<td>233</td>
</tr>
<tr>
<td>String-type(252)</td>
<td></td>
</tr>
<tr>
<td>Procedure</td>
<td>95</td>
</tr>
<tr>
<td>Procedure-call-statement</td>
<td>234</td>
</tr>
<tr>
<td>Procedure-Decl</td>
<td>235</td>
</tr>
<tr>
<td>Procedure-ID</td>
<td>168</td>
</tr>
<tr>
<td>Procedure-decl(235)</td>
<td></td>
</tr>
<tr>
<td>Qualification</td>
<td>236</td>
</tr>
<tr>
<td>Real</td>
<td>96</td>
</tr>
<tr>
<td>Real-divide-operator</td>
<td>97</td>
</tr>
<tr>
<td>Real-literal</td>
<td>169</td>
</tr>
<tr>
<td>Real-type</td>
<td>237</td>
</tr>
<tr>
<td>Relational-operator</td>
<td>170</td>
</tr>
<tr>
<td>INTERVAL(221)</td>
<td></td>
</tr>
<tr>
<td>Relational-operator-extended</td>
<td>171</td>
</tr>
<tr>
<td>Repeat</td>
<td>98</td>
</tr>
<tr>
<td>Repeat-control</td>
<td>238</td>
</tr>
<tr>
<td>Repeat-statement</td>
<td>239</td>
</tr>
<tr>
<td>Result-can-be</td>
<td>240</td>
</tr>
<tr>
<td>Return</td>
<td>99</td>
</tr>
<tr>
<td>Return-statement</td>
<td>241</td>
</tr>
<tr>
<td>Retype</td>
<td>100</td>
</tr>
<tr>
<td>Retype-decl</td>
<td>242</td>
</tr>
<tr>
<td>Right-bracket</td>
<td>101</td>
</tr>
<tr>
<td>SET-CONSTRUCT(246) SUBSCRIPT(253)</td>
<td></td>
</tr>
<tr>
<td>INTERVAL(221)</td>
<td></td>
</tr>
<tr>
<td>Right-curl</td>
<td>102</td>
</tr>
<tr>
<td>Assume-directive(184)</td>
<td></td>
</tr>
<tr>
<td>Formal-parameter-list(209)</td>
<td></td>
</tr>
<tr>
<td>Nested-id-list(230) Precision-spec(233)</td>
<td></td>
</tr>
<tr>
<td>Return-statement(241)</td>
<td></td>
</tr>
<tr>
<td>Simple-factor(249) Supertype-decl(256)</td>
<td></td>
</tr>
<tr>
<td>Supertype-factor(258)</td>
<td></td>
</tr>
<tr>
<td>Rule</td>
<td>104</td>
</tr>
<tr>
<td>Rule-decl</td>
<td>243</td>
</tr>
<tr>
<td>Rule-id</td>
<td>173</td>
</tr>
<tr>
<td>Schema</td>
<td>105</td>
</tr>
<tr>
<td>Schema-decl</td>
<td>244</td>
</tr>
<tr>
<td>TERM</td>
<td>PAGE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
</tr>
<tr>
<td>SCHDU</td>
<td>174</td>
</tr>
<tr>
<td>SELECT</td>
<td>106</td>
</tr>
<tr>
<td>SELECT-TYPE</td>
<td>245</td>
</tr>
<tr>
<td>SEMICOLON</td>
<td>107</td>
</tr>
<tr>
<td>ASSUME-DIRECTIVE</td>
<td>184</td>
</tr>
<tr>
<td>SCHEMA-DECL</td>
<td>244</td>
</tr>
<tr>
<td>SELECT-TYPE</td>
<td>245</td>
</tr>
<tr>
<td>TYPE-CAN-BE</td>
<td>260</td>
</tr>
<tr>
<td>ASSIGNMENT-STATEMENT</td>
<td>183</td>
</tr>
<tr>
<td>ASSUME-DIRECTIVE</td>
<td>184</td>
</tr>
<tr>
<td>CASE-STATEMENT</td>
<td>191</td>
</tr>
<tr>
<td>COMPOUND-STATEMENT</td>
<td>192</td>
</tr>
<tr>
<td>DEFINE-DIRECTIVE</td>
<td>193</td>
</tr>
<tr>
<td>DERIVED-ATTRIBUTE</td>
<td>196</td>
</tr>
<tr>
<td>DUMMY-STATEMENT</td>
<td>198</td>
</tr>
<tr>
<td>ENTITY-DECL</td>
<td>199</td>
</tr>
<tr>
<td>ESCAPE-STATEMENT</td>
<td>202</td>
</tr>
<tr>
<td>EXPPLICIT-ATTRIBUTE</td>
<td>203</td>
</tr>
<tr>
<td>EXPORT-DIRECTIVE</td>
<td>205</td>
</tr>
<tr>
<td>FORMAL-PARAMETER-LIST</td>
<td>209</td>
</tr>
<tr>
<td>FUNCTION-DECL</td>
<td>211</td>
</tr>
<tr>
<td>INCLUDE-DIRECTIVE</td>
<td>216</td>
</tr>
<tr>
<td>LOCAL-BODY-ITEM</td>
<td>224</td>
</tr>
<tr>
<td>LOCAL-DECL</td>
<td>226</td>
</tr>
<tr>
<td>MAP-DECL</td>
<td>228</td>
</tr>
<tr>
<td>PROCEDURE-CALL-STATEMENT</td>
<td>234</td>
</tr>
<tr>
<td>PROCEDURE-DECL</td>
<td>235</td>
</tr>
<tr>
<td>REPEAT-STATEMENT</td>
<td>239</td>
</tr>
<tr>
<td>RETURN-STATEMENT</td>
<td>241</td>
</tr>
<tr>
<td>RETYPE-DECL</td>
<td>242</td>
</tr>
<tr>
<td>RULE-DECL</td>
<td>243</td>
</tr>
<tr>
<td>SCHEMA-DECL</td>
<td>244</td>
</tr>
<tr>
<td>SKIP-STATEMENT</td>
<td>250</td>
</tr>
<tr>
<td>STATEMENT</td>
<td>251</td>
</tr>
<tr>
<td>STRING</td>
<td>108</td>
</tr>
<tr>
<td>SET-CONSTRUCT</td>
<td>246</td>
</tr>
<tr>
<td>SET-TYPE</td>
<td>247</td>
</tr>
<tr>
<td>SIGN</td>
<td>175</td>
</tr>
<tr>
<td>SIMPLE-EXPRESSION</td>
<td>248</td>
</tr>
<tr>
<td>SIMPLE-FACTOR</td>
<td>249</td>
</tr>
<tr>
<td>SKIP-STATEMENT</td>
<td>250</td>
</tr>
<tr>
<td>SQRT</td>
<td>251</td>
</tr>
<tr>
<td>STATEMENT</td>
<td>112</td>
</tr>
<tr>
<td>STRING</td>
<td>113</td>
</tr>
<tr>
<td>STRING-LITERAL</td>
<td>176</td>
</tr>
<tr>
<td>STRING-TYPE</td>
<td>252</td>
</tr>
<tr>
<td>SUBSCRIPT</td>
<td>253</td>
</tr>
<tr>
<td>SUBSUPER-DECL</td>
<td>254</td>
</tr>
<tr>
<td>SUBTYPE</td>
<td>114</td>
</tr>
<tr>
<td>SUBTYPE-DECL</td>
<td>255</td>
</tr>
<tr>
<td>SUPERTYPE</td>
<td>115</td>
</tr>
</tbody>
</table>
bf RULE DEFINITION

1 ABS = 'ABS'
2 ACOS = 'ACOS'
3 AGGREGATE = 'AGGREGATE'
4 AND = 'AND'
5 ARRAY = 'ARRAY'
6 AS = 'AS'
7 ASIN = 'ASIN'
8 ASSIGNMENT-OPERATOR = ':='
9 ASSUME = 'ASSUME'
10 ATAN = 'ATAN'
BEGIN = 'BEGIN' .
BY = 'BY' .
CASE = 'CASE' .
COLON = ':' .
CONMA = ',' .
CONCATENATION-OPERATOR = '||' .
COS = 'COS' .
DEFINE = 'DEFINE' .
DERIVE = 'DERIVE' .
DIV = 'DIV' .

DOT = '.' .
DYNAMIC = 'DYNAMIC' .
ELSE = 'ELSE' .
END = 'END' .
END-CASE = 'END_CASE' .
END-ENTITY = 'END_ENTITY' .
END-FUNCTION = 'END_FUNCTION' .
END-IF = 'END_IF' .
END-LOCAL = 'END_LOCAL' .
END-MAP = 'END_MAP' .

END-PROCEDURE = 'END_PROCEDURE' .
END-REPEAT = 'END_REPEAT' .
END-RULE = 'END_RULE' .
END SCHEMA = 'END_SCHEMA' .
END-TYPE = 'END_TYPE' .
ENTITY = 'ENTITY' .
ENUMERATION = 'ENUMERATION' .
EQUAL = '=' .
ESCAPE = 'ESCAPE' .
EVERYTHING = 'EVERYTHING' .

EXCEPT = 'EXCEPT' .
EXCLUSIVE-OR-OPERATOR = 'XOR' .
EXISTS = 'EXISTS' .
EXP = 'EXP' .
EXPONENTIATION-OPERATOR = '**' .
EXPORT = 'EXPORT' .
EXTERNAL = 'EXTERNAL' .
FALSE = 'FALSE' .
40 FOR = 'FOR'
40 FORMAT = 'FORMAT'

51 FROM = 'FROM'
52 FUNCTION = 'FUNCTION'
53 GENERIC = 'GENERIC'
54 GREATER-EQUAL-OPERATOR = '>='
55 GREATER-THEAN-OPERATOR = '>
56 HIBOUND = 'HIBOUND'
57 IF = 'IF'
58 IN = 'IN'
59 INCLUDE = 'INCLUDE'
60 INCLUSIVE-OR-OPERATOR = 'OR'

61 INFINITY = '#'
62 INSERT = 'INSERT'
63 INSTANCE-EQUAL-OPERATOR = ':='
64 INSTANCE-NOTEQUAL-OPERATOR = ':<>:'
65 INTEGER = 'INTEGER'
66 INTERNAL = 'INTERNAL'
67 LEFT-BRACKET = '['
68 LEFT-CURL = '{'
69 LEFT-PAREN = '('
70 LESS-EQUAL-OPERATOR = '<='

71 LESS-THAN-OPERATOR = '<'
72 LIKE = 'LIKE'
73 LIST = 'LIST'
74 LOBOUND = 'LOBOUND'
75 LOCAL = 'LOCAL'
76 LOG = 'LOG'
77 LOG10 = 'LOG10'
78 LOG2 = 'LOG2'
79 LOGICAL = 'LOGICAL'
80 MAP = 'MAP'

81 MINUS = '-'
82 MOD = 'MOD'
83 MULTIPLY-OPERATOR = '*'
84 NOT = 'NOT'
85 NOT-EQUAL-OPERATOR = '<>'

82
86 NULL = 'NULL'.
87 NULLIFY = 'NULLIFY'.
88 NUMBER = 'NUMBER'.
89 ODD = 'ODD'.
90 OF = 'OF'.

91 OPTIONAL = 'OPTIONAL'.
92 OTHERWISE = 'OTHERWISE'.
93 PI = 'PI'.
94 PLUS = '+'.
95 PROCEDURE = 'PROCEDURE'.
96 REAL = 'REAL'.
97 REAL-DIVIDE-OPERATOR = '/'.
98 REPEAT = 'REPEAT'.
99 RETURN = 'RETURN'.
100 RETYPE = 'RETYPE'.

101 RIGHT-BRACKET = ']'.
102 RIGHT-CURL = ']').
103 RIGHT-PAREN = ')'.
104 RULE = 'RULE'.
105 SCHEMA = 'SCHEMA'.
106 SELECT = 'SELECT'.
107 SEMICOLON = ';'.
108 SET = 'SET'.
109 SIN = 'SIN'.
110 SIZEOF = 'SIZEOF'.

111 SKIP = 'SKIP'.
112 SQRT = 'SQRT'.
113 STRING = 'STRING'.
114 SUBTYPE = 'SUBTYPE'.
115 SUPERTYPE = 'SUPERTYPE'.
116 TAN = 'TAN'.
117 THEN = 'THEN'.
118 TO = 'TO'.
119 TRUE = 'TRUE'.
120 TYPE = 'TYPE'.

121 TYPEOF = 'TYPEOF'.
122 UNDERSCORE = '_'.

83
123 UNIQUE = 'UNIQUE' .
124 UNKNOWN = 'UNKNOWN' .
125 UNTIL = 'UNTIL' .
126 USES = 'USES' .
127 VALUE = 'VALUE' .
128 VAR = 'VAR' .
129 VARYING = 'VARYING' .
130 VIOLATION = 'VIOLATION' .
131 WHERE = 'WHERE' .
132 WHILE = 'WHILE' .
133 WITH = 'WITH' .
134 ADDITION-LIKE-OPERATIONS = PLUS-OPERATOR | MINUS-OPERATOR 
| OR-OPERATOR | CONCATENATION-OPERATOR .
135 AND-OPERATOR = AND .
136 ATTRIBUTE-ID = ID-SIMPLE .
137 BUILT-IN-FUNCTION-NAME = ABS | ACOS | ASIN | ATAN | COS | EXISTS 
| EXP | FORMAT | HIBOUND | LOBOUND | LOG | LOG10 | LOG2 | ODD 
| SIN | SIZEOF | SQRT | TAN | TYPEOF | VALUE .
138 BUILT-IN PROCEDURE NAME = INSERT | NULLIFY | VIOLATION .
139 CARDINALITY-OPERATOR = LIMIT-SPEC .
140 CONSTANT = FALSE | TRUE | UNKNOWN | PI .
141 DEFINED-TYPE-ID = ID-SIMPLE .
142 DIGIT = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .
143 DIVIDE-OPERATOR = INTEGER-DIVIDE-OPERATOR | REAL-DIVIDE-OPERATOR .
144 EMBEDDED-REMARK = '(*' { \A | \T | \N } ')' .
145 ENTITY-ID = ID-SIMPLE .
146 EQUAL-OPERATOR = EQUAL .
147 FUNCTION-ID = BUILT-IN-FUNCTION-NAME | ID-SIMPLE .
148 ID = ID-SIMPLE | ID-QUALIFIED .
149 ID-CHAR = LETTER | DIGIT | UNDERSCORE .
150 ID-INDEXED = ID { SUBSCRIPT } .
151 ID-QUALIFIED = ID-INDEXED { DOT ID-INDEXED } .
152 ID-SIMPLE = LETTER { ID-CHAR } .
153 IN-OPERATOR = IN .
154 INTEGER-DIVIDE-OPERATOR = DIV .
155 INTEGER-LITERAL = DIGIT { DIGIT } .
156 LETTER = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' 
| 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' 
| 'V' | 'W' | 'X' | 'Y' | 'Z' .
157 LIKE-OPERATOR = LIKE .
158 LITERAL = INTEGER-LITERAL | REAL-LITERAL | STRING-LITERAL
   | LOGICAL-LITERAL | CONSTANT .
159 LOCAL-ID = ID .
160 LOGICAL-LITERAL = TRUE | FALSE | UNKNOWN .

161 MAP-ID = ID-SIMPLE .
162 MINUS-OPERATOR = MINUS .
163 MODULO-OPERATOR = MOD .
164 MULTIPLICATION-LIKE-OPERATIONS = MULTIPLY-OPERATOR | DIVIDE-OPERATOR
   | MODULO-OPERATOR | AND-OPERATOR .
165 NOT-OPERATOR = NOT .
166 OR-OPERATOR = INCLUSIVE-OR-OPERATOR | EXCLUSIVE-OR-OPERATOR .
167 PLUS-OPERATOR = PLUS .
168 PROCEDURE-ID = BUILT-IN-PROCEDURE-NAME | ID-SIMPLE .
170 RELATIONAL-OPERATOR = LESS-THAN-OPERATOR | GREATER-THAN-OPERATOR
   | EQUAL-OPERATOR | LESS-EQUAL-OPERATOR | GREATER-EQUAL-OPERATOR
   | NOT-EQUAL-OPERATOR | INSTANCE-EQUAL-OPERATOR
   | INSTANCE-NOTEQUAL-OPERATOR .
171 RELATIONAL-OPERATOR-EXTENDED = RELATIONAL-OPERATOR | IN-OPERATOR
   | LIKE-OPERATOR | CARDINALITY-OPERATOR .
172 REMARK = EMBEDDED-REMARK | TAIL-REMARK .
173 RULE-ID = ID-SIMPLE .
174 SCHEMA-ID = ID-SIMPLE .
175 SIGN = PLUS | MINUS .
176 STRING-LITERAL = \Q { \A | ( \Q \Q ) } \Q .
177 TAIL-REMARK = '--' { \A | \T } \N .
178 ACTION-BODY = { BLOCK-MEMBER | LOCAL-DECL } ( STATEMENT ) .
179 ACTUAL-PARAMETER-LIST = LEFT-PAREN EXPRESSION { COMMA EXPRESSION } RIGHT-PAREN .

181 AGGREGATION-TYPE = AGGREGATE-TYPE | ARRAY-TYPE | LIST-TYPE | SET-TYPE .
182 ARRAY-TYPE = ARRAY INDEX-SPEC OF [ OPTIONAL ] [ UNIQUE ] ATTRIBUTE-CAN-BE .
183 ASSIGNMENT-STATEMENT = ID ASSIGNMENT-OPERATOR EXPRESSION SEMICOLON .
184 ASSUME-DIRECTIVE = ASSUME LEFT-PAREN SCHEMA-ID { COMMA SCHEMA-ID } RIGHT-PAREN SEMICOLON .
185 ATTRIBUTE-CAN-BE = AGGREGATION-TYPE | BASE-TYPE | DEFINED-TYPE
   | ENTITY-TYPE | MAP-TYPE .
186 BASE-TYPE = INTEGER-TYPE | LOGICAL-TYPE | NUMBER-TYPE | REAL-TYPE
   | STRING-TYPE .
187 BLOCK-MEMBER = ENTITY-DECL | MAP-DECL | FUNCTION-DECL | PROCEDURE-DECL
   | RULE-DECL | SCHEMA-DECL | TYPE-DECL | DIRECTIVE .

85
188 CASE-ACTION = EXPRESSION { COMMA EXPRESSION } COLON STATEMENT .
189 CASE-BLOCK = CASE-ACTION { CASE-ACTION } [ CASE-OTHERWISE ] .
190 CASE-OTHERWISE = OTHERWISE COLON STATEMENT .

191 CASE-STATEMENT = CASE EXPRESSION OF CASE-BLOCK END-CASE SEMICOLON .
192 COMPOUND-STATEMENT = BEGIN { STATEMENT } END SEMICOLON .
193 DEFINE-DIRECTIVE = DEFINE DEFINED-TYPE-ID STRING-LITERAL SEMICOLON .
194 DEFINED-TYPE = DEFINED-TYPE-ID .
195 DERIVED-ATTRIBUTE = ATTRIBUTE-ID COLON RESULT-CAN-BE INITIALIZER SEMICOLON .
196 DERIVED-ATTRIBUTE-DECL = DERIVE DERIVED-ATTRIBUTE { DERIVED-ATTRIBUTE } .
197 DIRECTIVE = ASSUME-DIRECTIVE | DEFINE-DIRECTIVE | EXPORT-DIRECTIVE |
INCLUDE-DIRECTIVE | USES-DIRECTIVE .
198 DUMMY-STATEMENT = SEMICOLON .
199 ENTITY-DECL = ENTITY ENTITY-ID [ SUBSUPER-DECL ] SEMICOLON |
[ EXPLICIT-ATTRIBUTE-DECL ] [ DERIVED-ATTRIBUTE-DECL ] [ UNIQUE-RULE ] |
200 ENTITY-TYPE = ENTITY-ID .

201 ENUMERATION-TYPE = ENUMERATION OF NESTED-ID-LIST .
202 ESCAPE-STATEMENT = ESCAPE SEMICOLON .
203 EXPLICIT-ATTRIBUTE = ID-LIST COLON [ OPTIONAL ] [ EXTERNAL | INTERNAL |
DYNAMIC ] ATTRIBUTE-CAN-BE SEMICOLON .
204 EXPLICIT-ATTRIBUTE-DECL = EXPLICIT-ATTRIBUTE { EXPLICIT-ATTRIBUTE } .
205 EXPORT-DIRECTIVE = EXPORT ( NESTED-ID-LIST | EVERYTHING | EVERYTHING |
EXCEPT NESTED-ID-LIST ) SEMICOLON .
206 EXPRESSION = SIMPLE-EXPRESSION { RELATIONAL-OPERATOR-EXTENDED |
SIMPLE-EXPRESSION } .
207 FACTOR = SIMPLE-FACTOR [ EXPONENTIATION-OPERATOR SIMPLE-FACTOR ] .
209 FORMAL-PARAMETER-LIST = LEFT-PAREN FORMAL-PARAMETER { SEMICOLON |
FORMAL-PARAMETER } RIGHT-PAREN .
210 FUNCTION-CALL = FUNCTION-ID [ ACTUAL-PARAMETER-LIST ] { QUALIFICATION } .

211 FUNCTION-DECL = FUNCTION FUNCTION-ID [ FORMAL-PARAMETER-LIST ] COLON |
RESULT-CAN-BE SEMICOLON ACTION-BODY END-FUNCTION SEMICOLON .
212 GENERAL-AGGREGATION = ( AGGREGATE | ARRAY | LIST | SET ) OF |
PARAMETER-CAN-BE .
213 GENERIC-TYPE = GENERIC .
214 ID-LIST = ID-SIMPLE { COMMA ID-SIMPLE } .
215 IF-STATEMENT = IF EXPRESSION THEN STATEMENT { STATEMENT } [ ELSE |
STATEMENT { STATEMENT } ] END-IF .
216 INCLUDE-DIRECTIVE = INCLUDE STRING-LITERAL SEMICOLON .
217 INCREMENT-CONTROL = LOCAL-ID ASSIGNMENT-OPERATOR EXPRESSION TO |
EXPRESSION [ BY EXPRESSION ] .
218 INDEX-SPEC = LEFT-BRACKET EXPRESSION COLON EXPRESSION RIGHT-BRACKET .
INITIALIZER = ASSIGNMENT-OPERATOR EXPRESSION.

INTEGER-TYPE = INTEGER [ PRECISION-SPEC ].

INTERVAL = LEFT-CURL EXPRESSION RELATIONAL-OPERATOR EXPRESSION RELATIONAL-OPERATOR EXPRESSION RIGHT-CURL.

LIMIT-SPEC = LEFT-BRACKET EXPRESSION COLON ( EXPRESSION | INFINITY ) RIGHT-BRACKET.

LIST-TYPE = LIST LIMIT-SPEC OF [ UNIQUE ] ATTRIBUTE-CAN-BE.

LOCAL-BODY-ITEM = ID-LIST COLON LOCAL-CAN-BE [ INITIALIZER ] SEMICOLON.

LOCAL-CAN-BE = AGGREGATION-TYPE | BASE-TYPE | DEFINED-TYPE | ENTITY-TYPE.

LOCAL-DECL = LOCAL { LOCAL-BODY-ITEM } END-LOCAL SEMICOLON.

LOCAL-TYPE = LOGICAL.

MAP-DECL = MAP MAP-ID FROM ENTITY-ID [ SUBSUPER-DECL ] SEMICOLON
          { RETYPE-DECL } [ UNIQUE-RULE ] [ WHERE-RULE ] END-MAP SEMICOLON.

MAP-TYPE = MAP-ID.

NESTED-ID-LIST = LEFT-PAREN ID-LIST RIGHT-PAREN.

NUMBER-TYPE = NUMBER.

PARAMETER-CAN-BE = GENERIC-TYPE | GENERAL-AGGREGATION | BASE-TYPE
                  | DEFINED-TYPE | ENTITY-TYPE | MAP-TYPE.

PRECISION-SPEC = LEFT-PAREN EXPRESSION RIGHT-PAREN.

PROCEDURE-CALL-STATEMENT = PROCEDURE-ID [ ACTUAL-PARAMETER-LIST ] SEMICOLON.

PROCEDURE-DECL = PROCEDURE PROCEDURE-ID [ FORMAL-PARAMETER-LIST ]
                SEMICOLON ACTION-BODY END-PROCEDURE SEMICOLON.

QUALIFICATION = { SUBSCRIPT } [ DOT ID ].

REAL-TYPE = REAL [ PRECISION-SPEC ].

REPEAT-CONTROL = INCREMENT-CONTROL | WHILE-CONTROL | UNTIL-CONTROL.

REPEAT-STATEMENT = REPEAT { REPEAT-CONTROL } SEMICOLON { STATEMENT }
                  END-REPEAT SEMICOLON.

RESULT-CAN-BE = BASE-TYPE | DEFINED-TYPE | ENTITY-TYPE | MAP-TYPE |
                GENERIC-TYPE.

RETURN-STATEMENT = RETURN [ LEFT-PAREN EXPRESSION RIGHT-PAREN ] SEMICOLON

RETTYPE-DECL = ID-LIST COLON RETYPE AS ATTRIBUTE-CAN-BE SEMICOLON.

RULE-DECL = RULE RULE-ID FOR NESTED-ID-LIST SEMICOLON ACTION-BODY
           END-RULE SEMICOLON.

SCHEMA-DECL = SCHEMA SCHEMA-ID SEMICOLON { BLOCK-MEMBER } END-SCHEMA
             SEMICOLON.

SELECT-TYPE = SELECT NESTED-ID-LIST.

SET-CONSTRUCT = LEFT-BRACKET [ EXPRESSION { COMMA EXPRESSION } ]
                RIGHT-BRACKET.

SET-TYPE = SET [ LIMIT-SPEC ] OF ATTRIBUTE-CAN-BE.

SIMPLE-EXPRESSION = [ SIGN ] TERM { ADDITION-LIKE-OPERATIONS TERM }.

SIMPLE-FACTOR = LITERAL | FUNCTION-CALL | ID | SET-CONSTRUCT |
LEFT-PAREN EXPRESSION RIGHT-PAREN | INTERVAL | NOT-OPERATOR SIMPLE-FACTOR.
250 SKIP-STATEMENT = SKIP SEMICOLON.

251 STATEMENT = ASSIGNMENT-STATEMENT | CASE-STATEMENT | COMPOUND-STATEMENT
| DUMMY-STATEMENT | ESCAPE-STATEMENT | IF-STATEMENT
| PROCEDURE-CALL-STATEMENT | REPEAT-STATEMENT | RETURN-STATEMENT
| SKIP-STATEMENT | WITH-STATEMENT.
252 STRING-TYPE = STRING [ PRECISION-SPEC ] [ VARYING ].
253 SUBSCRIPT = LEFT-BRACKET EXPRESSION RIGHT-BRACKET.
255 SUBTYPE-DECL = SUBTYPE OF NESTED-ID-LIST.
256 SUPERTYPE-DECL = SUPERTYPE OF LEFT-PAREN SUPERTYPE-EXPRESSION RIGHT-PAREN.
257 SUPERTYPE-EXPRESSION = SUPERTYPE-FACTOR { ( AND-OPERATOR | OR-OPERATOR )
SUPERTYPE-FACTOR }.
258 SUPERTYPE-FACTOR = ID-SIMPLE | NULL | LEFT-PAREN SUPERTYPE-EXPRESSION
RIGHT-PAREN.
259 TERM = FACTOR { MULTIPLICATION-LIKE-OPERATIONS FACTOR }.
260 TYPE-CAN-BE = AGGREGATION-TYPE | BASE-TYPE | DEFINED-TYPE
| ENUMERATION-TYPE | NUMBER-TYPE | SELECT-TYPE.

261 TYPE-DECL = TYPE TYPE-ITEM { TYPE-ITEM } END-TYPE SEMICOLON.
262 TYPE-ITEM = DEFINED-TYPE-ID EQUAL TYPE-CAN-BE SEMICOLON.
263 UNIQUE-RULE = UNIQUE { ID-LIST SEMICOLON }.
264 UNTIL-CONTROL = UNTIL EXPRESSION.
265 USES-DIRECTIVE = USES NESTED-ID-LIST SEMICOLON.
266 WHERE-RULE = WHERE { EXPRESSION SEMICOLON }.
267 WHILE-CONTROL = WHILE EXPRESSION.
268 WITH-STATEMENT = WITH ID STATEMENT.
A.13.2 EXPRESS Glossary

Attribute An essential property, trait, quality, or characteristic (of an entity).
BNF Baucus-Nauer Form. A method of defining the morphology of a grammar.
Constraint Same as rule.
Data The representation forms of information dealt with by information processing systems and users thereof.
Data Base Also database. See Information Base.
Entity An information unit that has a uniform meaning and behavior within a UoD.
Fact A fact is any assertion that can be proved (or is said) to be true for some UoD.
Grammar The system of rules that governs the use of a language; the morphology and semantics of a language.
IGES Initial Graphic Exchange Specification.
Information Any kind of knowledge about things, facts, concepts, etc. that is exchangeable among users. Data that has meaning.
Information Base A consistent collection of data that can be interpreted and operated on by the information system.
Information System The system of elements including the schema, information base, hardware devices, and the user that forms a predictable mechanism for keeping and manipulating information.
PDDI Product Data Definition Interface.
PDES Product Data Exchange Specification.
Rule A procedure that enforces a fact.
Schema A consistent collection of statements expressing the necessary propositions that hold for some UoD. A description of a UoD.
STEP STandard for the Exchange of Product data.
Token An element (word) of a grammar, which can be distinguished from all other tokens.
Type A particular representation form of data such as an integer or a character string.
Universe of Discourse (UoD) A Universe of Discourse is an area of interest that is of importance to an individual or a group of individuals.
User Anybody or anything that issues messages to the information system and receives messages in return.
A.13.3 Case Study One

This case study deals with rooms, doors, and warehousing of doors.

A door is either an interior door or an exterior door.

```
SCHEMA DOORS;
    ENTITY DOOR SUPERTYPE OF (INTERIOR_DOOR XOR EXTERIOR_DOOR);
    END_ENTITY;
    ENTITY INTERIOR_DOOR SUBTYPE OF (DOOR);
    END_ENTITY;
    ENTITY EXTERIOR_DOOR SUBTYPE OF (DOOR);
    END_ENTITY;
    ENTITY ROOM;
    END_ENTITY;
SCHEMA WHAREHOUSING;
END_SCHEMA;
SCHEMA BUILDING_DESIGN;
END_SCHEMA;
END_SCHEMA;
```
THE STEP FILE STRUCTURE

DOCUMENT NUMBER: 4.2.1
VERSION: 12.0

TITLE: THE STEP FILE STRUCTURE

ABSTRACT:
This document identifies the physical file structure for a STEP exchange file.

KEY WORDS:
Section, Data Type, Entity, Tokens, Exchange Format, Character Set, Medium, File Structure

DATE: September 1988
OWNER: Jeff Altemueller
PHONE NUMBER: (314) 234-5272
ISO REPRESENTATIVE: Jeff Altemueller
STATUS: Draft Paper
THE STEP FILE STRUCTURE

AUTHORS

Jeff Altemueller  MDAIS
Richard Cochran  Pratt & Whitney
Tony Day  Sikorsky Aircraft
Dennette A. Harrod, Jr.  COMPUTERVERSION
Janet Oakes  MDAIS
Ernst G. Schlechtendahl  Kernforschungszentrum
Chia-Hui Shih  SDRC
Guy Stil  Aerospatiale
Anna M. Whelan  MDAIS
Akio Yajima  Hitachi Ltd.
Jan Van Maanen  Rutherford Appleton Lab.

ACKNOWLEDGEMENTS

Bill Carr  Digital Equipment Corporation
William B. Gruttke  MDAIS
Edward Hong  GM/EDS
Geoffrey Parker  Intergraph Corporation
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Document Organization</td>
<td>1-1</td>
</tr>
<tr>
<td>2.0 EXCHANGE FORMAT OVERVIEW</td>
<td></td>
</tr>
<tr>
<td>2.1 Theoretical Basis</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2 General Structure</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2.1 File Format.</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2.2 Organization</td>
<td>2-1</td>
</tr>
<tr>
<td>2.2.3 High-Level File Syntax</td>
<td>2-1</td>
</tr>
<tr>
<td>2.3 Sections - Functional Descriptions</td>
<td>2-2</td>
</tr>
<tr>
<td>2.3.1 HEADER Section</td>
<td>2-2</td>
</tr>
<tr>
<td>2.3.2 DATA Section</td>
<td>2-2</td>
</tr>
<tr>
<td>3.0 FORMAL DEFINITIONS</td>
<td></td>
</tr>
<tr>
<td>3.1 Formal Notation</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1.1 WSN Notational Conventions</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2 Alphabet</td>
<td>3-3</td>
</tr>
<tr>
<td>3.2.1 Alphabet Subsets</td>
<td>3-3</td>
</tr>
<tr>
<td>3.3 Tokens Defined by Regular Expressions</td>
<td>3-4</td>
</tr>
<tr>
<td>3.4 WSN OF The STEP File Structure</td>
<td>3-6</td>
</tr>
<tr>
<td>4.0 TOKENS</td>
<td></td>
</tr>
<tr>
<td>4.1 Token Separators</td>
<td>4-1</td>
</tr>
<tr>
<td>4.2 Delimiters</td>
<td>4-1</td>
</tr>
<tr>
<td>4.3 Reserved Words</td>
<td>4-2</td>
</tr>
<tr>
<td>4.4 Keywords</td>
<td>4-3</td>
</tr>
<tr>
<td>4.5 Simple Data Types</td>
<td></td>
</tr>
<tr>
<td>4.5.1 Integer</td>
<td>4-3</td>
</tr>
<tr>
<td>4.5.2 Real</td>
<td>4-4</td>
</tr>
<tr>
<td>4.5.3 String</td>
<td>4-5</td>
</tr>
<tr>
<td>4.5.3.1 Alternate Character Sets</td>
<td>4-5</td>
</tr>
<tr>
<td>4.5.3.2 Maximum String Length</td>
<td>4-6</td>
</tr>
<tr>
<td>4.5.4 Entity Identifier</td>
<td>4-7</td>
</tr>
<tr>
<td>4.5.5 Entity Reference</td>
<td>4-7</td>
</tr>
<tr>
<td>4.5.6 Logical</td>
<td>4-8</td>
</tr>
<tr>
<td>5.0 STRUCTURED DATA TYPES</td>
<td></td>
</tr>
<tr>
<td>5.1 List</td>
<td>5-1</td>
</tr>
<tr>
<td>6.0 HEADER SECTION</td>
<td></td>
</tr>
<tr>
<td>6.1 HEADER Section Entities</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1.1 HEADER Section User Defined Entities</td>
<td>6-3</td>
</tr>
<tr>
<td>7.0 DATA SECTION</td>
<td></td>
</tr>
<tr>
<td>7.1 DATA Section Entities</td>
<td>7-1</td>
</tr>
<tr>
<td>7.1.1 Optional Values</td>
<td>7-2</td>
</tr>
<tr>
<td>7.2 SCOPE Structure</td>
<td>7-2</td>
</tr>
<tr>
<td>7.2.1 Syntax</td>
<td>7-2</td>
</tr>
<tr>
<td>7.2.2 Referencability</td>
<td>7-2</td>
</tr>
<tr>
<td>7.2.3 Behavior</td>
<td>7-2</td>
</tr>
<tr>
<td>7.3 External References</td>
<td>7-4</td>
</tr>
<tr>
<td>7.4 Index of Entity Aliases</td>
<td>7-4</td>
</tr>
<tr>
<td>7.5 DATA Section User Defined Entities</td>
<td>7-4</td>
</tr>
</tbody>
</table>
Table of Contents

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX A</td>
<td>Sample STEP Exchange Format File</td>
<td>A-1</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>The Graphical Alphabet</td>
<td>B-1</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>Entity Syntax on the Physical File</td>
<td>C-1</td>
</tr>
<tr>
<td>APPENDIX D</td>
<td>Physical Exchange Medium Formats</td>
<td>D-1</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The Standard for the Exchange of Product Data (STEP) is a neutral exchange medium capable of completely representing product definition data. It has been designed to meet both present and future requirements for exchanging such data between dissimilar CAD/CAM/CAE/CIM Systems.

1.1 DOCUMENT ORGANIZATION

This document specifies the STEP file structure format and the initial STEP entity set. The STEP File Structure document consists of seven chapters.

CHAPTER 1: INTRODUCTION - This section identifies the purpose of this document.

CHAPTER 2: EXCHANGE FORMAT OVERVIEW - This section provides a high level overview of the STEP file structure.

CHAPTER 3: FORMAL DEFINITIONS - This section presents a formal description of the exact syntax of the STEP file structure.

CHAPTER 4: TOKENS - This section presents an informal description of the parsing tokens as they should be recognized by STEP processors.

CHAPTER 5: STRUCTURED DATA TYPES - This section presents an informal description of the Structured Data Types.

CHAPTER 6: HEADER SECTION - This section presents an informal description of the HEADER Section.

CHAPTER 7: DATA SECTION - This section presents an informal description of the DATA Section.

Appendices have been included to provide additional information and relevant examples.
CHAPTER 2 - EXCHANGE FORMAT OVERVIEW

2.0  EXCHANGE FORMAT OVERVIEW

2.1  THEORETICAL BASIS

The STEP file structure is language-based and is described by an unambiguous, context-free grammar to facilitate parsing by software. The grammar is expressed in Wirth Syntax Notation and is included in Chapter 3.0. The majority of this document is devoted to the logical and syntactical description of a STEP exchange file. See Appendix D for detailed descriptions of the format of the file on various exchange media.

2.2  GENERAL STRUCTURE

2.2.1  File Format

The STEP exchange file is a sequential file. The information contained within the file is in free format. There is no column-dependent information.

2.2.2  Organization

The STEP exchange file is organized in a modular manner. The file consists of several sections. These sections consist of one or many entities. An entity consists of attributes. These file components are defined below.

<table>
<thead>
<tr>
<th>Sections</th>
<th>A section is a collection of data of the same functional category of information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>An entity is a collection of logically associated data.</td>
</tr>
<tr>
<td>Attribute</td>
<td>An attribute is a (data) fact about an entity. It is conceptually atomic (i.e., it represents a single fact) although it may be arbitrarily complex in structure. An entity is defined or described by its attributes.</td>
</tr>
</tbody>
</table>

2.2.3  High Level File Syntax

The STEP file is begun by "STEP;" and is terminated by "ENDSTEP;". Following the "STEP;", the HEADER and DATA sections will be added. These sections are described in subsections 2.3.1 and 2.3.2 of this document.

Additional sections may be added to the STEP file structure in the future in response to technological advances or industry requirements. Therefore, in order to preserve future compatibility, postprocessors should be prepared to encounter and ignore any information found before, between, or following file sections.
2.3 SECTIONS - FUNCTIONAL DESCRIPTIONS

Collections of data serving similar functions in the exchange are grouped into sections. There will be two sections in the STEP file structure: the HEADER section and the DATA section. Each section begins with a keyword and each section keyword is followed by a semicolon ";". The keywords for the two sections are listed below.

<table>
<thead>
<tr>
<th>Section</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEADER</td>
<td>HEADER</td>
</tr>
<tr>
<td>DATA</td>
<td>DATA</td>
</tr>
</tbody>
</table>

2.3.1 HEADER Section

The HEADER section provides information such as author and creation date, which is applicable to the entire exchange format file. The section is begun by "HEADER;" and is terminated by "ENDSEC;". This section must appear exactly once in the file and must be first in the file. For more information concerning the HEADER section, refer to Chapter 6.0 of this document.

2.3.2 DATA Section

The DATA section is used to define the product data to be transferred. It is a collection of occurrences of the standard types defined in the Conceptual Schema. The DATA section is begun by "DATA;" and is followed by DATA section entities. The DATA section is terminated by "ENDSEC;". This section must appear exactly once in the file. For more information concerning the DATA section, refer to Chapter 7.0 of this document.
CHAPTER 3 - FORMAL DEFINITIONS

3.0 FORMAL DEFINITIONS

3.1 FORMAL NOTATION

The STEP exchange format file may be considered as a continuous stream of characters from the basic alphabet. These can be collected into recognizable strings of characters called tokens.

The neutral file can therefore be considered as a sequence of tokens. The tokens can be formally defined by a regular set R(G) over the graphical alphabet G. STEP tokens are discussed in detail in Chapter 3.3.

3.1.1 WSN NOTATIONAL CONVENTIONS

The syntax of the STEP exchange file is defined in Wirth Syntax Notation (WSN). WSN was first described in a letter by Niklaus Wirth, which was published in Communications of the ACM, 20:11 (Nov 77), 822-823. The additional notational conventions are given below. Table 3-1 presents WSN defined in itself.

1. A string in uppercase (capital letters) is an element of the language; the string is the name of the element. In syntax diagrams, this corresponds to anything in a box with square corners. (For convenience, lowercase is used for undefined identifiers and commentary.)

2. Any string enclosed in double quotation marks (" ) is literally what is contained within the quotation marks. In syntax diagrams, this corresponds to anything in a box with curved corners. The one exception to this rule occurs when one wishes to specify a double quotation mark within a literal. In order to accomplish this, the double quotation mark is immediately repeated one time.

   (e.g., " " " is interpreted as " )

3. The equals sign (=) indicates a production. The element on the left is defined to be the combination of the elements on the right. Any spaces appearing between the elements of a production are meaningless unless they appear within a quoted literal. A production is terminated by a period ( ).
4. The semantics of the enclosing braces are defined below.

**CURLY BRACKETS** -- "{}" indicates zero or more repetitions

**SQUARE BRACKETS** -- "[]" indicate optional parameters

**VERTICAL BAR** -- "|" indicates the logical OR

**PARENTHESES** -- "(" and ")" indicate priority operations. In particular, when they enclose elements separated by vertical bars, one of the elements is to be chosen in conjunction with any other operation.

* e.g. \( A(B|C|D) = AB|AC|AD \)

<table>
<thead>
<tr>
<th>Table 3-1 Wirth Syntax Notation (WSN)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong> = ( PRODUCTION ).</td>
</tr>
<tr>
<td><strong>PRODUCTION</strong> = IDENTIFIER &quot;=&quot; EXPRESSION &quot;.&quot;.</td>
</tr>
<tr>
<td><strong>EXPRESSION</strong> = TERM ( &quot;</td>
</tr>
<tr>
<td><strong>TERM</strong> = FACTOR ( FACTOR ).</td>
</tr>
<tr>
<td><strong>FACTOR</strong> = IDENTIFIER</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>IDENTIFIER</strong> = letter ( letter ).</td>
</tr>
<tr>
<td><strong>LITERAL</strong> = &quot;&quot;&quot;&quot; character ( character ) &quot;&quot;&quot;&quot;.</td>
</tr>
</tbody>
</table>

3-2
3.2 ALPHABET DEFINITION

The alphabet of the STEP neutral file language is defined as a set of bytes with integer values ranging from 32 to 126. The standard "ISO 6937/1 1983(E)" defines the correspondence of the neutral file alphabet into graphical representation on paper and terminals. This function is limited by the capability of the hardware to accommodate such transformation from code to graphics.

The correspondence of the neutral file alphabet into graphical representation is presented in Appendix B - The Graphical Alphabet within this document. To define the syntax of the neutral file language, only the graphical representation is used, where G(k) denotes the graphical representation of the byte with the decimal value "k". The international monetary symbol cannot be printed by the text system that generates this document. In its place, a dollar sign will appear.

3.2.1 ALPHABET SUBSETS

Table 3-2 Alphabet Subsets

| SPACE     | = " ". |
| DIGIT     | = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
| LOWER     | = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
|           | = "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" |
|           | = "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" |
| UPPPER    | = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
|           | = "I" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" |
|           | = "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" |
| SPECIAL   | = ";" | "," | ";" | ";" | ";" | ";" | ";" | ";" |
|           | = ";" | ";" | ";" | ";" | ";" | ";" | ";" | ";" |
| ALPHABET  | = SPACE | DIGIT | LOWER | UPPPER | SPECIAL. |
3.3 TOKENS DEFINED BY REGULAR EXPRESSIONS

The following regular expressions in Table 3-3 will refer to those regular expressions defined previously in Table 3-2 Alphabet Subsets. The tokens that are intended to be recognized during lexical analysis are given below in the TOKEN definition. Please note that this TOKEN definition has been supplied only as a convenience to the reader, it will not be referenced by the WSN description of the file.

```
TOKEN = DELIMITER|RESERVED WORD|KEYWORD|INTEGER|REAL|STRING|COMMENT
        | ENTITY_IDENTIFIER|ENUMERATION|ENTITY
```

Table 3-3 Token Definition (1 of 2)

<table>
<thead>
<tr>
<th>COMMENT</th>
<th>= &quot;/*&quot; any_sequence_except_*<em>followed_by</em> &quot;*/&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELIMITER</td>
<td>= &quot;(&quot;</td>
</tr>
<tr>
<td>KEYWORD</td>
<td>= UPPER(UPPER</td>
</tr>
<tr>
<td>SIGN</td>
<td>= &quot;+&quot;</td>
</tr>
<tr>
<td>INTEGER</td>
<td>= [ SIGN ] DIGIT ( DIGIT ).</td>
</tr>
<tr>
<td>REAL</td>
<td>= [ SIGN ] DIGIT ( DIGIT ) &quot;.&quot; ( DIGIT ) [ &quot;E&quot; [ SIGN ] DIGIT ( DIGIT ) ].</td>
</tr>
<tr>
<td>NON_Q_CHAR</td>
<td>= any_character_from_the_alphabet_except_the_single_apostrophe.</td>
</tr>
<tr>
<td>QUOTE_CHAR</td>
<td>= &quot;'/'.&quot;</td>
</tr>
<tr>
<td>ESC_CHAR</td>
<td>= &quot;&quot;:&quot;:&quot;:&quot;.</td>
</tr>
<tr>
<td>ESC_ITEM</td>
<td>= &quot;ISO6937&quot;</td>
</tr>
<tr>
<td>ESC_SEQ</td>
<td>= &quot;&quot;: [ ESC_ITEM ( &quot;,&quot; ESC_ITEM ) ] &quot;:&quot;.</td>
</tr>
<tr>
<td>STRING</td>
<td>= &quot;&quot;( NON_Q_CHAR</td>
</tr>
</tbody>
</table>

3-4
Table 3-3 Token Definition (2 of 2)

<table>
<thead>
<tr>
<th>Token Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTITY_NAME</td>
<td>= DIGIT [ DIGIT [ DIGIT [ DIGIT [ DIGIT [ DIGIT</td>
</tr>
<tr>
<td>ENTITY_IDENTIFIER</td>
<td>= &quot;@&quot; ENTITY_NAME.</td>
</tr>
<tr>
<td>ENTITY_REFERENCE</td>
<td>= &quot;#&quot; ENTITY_NAME.</td>
</tr>
<tr>
<td>ENUMERATION</td>
<td>= &quot;.&quot; UPPER(UPPER(DIGIT) &quot;.&quot;.</td>
</tr>
</tbody>
</table>
3.4 WSN OF THE STEP FILE STRUCTURE

Table 3-4 identifies the WSN for the syntax of a STEP file. This WSN does not take into account any particular conceptual schema, but represents the physical structure for all possible conceptual schemata.

This syntax includes the tokens defined in section 3.3 of this document and corresponds to the syntax diagrams in Appendix C of this document.

Table 3-4 WSN of the STEP File Structure

<table>
<thead>
<tr>
<th>EXCHANGE_FILE</th>
<th>&quot;STEP;&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEADER_SECTION</td>
<td>DATA_SECTION &quot;ENDSTEP;&quot;.</td>
</tr>
<tr>
<td>HEADER_SECTION</td>
<td>&quot;HEADER;FILE_IDENTIFICATION(&quot; STRING &quot;,&quot; STRING &quot;,&quot; STRING &quot;,&quot; STRING &quot;,&quot; STRING &quot;,&quot; STRING &quot;);FILE_DESCRIPTION(&quot; STRING &quot;);IMP_LEVEL(&quot; STRING &quot;);&quot; HEADER_ENTITY_LIST &quot;ENDSEC;&quot;.</td>
</tr>
<tr>
<td>HEADER_ENTITY_LIST</td>
<td>HEADER_ENTITY</td>
</tr>
<tr>
<td>HEADER_ENTITY</td>
<td>KEYWORD &quot;(&quot; PARAMETER_LIST &quot;)&quot; &quot;;&quot;.</td>
</tr>
<tr>
<td>PARAMETER_LIST</td>
<td>PARAMETER</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DATA_SECTION</td>
<td>&quot;DATA;&quot; ENTITY_OCCUR_LIST &quot;ENDSEC;&quot;.</td>
</tr>
<tr>
<td>ENTITY_OCCUR_LIST</td>
<td>ENTITY_OCCURRENCE</td>
</tr>
<tr>
<td>ENTITY_OCCURRENCE</td>
<td>ENTITY_IDENTIFIER &quot;=&quot; EMBEDDED_ENTITY &quot;;&quot;.</td>
</tr>
<tr>
<td>EMBEDDED_ENTITY</td>
<td>[KEYWORD [&quot; SCOPE&quot; ENTITY_OCCUR_LIST &quot;ENDSCOPE&quot; [ENTITY_EXPORT_LIST]]] &quot;(&quot; PARAMETER_LIST &quot;).</td>
</tr>
<tr>
<td>ENTITY_EXPORT_LIST</td>
<td>&quot;/&quot; ENTITY_REFERENCE &quot;,&quot; ENTITY_REFERENCE) &quot;/&quot;.</td>
</tr>
</tbody>
</table>
CHAPTER 4 - TOKENS

4.0 TOKENS

A STEP file can be represented by a sequence of tokens. In the STEP file structure, a token is defined to be a delimiter, reserved word, keyword, or simple data type. The definition of the tokens in regular expression form can be found in Chapter 3.0 of this document. Syntax diagrams for the simple data types can be found in Appendix C of this document.

4.1 TOKEN SEPARATORS

A token separator is a character or a string of characters that can appear between any two tokens. Any number or combination of token separators may appear together between any two tokens. Token separators are insignificant and should be ignored during lexical analysis. Token separators may not appear within integers, real numbers, lists, entity references, entity identifiers, reserved words, keywords, or delimiters.

Blanks and comments are token separators. The blank is the space character. A comment is a slash followed by an asterisk "/*" followed by any number of characters, and terminated by an asterisk immediately followed by a slash "*/". Any characters appearing inside a comment are meaningless to the file structure. These characters are intended to be read by humans.

4.2 DELIMITERS

Delimiters are special characters or combinations of special characters that serve to separate, terminate, or otherwise denote meaningful collections of data. Note that some reserved words function as delimiters. Delimiters are listed in Table 4-1
### Table 4-1 Delimiters

<table>
<thead>
<tr>
<th>CHARACTER(S)</th>
<th>DELIMITER</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOPE</td>
<td>reserved word</td>
<td>Begins a scope section</td>
</tr>
<tr>
<td>ENDSCOPE</td>
<td>reserved word</td>
<td>Terminates a scope section</td>
</tr>
<tr>
<td>,</td>
<td>comma</td>
<td>Separates fields</td>
</tr>
<tr>
<td>;</td>
<td>semicolon</td>
<td>Terminates an entity occurrence</td>
</tr>
<tr>
<td>=</td>
<td>equal sign</td>
<td>Separates identifier from entity type keyword</td>
</tr>
<tr>
<td>()</td>
<td>parenthesis</td>
<td>Encloses a list, set, array or parameter list</td>
</tr>
<tr>
<td>ENDSEC;</td>
<td>reserved word</td>
<td>Terminates a section</td>
</tr>
<tr>
<td>ENDSOTP;</td>
<td>reserved word</td>
<td>Terminates a STEP file</td>
</tr>
<tr>
<td>/*</td>
<td>slash-asterisk</td>
<td>Begins a comment</td>
</tr>
<tr>
<td>*/</td>
<td>asterisk-slash</td>
<td>Terminates a comment</td>
</tr>
<tr>
<td>/</td>
<td>slash</td>
<td>Encloses an entity export list</td>
</tr>
</tbody>
</table>

#### 4.3 RESERVED WORDS

Reserved words are special strings of characters associated with pre-designated meanings or functions. Reserved words consist of uppercase letters, digits, underscore characters, and possibly a semicolon. The first character must be a letter. Only the last character may be a semicolon. Throughout this document,
a reserved word or keyword is expressed in uppercase letters. These reserved words as a whole or their leading portion (not including the semicolon) cannot be used in entity type keywords.

List of Reserved Words:

SCOPE
ENDSCOPE
DATA;
ENDSEC;
STEP;
HEADER;
ENDSTEP;

4.4 KEYWORDS

Keywords are special strings of characters indicating the occurrence of an entity of a specific type on the STEP file. Keywords consist of uppercase letters, digits, underscore characters, and possibly an exclamation mark. The exclamation mark may occur at most once, and only as the first character in a keyword. Keywords beginning with the exclamation mark are "user-defined entity" keywords. Keywords not beginning with the exclamation mark are standard keywords. The first character in standard keywords as well as the second character of user-defined entity keywords (which is the first one following the exclamation mark) must be an uppercase letter.

The meaning of the user-defined entity keywords is not defined in the standard but is rather a matter of agreement among the partners exchanging STEP files.

The standard keywords are specified with respect to their appearance in the STEP file and with respect to their meaning in this standard. For the header section entities, this specification is given in chapter 6 of this document. For the data section entities, the appearance is defined by the mapping of EXPRESS ENTITIES onto the STEP file format as given in Appendix C. Their semantic meaning is specified in STEP DOCUMENT 4.1.2.

4.5 SIMPLE DATA TYPES

There are six simple data types utilized in STEP exchange format files: integer, real, string, entity identifier, entity reference, and enumeration. The detailed definitions of the data types are presented on the following pages.

4.5.1 Integer

An integer is a string of one or more digits, optionally preceded by a minus sign "-" or a plus sign "+". Integers are expressed in base 10. If there is no sign associated with the integer, it is assumed to be positive.
Examples:

<table>
<thead>
<tr>
<th>Valid STEP Expressions of Integer</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Positive 16</td>
</tr>
<tr>
<td>+12</td>
<td>Positive 12</td>
</tr>
<tr>
<td>-349</td>
<td>Negative 349</td>
</tr>
<tr>
<td>012</td>
<td>Positive 12</td>
</tr>
<tr>
<td>00</td>
<td>Zero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Invalid STEP Expressions of Integer</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 54</td>
<td>Contains blanks</td>
</tr>
<tr>
<td>32.0</td>
<td>Contains decimal point</td>
</tr>
<tr>
<td>+ 12</td>
<td>Contains blank between sign and digits</td>
</tr>
</tbody>
</table>

4.5.2  Real

A Real consists of a required decimal mantissa followed by an optional decimal exponent. The mantissa consists of an optional plus sign "+" or minus sign ",", followed by a string of one or more digits, followed by a decimal point ",", followed by a string of zero or more digits. A decimal exponent consists of the character E followed by an optional plus "+" or minus "-" sign, followed by one or more digits.

Examples:

<table>
<thead>
<tr>
<th>Valid STEP Expressions of Real</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0.0E0</td>
<td>0.0</td>
</tr>
<tr>
<td>-0.0E-0</td>
<td>0.0, as above example</td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>-32.178E+02</td>
<td>-3217.8</td>
</tr>
<tr>
<td>0.25E8</td>
<td>25 million</td>
</tr>
<tr>
<td>0.0E25</td>
<td>0.0</td>
</tr>
<tr>
<td>2.</td>
<td>2.0</td>
</tr>
<tr>
<td>05.</td>
<td>5.0</td>
</tr>
</tbody>
</table>
Invalid STEP Expressions of Real

<table>
<thead>
<tr>
<th></th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2E3.</td>
<td>Decimal point not allowed in exponent</td>
</tr>
<tr>
<td>1E05</td>
<td>Decimal point required in mantissa</td>
</tr>
<tr>
<td>1,000.00</td>
<td>Comma not allowed</td>
</tr>
<tr>
<td>3.E</td>
<td>Digit(s) required in exponent</td>
</tr>
<tr>
<td>.5</td>
<td>At least one digit must precede decimal point</td>
</tr>
<tr>
<td>1</td>
<td>Decimal point required in mantissa</td>
</tr>
</tbody>
</table>

4.5.3 String

A string consists of an apostrophe "'", followed by zero or more characters, and ended by a single apostrophe "'". The null string is defined by two apostrophes in a row. Within a string, a single apostrophe is coded as two apostrophes in a row. The default characters to be used within a string are the printable characters defined by the table entries 32 through 126 (inclusive) of ISO 6937/2. This table is presented in Appendix B of this document. Four examples are listed below.

<table>
<thead>
<tr>
<th>String</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>'CAT'</td>
<td>CAT</td>
</tr>
<tr>
<td>'Don''t'</td>
<td>Don't</td>
</tr>
<tr>
<td>'&quot;'</td>
<td>'</td>
</tr>
<tr>
<td>'&quot;'</td>
<td>'' Null</td>
</tr>
</tbody>
</table>

4.5.3.1 Alternate Character Sets

An alternate character set may be used if an escape sequence is coded into the character string. A description of a character string that contains an escape sequence is defined as follows:

1. A single apostrophe to start the character string.
2. Any number of characters from the default character set.
3. A single exclamation point immediately followed by a single colon to open or begin the escape sequence.
4. A reserved word to indicate the character set table to be used (e.g., GREEK or KANJI).
5. A comma to delimit the reserved word.
6. An integer value that corresponds to the table index of the desired character in the character set table.

7. A comma to delimit the integer value.

8. Successive integer values, each separated from the next by a comma.

9. The final integer value corresponding to a table index for the desired character.

10. The closing escape sequence character pair (i.e., a single colon immediately followed by an exclamation point).

11. Any additional characters or escape sequences.

12. A single apostrophe to close the character string.

In order to avoid being interpreted as an escape sequence, the character "!:" must be encoded in a normal character string by being repeated once as a pair, (e.g., "!:!").

The recognized character set tables and their associated reserved words are presented in Appendix B.

4.5.3.2 Maximum String Length

The maximum length of a string in a STEP file is 32767 characters not counting the beginning and ending apostrophes.
4.5.4 Entity Identifier

An entity identifier consists of an at sign "@" followed by an unsigned integer. The integer may consist of any combination of 1 to 9 digits as long as uniqueness within the file is maintained. Leading zeroes in entity identifiers are not significant. The entity identifier is not based on a relative record position. They need not be sequentially ordered in the file. (See Section 7.1).

<table>
<thead>
<tr>
<th>Valid STEP Expressions of Entity Identifiers</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>@23</td>
<td>Names this entity with identifier 23</td>
</tr>
<tr>
<td>@012</td>
<td>Names this entity with identifier 12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Invalid STEP Expressions of Entity Identifier</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>@+23</td>
<td>Contains '+' sign</td>
</tr>
<tr>
<td>@500.1</td>
<td>Contains decimal point</td>
</tr>
</tbody>
</table>

4.5.5 Entity Reference

An entity reference consists of a number sign "#", followed by an unsigned integer. The integer may consist of any combination of 1 to 9 digits. This integer value must equal the integer value within some previously defined entity identifier. The entity reference is not based on a relative record position. Leading zeroes are not significant.

Examples:

<table>
<thead>
<tr>
<th>Valid STEP Expressions of Entity References</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>#23</td>
<td>Refers to entity occurrence with identifier 23</td>
</tr>
<tr>
<td>#023</td>
<td>Refers to entity occurrence with identifier 23</td>
</tr>
<tr>
<td>#694376142</td>
<td>Refers to entity occurrence with identifier 694376142</td>
</tr>
</tbody>
</table>
4.5.6 Enumeration

An enumeration value consists of some combination of uppercase letters or digits beginning with an uppercase letter. The entire collection of uppercase letters and digits is bracketed by periods. The meaning of a given value is determined by the conceptual schema which defined the enumerated list.

Examples:

<table>
<thead>
<tr>
<th>Valid STEP Expressions of Enumerated Values</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>.FALSE.</td>
<td>Indicates a value of FALSE</td>
</tr>
<tr>
<td>.STEEL.</td>
<td>Indicates a value of STEEL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Invalid STEP Expressions of Enumerated Values</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>.TRUE</td>
<td>Missing ending period</td>
</tr>
<tr>
<td>.123.</td>
<td>Did not start with an alphabetic character.</td>
</tr>
</tbody>
</table>
CHAPTER 5 - STRUCTURED DATA TYPES

5.0 STRUCTURED DATA TYPE

The list is the only structured data type in the STEP file structure.

5.1 LIST

A list is a homogeneous collection of instances of simple or structured data types. It is begun by a left parenthesis "(" and terminated by a right parenthesis ")". Instances are delimited by commas. Lists may be nested arbitrarily deep.

Examples:

List of List of Real = ((0.0, 1.0, 2.0), (3.0, 4.0, 5.0))
List of Integer = (0, 1, 2, 3, 7, 2, 4)
List of String = ('CAT', 'HELLO')
CHAPTER 6 - HEADER SECTION

6.0 HEADER SECTION

The HEADER section contains information that is applicable to the entire exchange format file. This section must be present in any STEP file. Refer to Appendix A for a complete example.

6.1 HEADER SECTION ENTITIES

These entities are to aid in the overall data exchange process. The HEADER section entities follow the same general syntax of the DATA section entities, which simplifies postprocessor parsing software. HEADER section entity occurrences are defined in the following manner:

ENTITY_TYPE ( one or more attributes);

where

Entity Type - A keyword through which the type of the entity is identified. The keywords correspond to those listed in Table 6-1. The keyword is delimited by a left parenthesis "(".

Attribute - An instance of a simple or structured data type. The correspondence of attributes and keywords is maintained in Table 6-1. Attributes are separated by commas. The final attribute is delimited by a right parenthesis ")". 
### Table 6-1: Keyword-To-Attribute Relationships

<table>
<thead>
<tr>
<th>ENTITY KEYWORD</th>
<th>ATTRIBUTE NAME</th>
<th>DATATYPE</th>
<th>ATTRIBUTE POSITION</th>
<th>STATUS OF ENTITY OR ATTRIBUTE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_IDENTIFICATION</td>
<td>FILE NAME</td>
<td>String</td>
<td>1</td>
<td>Required</td>
<td>This string of characters is used to name this STEP file. It is intended to be used as a person-to-person communication between sender and receiver. There are no special restrictions on the content of this field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATE</td>
<td>String</td>
<td>2</td>
<td>Required</td>
<td>This identifies the local date and time the file was created, expressed as YYYY-MM-DD HH:MM:SS. This format corresponds to one of those specified in ISO Standard 8601.</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>List of String</td>
<td>3</td>
<td>Required</td>
<td>This identifies the person responsible for creating the exchange format file and the mailing address of that person.</td>
<td></td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>List of String</td>
<td>4</td>
<td>Required</td>
<td>This identifies the group or organization with which the author is associated.</td>
<td></td>
</tr>
<tr>
<td>STEP VERSION</td>
<td>String</td>
<td>5</td>
<td>Required</td>
<td>This indicates the STEP version used in creating the file.</td>
<td></td>
</tr>
<tr>
<td>PREPROCESSOR VERSION</td>
<td>String</td>
<td>6</td>
<td>Required</td>
<td>This identifies the preprocessor software used to create the STEP file. It includes the software product name and version.</td>
<td></td>
</tr>
<tr>
<td>ORIGINATING SYSTEM</td>
<td>String</td>
<td>7</td>
<td>Required</td>
<td>This identifies the CAD/CAM/CIM system from which the data in this STEP file originated.</td>
<td></td>
</tr>
<tr>
<td>FILE_DESCRIPTION</td>
<td>DESCRIPTION</td>
<td>String</td>
<td>1</td>
<td>Required</td>
<td>This is an informal description of the contents of this STEP file.</td>
</tr>
<tr>
<td>IMP_LEVEL</td>
<td>IMPLEMENTATION LEVEL</td>
<td>String</td>
<td>1</td>
<td>Optional Required</td>
<td>This indicates the required implementation level of the postprocessor software in order for it to completely process all of the data in this STEP file. The actual values for this will be determined at a later date.</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>SECURITY CLASSIFICATION</td>
<td>String</td>
<td>1</td>
<td>Optional Required</td>
<td>This identifies the restrictions on the use of the information contained in this file. The default is an empty string.</td>
</tr>
<tr>
<td>MAXSIG</td>
<td>MAXIMUM SIGNIFICANT DIGIT</td>
<td>Integer</td>
<td>1</td>
<td>Optional Required</td>
<td>This indicates the number of decimal digits of significance that can be represented accurately on this file.</td>
</tr>
</tbody>
</table>
6.1.1 HEADER Section User Defined Entities

User defined entities may be placed into the HEADER section under specific restrictions. Post-processor translator software should be prepared to ignore those entities it does not recognize. This precaution will allow the future introduction of new HEADER section entities without adversely affecting any existing translator software.

The four restrictions that limit the specification of user defined entities are listed below.

1. User defined entities must conform to the same general syntax of all HEADER section entities with the exception that the entity-type keyword is immediately preceded by an exclamation point "!".

2. User defined entities may not appear as embedded entities within a standard entity.

3. User defined entities may only be embedded within other user defined entities.

4. User defined entities may not use any delimiters, simple data types, or structured data types beyond those defined in the standard.

Example

HEADER;

IMP_LEVEL ('BREP - LEVEL 1.0');
!A_SPECIAL_ENTITY ('ABC',123); /* SAMPLE USER DEFINED ENTITY */

ENDSEC;
7.0 DATA SECTION

The DATA section is used to define the product data to be transferred. It contains occurrences of entities defined in the STEP conceptual schema. This section must appear in the file. Throughout the DATA section, entity occurrences must be defined before they may be referenced. Refer to Appendix A for a complete example.

7.1 DATA SECTION ENTITIES

DATA section entity occurrences are defined in the following manner:

ENTITY_IDENTIFIER = ENTITY_TYPE optional SCOPE Structure
(one or more attribute values);

where

Entity Identifier - Refer to the definition given in section 4.5.4.
Entity Type - A keyword through which the type of the entity is identified. The keyword is delimited by a left parenthesis "(", or a space " ". These keywords correspond to the entities defined in the conceptual schema. Refer to the definition given in section 4.4.
Optional SCOPE Structure See the discussion in Chapter 7.2.
Attribute - An instance of a simple or structured data type, or an instance of an embedded entity. Attributes are separated by commas. The final attribute is delimited by a right parenthesis ")". The entire entity occurrence is delimited by a semicolon ";".

Embedded Entity - An embedded entity is an entity that is serving as an attribute of the entity in which it is embedded. Syntactically, it appears like an entity occurrence that is missing the entity identifier, the equal sign "=" and the final semicolon ";". Under circumstances described in Normative Annex C, the entity type keyword may be omitted.

The significance of an embedded entity is discussed within Normative Annex C, "Mapping From EXPRESS To Physical File Structure".

7-1
7.1.1 Optional Values

When an optional attribute value is not provided, all delimiters that would normally appear when the value is present will be maintained.

Example: @32 = NODE(1.0,2.0,'ABC');

This entity has an optional second attribute value that is not provided in this occurrence.

7.2 SCOPE STRUCTURE

The SCOPE structure within entity definitions is a mechanism for providing a scope of reference.

7.2.1 Syntax

The SCOPE structure is optional for all entity types. If it appears, it must appear following the entity type keyword and before the actual defining attributes of the entity in which it is appearing. It is begun by the reserved word "SCOPE". Following that are any number of entity occurrences. Syntactically, it is legal for the entity occurrences to be of any type of entity. The structure is terminated by the reserved word "ENDSCOPE". After the ENDScope reserved word, an optional entity export list may appear. Entities within a scope structure may themselves have scope sections. See Chapter 3.4 for the formal syntactical descriptions.

7.2.2 Referencability

In a similar manner to the scoping of variables in common structured programming languages, entities defined in a SCOPE structure are not "visible" to, or referencable by, entities that reside in outer scopes unless they are explicitly made "visible" by means of their inclusion in the entity export list. The defining attributes of the entity may refer to any entity occurrences in its SCOPE structure. Entities which need to be "shared" i.e., multiply referenced, by a number of entities, need to be defined within a scope that is common to the referencing entities.

7.2.3 Behavior

The behavior of entity occurrences which appear within a SCOPE structure is defined as follows:

- Modifications of the parent entity will be passed onto the scoped entity(s) if and only if that modification is applicable to the scoped entity(s). For example, transformation of the parent entity will transform all the scoped entities to which the transformation operation applies; deletion of the parent entity will delete all the scoped entities (deletion applies to entities of any type). Approval of the parent entity will approve all the scoped entities to which this operation applies.
 Scoped entity occurrences made "visible" by the entity export list are considered in the same scope of reference as the parent entity. Other entities, capable of referencing the parent entity, may legally reference the entities appearing in the entity export list.

 When modifications to an exported entity conflict with modifications performed on the parent entity, the modifications on the parent entity will take precedence. For example, if an exported entity is referenced by a given transformation matrix (TM₁), and the parent entity is referenced by a different transformation matrix (TM₂), the exported entity is transformed by TM₂ and not TM₁.

 Some examples of the use of scoping follow:

 Example 1

```
@1 = TRIANGLE
  SCOPE
@2 = POINT (0.0,0.0,0.0);
@3 = POINT (0.0,1.0,0.0);
@4 = POINT (3.0,1.0,0.0);
ENDSCOPE
  (LINE(#2,#3),LINE(#3,#4),LINE(#4,#2));
```

In the above example the entity TRIANGLE is defined by three LINE entities. Since the LINE entities are embedded within the TRIANGLE, they have an existence dependency upon the TRIANGLE. The POINT entities which define the lines are also dependent upon the TRIANGLE for their existence. Since the POINT entities have entity identifiers, they are available to be referenced by the embedded LINE entities in the attribute list.

 Example 2

```
@1 = ASSEMBLY
  SCOPE
@12 = PART ('PART #74A',...);
@13 = PART ('PART #68B',...);
@14 = PART ('PART #12C',...);
.
.
ENDSCOPE/#13, #14/
  ((#6,#7),...);
@15 = APPROVAL ((#13), 'JOHN SMITH APP. AUTHORITY');
```
In the above example, the PART entities with entity identifiers 013 and 014 are exported, or made referencable, from the SCOPE structure of the ASSEMBLY entity. Following the defining attributes of the ASSEMBLY entity, an APPROVAL entity occurs which references the PART entity with entity identifies 013. In this case, it is only that particular PART entity which is approved, while at the same time maintaining the existence dependency of that PART entity upon its parent ASSEMBLY.

7.3 EXTERNAL REFERENCES

An external reference is the mechanism by which an entity in a STEP file may refer to information that is physically external to the STEP file being processed. The type of information that is referred to can be of any type, not necessarily geometric in nature. An external reference can be thought of as an extension of the basic entity reference capability that has already been provided (see Chapter 4.5.5). A number of specialized entities have been formulated to support external references. Specifically, these are the EXPORT, LIBRARY, and EXTERNAL REFERENCE entities. Their definitions and the details of their use can be found in the Conceptual Schema document (STEP document 4.1.2).

7.4 INDEX OF ENTITY ALIASES

An INDEX entity has been formulated which allows the establishment of a character string alias(s) for a given entity. Its definition and the details of its use can be found in the Conceptual Schema document (STEP document 4.1.2).

7.5 DATA SECTION USER DEFINED ENTITIES

User defined entities may be placed into the DATA section under specific restrictions. Post-processor translator software should be prepared to ignore those entities it does not recognize. This precaution will allow the future introduction of new DATA section entities without adversely affecting any existing translator software.

The four restrictions that limit the specification of user defined entities are listed below.

1. User defined entities must conform to the same general syntax of all DATA section entities with the exception that the entity-type keyword is immediately preceded by an exclamation point "!".

2. User defined entities may not appear as embedded entities within a standard entity.

3. User defined entities may only be embedded within other user defined entities.

4. User defined entities may not use any delimiters, simple data types, or structured data types beyond those defined in the standard.
EXAMPLE

DATA;

@1=PT(1.0,2.0,3.0); /* STANDARD POINT ENTITY */
@2=PT(1.0,2.0,5.0);

@12=MYCURVE(0.0,0.0,0.0,1.0,..); /* SAMPLE USER DEFINED ENTITY */

ENDSEC;
APPENDIX A - SAMPLE STEP EXCHANGE FORMAT FILE

A.1 STEP FILE FORMAT

An example STEP file is presented below.

STEP;
HEADER;
FILE_IDENTIFICATION('EXAMPLE STEP FILE #1', '19880211.153000', ('JOHN DOE', 'ACME INC.', 'METROPOLIS USA'), ('ACME INC. A SUBSIDIARY OF GIANT INDUSTRIES', 'METROPOLIS USA'), 'STEP VERSION 1.0', 'CIM/STEP VERSION 2', 'SUPER CIM SYSTEM RELEASE 4.0');
FILE_DESCRIPTION('THIS FILE CONTAINS A SMALL SAMPLE STEP MODEL');IMP_LEVEL ('BREP-LEVEL 1.0');
ENDSEC;
DATA;
/*
THE FOLLOWING 13 ENTITIES REPRESENT A TRIANGULAR EDGE LOOP */
@1=PT(0.0,0.0,0,0.0); /* THIS IS A POINT ENTITY */
@2=PT(0.0,1.0,0,0.0);
@3=PT(1.0,0,0,0.0);
@11=VX(#1); /* THIS IS A VERTEX ENTITY */
@12=VX(#2);
@13=VX(#3);
@16=ED(#11,#12); /* THIS IS AN EDGE ENTITY */
@17=ED(#11,#13);
@18=ED(#13,#12);
@21=ED_STRC(#17,0); /* THIS IS AN EDGE STRUCTURE ENTITY */
@22=ED_STRC(#18,0);
@23=ED_STRC(#16,1);
@24=ED_LOOP((#21,#22,#23)); /* THIS IS AN EDGE_LOOP ENTITY */

/* ANOTHER WAY OF REPRESENTING THE TRIANGULAR EDGE_LOOP FOLLOWS: */
@110=VX(PT(0.0,0.0,0.0));
@120=VX(PT(0.0,1.0,0.0));
@130=VX(PT(1.0,0,0,0.0));
@160=ED(#110,#120);
@170=ED(#110,#130);
@180=ED(#130,#120);
@240=ED_LOOP((ED_STRC(#170,0),ED_STRC(#180,0),ED_STRC(#160,1)));
/* YET ANOTHER WAY OF REPRESENTING THE TRIANGULAR EDGE_LOOP FOLLOWS: */
@1100=VX(PT(0.0,0.0,0.0));
@1200=VX(PT(0.0,1.0,0.0));
@1300=VX(PT(1.0,0.0,0.0));
@2400=ED_LOOP
  SCOPE
    @1600=ED(#1100,#1200);
    @1700=ED(#1100,#1300);
    @1800=ED(#1300,#1200);
  ENDScope
  ((ED_STRC(#1700,0),ED_STRC(#1800,0),ED_STRC(#1600,1)));
/* OTHER SYNTACTICAL REPRESENTATIONS WERE POSSIBLE. THE PREVIOUS THREE
EXAMPLES ARE REPRESENTATIVE OF COMMON APPROACHES. IT SHOULD BE NOTED
THAT EACH APPROACH PRESENTED HERE CARRIES DIFFERENT SEMANTICS. */
ENDSEC;
ENDSTEP;

NOTE: This example STEP file has been presented in a line-oriented or
record-oriented manner in order to aid readability. Unnecessary
spaces have also been added to aid readability. Note that an ordinary
STEP file is not aligned in this manner, but is instead a continuous
stream of characters.
APPENDIX B - THE GRAPHICAL ALPHABET

B.0  THE GRAPHICAL ALPHABET

The graphical alphabet of this standard comprises an extendible set of alphabets. Each alphabet is identified by a reserved word. Each alphabet contains:

1) a listing of the graphical representations of all symbols of the alphabet.
2) an algorithm that uniquely maps each of these symbols onto a positive integer (called decimal coded character, DCC).

The representation of these DCC's on the physical file is subject to the various presentation techniques allowed in this standard (see Appendix D of this document). At this time, only the "clear text encoding" is defined (see also ESC_CHAR, ESC_SEQ, and ESC_ITEM in Table 3-3 "Token definition").

At this time the following alphabets are defined:

<table>
<thead>
<tr>
<th>Reserved Word</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>the basic character set</td>
</tr>
<tr>
<td>ISO6937</td>
<td>the Latin alphabet</td>
</tr>
<tr>
<td>GREEK</td>
<td>the Greek alphabet</td>
</tr>
<tr>
<td>KANJI</td>
<td>the Kanji alphabet</td>
</tr>
</tbody>
</table>

Other alphabets will be included in this standard:

- an alphabet for drafting symbols
- alphabets for various other applications

B.1  BASIC CHARACTER SET (Escape Sequence Reserved Word = ISO6937)

The character set used to encode STEP files is ASCII defined by ISO 6937/1. Furthermore, only the printable characters defined by the table entries 32 through 126 (inclusive) are ever allowed within a STEP file.
The decimal value of the alphabet element appears in the first line of each pair of lines illustrated below. The graphical representation of the alphabet element appears in the second line where $G(i)$ denotes the graphical representation of the decimal i. The international monetary symbol cannot be printed in the text system that generates this report, therefore a dollar sign is shown for $G(36)$.

Table B-1 Graphical Character Set

<table>
<thead>
<tr>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
</tr>
<tr>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
</tr>
<tr>
<td>-</td>
<td>.</td>
<td>/</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
</tr>
<tr>
<td>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>81</td>
<td>82</td>
<td>83</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
</tr>
<tr>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
</tr>
<tr>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
<td>109</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
</tr>
<tr>
<td>110</td>
<td>111</td>
<td>112</td>
<td>113</td>
<td>114</td>
<td>115</td>
<td>116</td>
<td>117</td>
<td>118</td>
<td>119</td>
<td>120</td>
<td>121</td>
<td>122</td>
</tr>
<tr>
<td>n</td>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
</tr>
<tr>
<td>123</td>
<td>124</td>
<td>125</td>
<td>126</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(</td>
<td></td>
<td>)</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B.1.1 Subsets

The range of characters, 32 through 126, are further refined into the following subsets.

B.1.2 Digit

<table>
<thead>
<tr>
<th>ASCII Decimal Number</th>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>0</td>
<td>Zero</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>One</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>Two</td>
</tr>
<tr>
<td>51</td>
<td>3</td>
<td>Three</td>
</tr>
<tr>
<td>52</td>
<td>4</td>
<td>Four</td>
</tr>
<tr>
<td>53</td>
<td>5</td>
<td>Five</td>
</tr>
<tr>
<td>54</td>
<td>6</td>
<td>Six</td>
</tr>
<tr>
<td>55</td>
<td>7</td>
<td>Seven</td>
</tr>
<tr>
<td>56</td>
<td>8</td>
<td>Eight</td>
</tr>
<tr>
<td>57</td>
<td>9</td>
<td>Nine</td>
</tr>
</tbody>
</table>
### ANNEX B (Normative)
* (Document 4.2.1 Draft Paper) 

#### B.1.3 Upper

<table>
<thead>
<tr>
<th>ASCII Decimal Number</th>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>A</td>
<td>UPPERCASE A</td>
</tr>
<tr>
<td>66</td>
<td>B</td>
<td>UPPERCASE B</td>
</tr>
<tr>
<td>67</td>
<td>C</td>
<td>UPPERCASE C</td>
</tr>
<tr>
<td>68</td>
<td>D</td>
<td>UPPERCASE D</td>
</tr>
<tr>
<td>69</td>
<td>E</td>
<td>UPPERCASE E</td>
</tr>
<tr>
<td>70</td>
<td>F</td>
<td>UPPERCASE F</td>
</tr>
<tr>
<td>71</td>
<td>G</td>
<td>UPPERCASE G</td>
</tr>
<tr>
<td>72</td>
<td>H</td>
<td>UPPERCASE H</td>
</tr>
<tr>
<td>73</td>
<td>I</td>
<td>UPPERCASE I</td>
</tr>
<tr>
<td>74</td>
<td>J</td>
<td>UPPERCASE J</td>
</tr>
<tr>
<td>75</td>
<td>K</td>
<td>UPPERCASE K</td>
</tr>
<tr>
<td>76</td>
<td>L</td>
<td>UPPERCASE L</td>
</tr>
<tr>
<td>77</td>
<td>M</td>
<td>UPPERCASE M</td>
</tr>
<tr>
<td>78</td>
<td>N</td>
<td>UPPERCASE N</td>
</tr>
<tr>
<td>79</td>
<td>O</td>
<td>UPPERCASE O</td>
</tr>
<tr>
<td>80</td>
<td>P</td>
<td>UPPERCASE P</td>
</tr>
<tr>
<td>81</td>
<td>Q</td>
<td>UPPERCASE Q</td>
</tr>
<tr>
<td>82</td>
<td>R</td>
<td>UPPERCASE R</td>
</tr>
<tr>
<td>83</td>
<td>S</td>
<td>UPPERCASE S</td>
</tr>
<tr>
<td>84</td>
<td>T</td>
<td>UPPERCASE T</td>
</tr>
<tr>
<td>85</td>
<td>U</td>
<td>UPPERCASE U</td>
</tr>
<tr>
<td>86</td>
<td>V</td>
<td>UPPERCASE V</td>
</tr>
<tr>
<td>87</td>
<td>W</td>
<td>UPPERCASE W</td>
</tr>
<tr>
<td>88</td>
<td>X</td>
<td>UPPERCASE X</td>
</tr>
<tr>
<td>89</td>
<td>Y</td>
<td>UPPERCASE Y</td>
</tr>
<tr>
<td>90</td>
<td>Z</td>
<td>UPPERCASE Z</td>
</tr>
<tr>
<td>95</td>
<td>_</td>
<td>Underscore</td>
</tr>
</tbody>
</table>
B.1.4 Lower

<table>
<thead>
<tr>
<th>ASCII Decimal Number</th>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>a</td>
<td>lowercase A</td>
</tr>
<tr>
<td>98</td>
<td>b</td>
<td>lowercase B</td>
</tr>
<tr>
<td>99</td>
<td>c</td>
<td>lowercase C</td>
</tr>
<tr>
<td>100</td>
<td>d</td>
<td>lowercase D</td>
</tr>
<tr>
<td>101</td>
<td>e</td>
<td>lowercase E</td>
</tr>
<tr>
<td>102</td>
<td>f</td>
<td>lowercase F</td>
</tr>
<tr>
<td>103</td>
<td>g</td>
<td>lowercase G</td>
</tr>
<tr>
<td>104</td>
<td>h</td>
<td>lowercase H</td>
</tr>
<tr>
<td>105</td>
<td>i</td>
<td>lowercase I</td>
</tr>
<tr>
<td>106</td>
<td>j</td>
<td>lowercase J</td>
</tr>
<tr>
<td>107</td>
<td>k</td>
<td>lowercase K</td>
</tr>
<tr>
<td>108</td>
<td>l</td>
<td>lowercase L</td>
</tr>
<tr>
<td>109</td>
<td>m</td>
<td>lowercase M</td>
</tr>
<tr>
<td>110</td>
<td>n</td>
<td>lowercase N</td>
</tr>
<tr>
<td>111</td>
<td>o</td>
<td>lowercase O</td>
</tr>
<tr>
<td>112</td>
<td>p</td>
<td>lowercase P</td>
</tr>
<tr>
<td>113</td>
<td>q</td>
<td>lowercase Q</td>
</tr>
<tr>
<td>114</td>
<td>r</td>
<td>lowercase R</td>
</tr>
<tr>
<td>115</td>
<td>s</td>
<td>lowercase S</td>
</tr>
<tr>
<td>116</td>
<td>t</td>
<td>lowercase T</td>
</tr>
<tr>
<td>117</td>
<td>u</td>
<td>lowercase U</td>
</tr>
<tr>
<td>118</td>
<td>v</td>
<td>lowercase V</td>
</tr>
<tr>
<td>119</td>
<td>w</td>
<td>lowercase W</td>
</tr>
<tr>
<td>120</td>
<td>x</td>
<td>lowercase X</td>
</tr>
<tr>
<td>121</td>
<td>y</td>
<td>lowercase Y</td>
</tr>
<tr>
<td>122</td>
<td>z</td>
<td>lowercase Z</td>
</tr>
</tbody>
</table>
B.2 THE LATIN ALPHABET (ISO6937)

ISO6937/2 is used as a basis for coding the "decimal coded character" representations (DCC) of Latin characters. The coding method B of the ISO standard is used. According to this method, Latin characters are coded by one or two pairs of integers:

(1) \( \frac{n_4}{n_3} \frac{n_2}{n_1} \)
(2) \( \frac{n_4}{n_3} \) -
(3) - \( \frac{n_2}{n_1} \)

where

- \( n_4 \) is greater than or equal to 8 and less than or equal to 15
- \( n_3 \) is greater than or equal to 0 and less than or equal to 7
- \( n_2 \) is greater than or equal to 0 and less than or equal to 7
- \( n_1 \) is greater than or equal to 0 and less than or equal to 7

Syntactically the three forms are distinguished as:

\[
\begin{align*}
\text{<one_to_seven> } & : = 1 | 2 | 3 | 4 | 5 | 6 | 7 \\
\text{<eight_to_fifteen> } & : = 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 \\
\text{<form1> } & : = \text{<form2> } \text{<form3> } \\
\text{<form2> } & : = \text{<eight_to_fifteen> } | \text{<one_to_seven> } \\
\text{<form3> } & : = \text{<one_to_seven> } | \text{<one_to_seven> }
\end{align*}
\]

The second form defines implicitly \( n_2 = n_1 = 0 \).

The third form defines implicitly \( n_4 = 8, n_3 = 0 \).

From this representation method the DCC is defined as follows:

\[
\text{DCC} = 16^*(8^*(16*(n_4-8) + n_3) + n_2) + n_1
\]
The algorithm (formulated in FORTRAN77) to derive the 4 numbers $n_1-n_4$ from a DCC is as follows:

```fortran
subroutine tran(dcc,n1,n2,n3,n4)
  integer dcc,var1,var2,n1,n2,n3,n4
  var1=dcc
  var2=var1/16
  n1=var1-(16*var2)
  var1=var2
  var2=var1/8
  n2=var1-(8*var2)
  var1=var2
  var2=var1/16
  n3=var1-(16*var2)
  n4=var2+8
return
end
```

B-7
B.3 GREEK CHARACTER SET (Escape Sequence Reserved Word = GREEK)

The Greek character set is conveyed by the character set encoding in ISO/DIS 6937/7 "Information processing -- Coded character sets for text communication -- Part 7 Greek graphic characters".
APPENDIX C - ENTITY SYNTAX ON THE PHYSICAL FILE

In this appendix, words typed in lowercase letters represent items that were previously defined by the file structure syntax (See Tables 3-3 and 3-4).

AXIS1_PLACEMENT-AXIS = entity_reference | DIRECTION .
AXIS1_PLACEMENT-LOCATION = entity_reference | POINT .
AXIS1_PLACEMENT = "(" [ AXIS1_PLACEMENT-AXIS ] ","
AXIS1_PLACEMENT-LOCATION ")" .

AXIS2_PLACEMENT-AXIS = entity_reference | DIRECTION .
AXIS2_PLACEMENT-LOCATION = entity_reference | DIRECTION .
AXIS2_PLACEMENT-REF_DIRECTION = entity_reference | DIRECTION .
AXIS2_PLACEMENT = "(" [ AXIS2_PLACEMENT-AXIS ] ","
AXIS2_PLACEMENT-LOCATION ","
[ AXIS2_PLACEMENT-REF_DIRECTION ] ")" .

AXIS_PLACEMENT = AXIS2_PLACEMENT | AXIS1_PLACEMENT .

BEZIER_CURVE-CONTROL_POINTS = list .
BEZIER_CURVE-DEGREE = integer .
BEZIER_CURVE-FORM_NUMBER = "."string"." .
BEZIER_CURVE-KNOTS = list .
BEZIER_CURVE-RATIONAL = logical .
BEZIER_CURVE-SELF_INTERSECT = logical .
BEZIER_CURVE-WEIGHTS = list .
BEZIER_CURVE = "(" [ BEZIER_CURVE-CONTROL_POINTS ] ","
BEZIER_CURVE-DEGREE ","
[ BEZIER_CURVE-FORM_NUMBER ] ","
BEZIER_CURVE-KNOTS ","
BEZIER_CURVE-RATIONAL ","
BEZIER CURVE-SELF_INTERSECT ","
BEZIER_CURVE-WEIGHTS ")" .

BEZIER_SURFACE-CONTROL_POINTS = list .
BEZIER_SURFACE-FORM_NUMBER = "."string"." .
BEZIER_SURFACE-U_DEGREE = integer .
BEZIER_SURFACE-U_RATIONAL = integer .
BEZIER_SURFACE-U_UNIFORM = logical .
BEZIER_SURFACE-U_UPPER = integer .
BEZIER_SURFACE-V_DEGREE = integer .
BEZIER_SURFACE-V_RATIONAL = logical .
BEZIER_SURFACE-V_UNIFORM = logical .
BEZIER_SURFACE-V_UPPER = integer .
BEZIER_SURFACE-WEIGHTS = list .
BEZIER_SURFACE = "(" [ BEZIER_SURFACE-CONTROL_POINTS ] ","
BEZIER_SURFACE-FORM_NUMBER ","
BEZIER_SURFACE-U_DEGREE ","
BEZIER_SURFACE-U_RATIONAL ")" .
BEZIER_SURFACE-U_UNIFORM ",", BEZIER_SURFACE-U_UPPER ",", BEZIER_SURFACE-V_DEGREE ",", BEZIER_SURFACE-V_RATIONAL ",", BEZIER_SURFACE-V_UNIFORM ",", BEZIER_SURFACE-V_UPPER ",", 
( BEZIER_SURFACE-WEIGHTS ) 
"

BOUNDUCURVE = COMPOSITE_CURVE
 | TRIMMED_CURVE
 | B_SPLINE_CURVE
 | BEZIER_CURVE
 | LINE_SEGMENT

BOUNDED_SURFACE = CURVE_BOUNDED_SURFACE
 | RECTANGULAR_TRIMMED_SURFACE
 | BEZIER_SURFACE
 | B_SPLINE_SURFACE

B_SPLINE_CURVE-CONTROL_POINTS = list.
B_SPLINE_CURVE-DEGREE = integer.
B_SPLINE_CURVE-FORM_NUMBER = "."string".".
B_SPLINE_CURVE-KNOTS = list.
B_SPLINE_CURVE-KNOT_MULTIPLICITIES = list.
B_SPLINE_CURVE-RATIONAL = logical.
B_SPLINE_CURVE-SELF_INTERSECT = logical.
B_SPLINE_CURVE-UNIFORM = logical.
B_SPLINE_CURVE-UPPER_INDEX_ON_CONTROL_POINTS = integer.
B_SPLINE_CURVE-WEIGHTS = list.
B_SPLINE_CURVE = "(
B_SPLINE_CURVE-CONTROL_POINTS ",",
B_SPLINE_CURVE-DEGREE ",",
[ B_SPLINE_CURVE-FORM_NUMBER ] ",",
[ B_SPLINE_CURVE-KNOTS ] ",",
B_SPLINE_CURVE-KNOT_MULTIPLICITIES ",",
B_SPLINE_CURVE-RATIONAL ",",
B_SPLINE_CURVE-SELF_INTERSECT ",",
B_SPLINE_CURVE-UNIFORM ",",
B_SPLINE_CURVE-UPPER_INDEX_ON_CONTROL_POINTS ",",
B_SPLINE_CURVE-WEIGHTS 
")"

B_SPLINE_SURFACE-CONTROL_POINTS = list.
B_SPLINE_SURFACE-FORM_NUMBER = "."string".".
B_SPLINE_SURFACE-U_DEGREE = integer.
B_SPLINE_SURFACE-U_KNOTS = list.
B_SPLINE_SURFACE-U_MULTIPLICITIES = list.
B_SPLINE_SURFACE-U_RATIONAL = logical.
B_SPLINE_SURFACE-U_UNIFORM = logical.
B_SPLINE_SURFACE-U_UPPER = integer.
B_SPLINE_SURFACE-V_DEGREE = integer.
B_SPLINE_SURFACE-V_KNOTS = list.
B_SPLINE_SURFACE-V_MULTIPLICITIES = list .
B_SPLINE_SURFACE-V_RATIONAL = logical .
B_SPLINE_SURFACE-V_UNIFORM = logical .
B_SPLINE_SURFACE-V_UPPER = integer .
B_SPLINE_SURFACE-WEIGHTS = list .
B_SPLINE_SURFACE = "(" 
B_SPLINE_SURFACE-CONTROL_POINTS "," 
[ B_SPLINE_SURFACE-FORM_NUMBER ] "," 
B_SPLINE_SURFACE-U_DEGREE "," 
B_SPLINE_SURFACE-U_KNOTS "," 
B_SPLINE_SURFACE-U_MULTIPLICITIES "," 
B_SPLINE_SURFACE-U_RATIONAL "," 
B_SPLINE_SURFACE-U_UNIFORM "," 
B_SPLINE_SURFACE-U_UPPER "," 
B_SPLINE_SURFACE-V_DEGREE "," 
B_SPLINE_SURFACE-V_KNOTS "," 
B_SPLINE_SURFACE-V_MULTIPLICITIES "," 
B_SPLINE_SURFACE-V_RATIONAL "," 
B_SPLINE_SURFACE-V_UNIFORM "," 
B_SPLINE_SURFACE-V_UPPER "," 
[ B_SPLINE_SURFACE-WEIGHTS ] 
")" .
CARTESIAN_POINT = CARTESIAN_THREE_COORDINATE 
| CARTESIAN_TWO_COORDINATE .
CARTESIAN_THREE_COORDINATE-X_COORDINATE = real .
CARTESIAN_THREE_COORDINATE-Y_COORDINATE = real .
CARTESIAN_THREE_COORDINATE-Z_COORDINATE = real .
CARTESIAN_THREE_COORDINATE = "(" 
CARTESIAN_THREECOORDINATE-XCOORDINATE "," 
CARTESIAN_THREECOORDINATE-YCOORDINATE "," 
CARTESIAN_THREECOORDINATE-ZCOORDINATE 
")" .
CARTESIAN_TWO_COORDINATE-X_COORDINATE = real .
CARTESIAN_TWO_COORDINATE-Y_COORDINATE = real .
CARTESIAN_TWO_COORDINATE = "(" 
CARTESIAN_TWOCOORDINATE-XCOORDINATE "," 
CARTESIAN_TWOCOORDINATE-YCOORDINATE 
")" .
CIRCLE-POSITION = entity_reference | AXIS2_PLACEMENT .
CIRCLE-RADIUS = real .
CIRCLE = "(" 
CIRCLE-POSITION "," 
CIRCLE-RADIUS 
")" .
CLOSED_SHELL-CSHELL_BOUNDARY = list .
CLOSED_SHELL = "(" 
CLOSED_SHELL-CSHELL_BOUNDARY 
")" .
COMPOSITE_CURVE-CLOSED_CURVE = logical.
COMPOSITE_CURVE-PARAM_RANGE = list.
COMPOSITE_CURVE-SEGMENTS = list.
COMPOSITE_CURVE-SELF_INTERSECT = logical.
COMPOSITE_CURVE-SENSES = list.
COMPOSITE_CURVE-TRANSITIONS = list.
COMPOSITE_CURVE = "("
    COMPOSITE_CURVE-CLOSED_CURVE ","
    [ COMPOSITE_CURVE-PARAM_RANGE ] ","
    COMPOSITE_CURVE-SEGMENTS ","
    COMPOSITE_CURVE-SELF_INTERSECT ","
    COMPOSITE_CURVE-SENSES ","
    COMPOSITE_CURVE-TRANSITIONS ")."

COMPOSITE_CURVE_ON_SURFACE-BASIS_SURFACE = entity_reference | SURFACE.
COMPOSITE_CURVE_ON_SURFACE-CLOSED_CURVE = logical.
COMPOSITE_CURVE_ON_SURFACE-SEGMENTS = list.
COMPOSITE_CURVE_ON_SURFACE-SENSES = list.
COMPOSITE_CURVE_ON_SURFACE-TRANSITIONS = list.
COMPOSITE_CURVE_ON_SURFACE = "("
    COMPOSITE_CURVE_ON_SURFACE-BASIS_SURFACE ","
    COMPOSITE_CURVE_ON_SURFACE-CLOSED_CURVE ","
    COMPOSITE_CURVE_ON_SURFACE-SEGMENTS ","
    COMPOSITE_CURVE_ON_SURFACE-SENSES ","
    COMPOSITE_CURVE_ON_SURFACE-TRANSITIONS ")."

CONIC = PARABOLA
    HYPERBOLA
    ELLIPSE
    CIRCLE.

CONICAL_SURFACE-POSITION__ = entity_reference | AXIS2_PLACEMENT.
CONICAL_SURFACE-RADIUS = real.
CONICAL_SURFACE-SEMI_ANGLE = real.
CONICAL_SURFACE = "("
    CONICAL_SURFACE-POSITION__ ","
    CONICAL_SURFACE-RADIUS ","
    CONISCAL_SURFACE-SEMI_ANGLE ")".

CURVE = OFFSET_CURVE
    CURVE_ON_SURFACE
    BOUNDED_CURVE
    CONIC
    LINE.

CURVE_BOUNDED_SURFACE-BASIS_SURFACE = entity_reference | SURFACE.
CURVE_BOUNDED_SURFACE-BOUNDARIES = list.
CURVE_BOUNDED_SURFACE-OUTER_PRESENT = logical.
CURVE_BOUNDED_SURFACE = "("
CURVE_BOUNDED_SURFACE-BASIS_SURFACE ",";
CURVE_BOUNDED_SURFACE-BOUNDARIES ",";
CURVE_BOUNDED_SURFACE-OUTER_PRESENT ").

CURVE_LOGICAL_STRUCTURE-CURVE_ELEMENT = logical.
CURVE_LOGICAL_STRUCTURE-FLAG = logical.
CURVE_LOGICAL_STRUCTURE = "(";
CURVE_LOGICAL_STRUCTURE-CURVE_ELEMENT ",";
CURVE_LOGICAL_STRUCTURE-FLAG ")."

CURVE_ON_SURFACE = COMPOSITE_CURVE_ON_SURFACE
| INTERSECTION_CURVE
| SURFACE_CURVE
| PCURVE.

CYLINDRICAL_SURFACE-POSITION__ = entity_reference | AXIS2_PLACEMENT.
CYLINDRICAL_SURFACE-RADIUS = real.
CYLINDRICAL_SURFACE = "(";
CYLINDRICAL_SURFACE-POSITION__ ",";
CYLINDRICAL_SURFACE-RADIUS ")."

D2_OFFSET_CURVE-BASIS_CURVE = integer.
D2_OFFSET_CURVE-DISTANCE = real.
D2_OFFSET_CURVE-SELF_INTERSECT = logical.
D2_OFFSET_CURVE = "(";
D2_OFFSET_CURVE-BASIS_CURVE ",";
D2_OFFSET_CURVE-DISTANCE ",";
D2_OFFSET_CURVE-SELF_INTERSECT ")".

D1_OFFSET_CURVE-BASIS_CURVE = entity_reference | CURVE.
D1_OFFSET_CURVE-DISTANCE = real.
D1_OFFSET_CURVE-REF_DIRECTION = entity_reference | DIRECTION.
D1_OFFSET_CURVE-SELF_INTERSECT = entity_reference | DIRECTION.
D1_OFFSET_CURVE = "(";
D1_OFFSET_CURVE-BASIS_CURVE ",";
D1_OFFSET_CURVE-DISTANCE ",";
D1_OFFSET_CURVE-REF_DIRECTION ",";
D1_OFFSET_CURVE-SELF_INTERSECT ")".

DIRECTION = THREE_SPACE_DIRECTION
| TWO_SPACE_DIRECTION.

EDGE-EDGE_CURVE = entity_reference | CURVE_LOGICAL_STRUCTURE.
EDGE-EDGE_END = entity_reference | VERTEX.
EDGE-EDGE_START = entity_reference | VERTEX.
EDGE = "(";
EDGE-EDGE_CURVE ",";
EDGE-EDGE_END ","
EDGE-EDGE_START
"

EDGE_LOGICAL_STRUCTURE-EDGE_ELEMENT = entity_reference | EDGE .
EDGE_LOGICAL_STRUCTURE-FLAG = logical .
EDGE_LOGICAL_STRUCTURE = "(" EDGE_LOGICAL_STRUCTURE-EDGE_ELEMENT "," EDGE_LOGICAL_STRUCTURE-FLAG ")" .

EDGE_LOOP-LOOP_EDGES = list .
EDGE_LOOP = "(" EDGE_LOOP-LOOP_EDGES ")" .

ELEMENTARY_SURFACE = TOROIDAL_SURFACE
SPHERICAL_SURFACE
CONICAL_SURFACE
CYLINDRICAL_SURFACE
PLANE .

ELLIPSE-POSITION = entity_reference | AXIS2_PLACEMENT .
ELLIPSE-SEMI_AXIS_1 = real .
ELLIPSE-SEMI_AXIS_2 = real .
ELLIPSE = "(" ELLIPSE-POSITION "," ELLIPSE-SEMI_AXIS_1 "," ELLIPSE-SEMI_AXIS_2 ")" .

FACE-BOUNDS = list .
FACE-FACE_SURFACE = entity_reference | SURFACE_LOGICAL_STRUCTURE .
FACE-OUTER_BOUND = entity_reference | LOOP_LOGICAL_STRUCTURE .

FACE_LOGICAL_STRUCTURE-FACE_ELEMENT = entity_reference | FACE .
FACE_LOGICAL_STRUCTURE-FLAG = logical .
FACE_LOGICAL_STRUCTURE = "(" FACE_LOGICAL_STRUCTURE-FACE_ELEMENT "," FACE_LOGICAL_STRUCTURE-FLAG ")" .

GEOMETRY = SURFACE
CURVE
TRANSFORMATION
AXIS_PLACEMENT
VECTOR
POINT .
N279

ISO TC184/SC4/WG1
ANNEX B (Normative)
(Document 4.2.1 Draft Paper)

September 1983

HYPERBOLA-POSITION__ = entity_reference | AXIS2_PLACEMENT.
HYPERBOLA-SEMI_AXIS = real.
HYPERBOLA-SEMI_IMAG_AXIS = real.
HYPERBOLA = "(" HYPERBOLA-POSITION__ ","
HYPERBOLA-SEMI_AXIS ","
HYPERBOLA-SEMI_IMAG_AXIS ")".

INTERSECTION_CURVE-BASIS_CURVE = entity_reference | CURVE.
INTERSECTION_CURVE-MASTER_REPRESENTATION = "string".
INTERSECTION_CURVE-PCURVE_S1 = entity_reference | PCURVE.
INTERSECTION_CURVE-PCURVE_S2 = entity_reference | PCURVE.
INTERSECTION_CURVE-SELF_INTERSECT = logical.
INTERSECTION_CURVE-SURFACE_S1 = entity_reference | SURFACE.
INTERSECTION_CURVE-SURFACE_S2 = entity_reference | SURFACE.
INTERSECTION_CURVE = "(" INTERSECTION_CURVE-BASIS_CURVE ","
[ INTERSECTION_CURVE-MASTER_REPRESENTATION ] ","
[ INTERSECTION_CURVE-PCURVE_S1 ] ","
[ INTERSECTION_CURVE-PCURVE_S2 ] ","
INTERSECTION_CURVE-SELF_INTERSECT ","
[ INTERSECTION_CURVE-SURFACE_S1 ] ","
[ INTERSECTION_CURVE-SURFACE_S2 ] ")".

LINE-DIR = entity_reference | DIRECTION.
LINE-PNT = entity_reference | CARTESIAN_POINT.
LINE = "(" LINE-DIR ","
LINE-PNT ")".

LINE_SEGMENT-FIRST_POINT = entity_reference | CARTESIAN_POINT.
LINE_SEGMENT-LAST_POINT = entity_reference | CARTESIAN_POINT.
LINE_SEGMENT = "(" LINE_SEGMENT-FIRST_POINT ","
LINE_SEGMENT-LAST_POINT ")".

LOOP = POLY_LOOP
| EDGE_LOOP
| VERTEX_LOOP.

LOOP_LOGICAL_STRUCTURE-FLAG = logical.
LOOP_LOGICAL_STRUCTURE-LOOP_ELEMENT = entity_reference | LOOP.
LOOP_LOGICAL_STRUCTURE = "(" LOOP_LOGICAL_STRUCTURE-FLAG ","
LOOP_LOGICAL_STRUCTURE-LOOP_ELEMENT ")".

OFFSET_CURVE = D1_OFFSET_CURVE
| D2_OFFSET_CURVE.
OFFSET_SURFACE-BASIS_SURFACE = entity_reference | SURFACE.
OFFSET_SURFACE-DISTANCE = real.
OFFSET_SURFACE-SELF_INTERSECT = logical.
OFFSET_SURFACE = "("
OFFSET_SURFACE-BASIS_SURFACE ","
OFFSET_SURFACE-DISTANCE ","
OFFSET_SURFACE-SELF_INTERSECT ")".

OPEN_SHELL-SHELL_BOUNDARY = list.
OPEN_SHELL = "("( OPEN_SHELL-SHELL_BOUNDARY ")" ")".

PARABOLA-FOCAL_DIST = real.
PARABOLA-POSITION__ = entity_reference | AXIS2_PLACEMENT.
PARABOLA = "("( PARABOLA-FOCAL_DIST ","
PARABOLA-POSITION__ ")".

PATH-OPEN_EDGE_LIST = list.
PATH = "("( PATH-OPEN_EDGE_LIST ")" ")".

PCURVE-BASIS_CURVE = entity_reference | CURVE.
PCURVE-BASIS_SURFACE = entity_reference | SURFACE.
PCURVE = "("( PCURVE-BASIS_CURVE ","
PCURVE-BASIS_SURFACE ")".

PLANE-POSITION__ = entity_reference | AXIS2_PLACEMENT.
PLANE = "("( PLANE-POSITION__ ")".

POINT = POINT_ON_SURFACE
| POINT_ON_CURVE
| CARTESIAN_POINT.

POINT_ON_CURVE-BASIS_CURVE = entity_reference | CURVE.
POINT_ON_CURVE-POINT_PARAMETER = real.
POINT_ON_CURVE = "("( POINT_ON_CURVE-BASIS_CURVE ","
POINT_ON_CURVE-POINT_PARAMETER ")".

POINT_ON_SURFACE-BASIS_SURFACE = entity_reference | SURFACE.
POINT_ON_SURFACE-POINT_PARAMETER_1 = real.
POINT_ON_SURFACE-POINT_PARAMETER_2 = real.
POINT_ON_SURFACE = "("
    POINT_ON_SURFACE-BASIS_SURFACE ",",
    POINT_ON_SURFACE-POINT_PARAMETER_1 ",",
    POINT_ON_SURFACE-POINT_PARAMETER_2 ")".

POLY_LOOP-POLYGON = list .
POLY_LOOP = "("
    POLY_LOOP-POLYGON
)" .

RECTANGULAR_COMPOSITE_SURFACE-N_U = integer .
RECTANGULAR_COMPOSITE_SURFACE-N_V = integer .
RECTANGULAR_COMPOSITE_SURFACE-SURFACES = list .
RECTANGULAR_COMPOSITE_SURFACE-U_SENSES = list .
RECTANGULAR_COMPOSITE_SURFACE-V_SENSES = list .
RECTANGULAR_COMPOSITE_SURFACE = "("
    RECTANGULAR_COMPOSITE_SURFACE-N_U ",",
    RECTANGULAR_COMPOSITE_SURFACE-N_V ",",
    RECTANGULAR_COMPOSITE_SURFACE-SURFACES ",",
    RECTANGULAR_COMPOSITE_SURFACE-U_SENSES ",",
    RECTANGULAR_COMPOSITE_SURFACE-V_SENSES ")" .

RECTANGULAR_TRIMMED_SURFACE-BASIS_SURFACE = integer .
RECTANGULAR_TRIMMED_SURFACE-UMAX = real .
RECTANGULAR_TRIMMED_SURFACE-UMIN = real .
RECTANGULAR_TRIMMED_SURFACE-VMAX = real .
RECTANGULAR_TRIMMED_SURFACE-VMIN = real .
RECTANGULAR_TRIMMED_SURFACE = "("
    RECTANGULAR_TRIMMED_SURFACE-BASIS_SURFACE ",",
    RECTANGULAR_TRIMMED_SURFACE-UMAX ",",
    RECTANGULAR_TRIMMED_SURFACE-UMIN ",",
    RECTANGULAR_TRIMMED_SURFACE-VMAX ",",
    RECTANGULAR_TRIMMED_SURFACE-VMIN ")" .

REGION-OUTER_REGION_BOUNDARY = entity_reference | SHELL_LOGICAL_STRUCTURE .
REGION-REGION_BOUNDARIES = list .
REGION = "("
    [ REGION-OUTER_REGION_BOUNDARY ] ",",
    REGION-REGION_BOUNDARIES
)" .

SHELL = CLOSED_SHELL
        OPEN_SHELL
        WIRE_SHELL
        VERTEX_SHELL .

SELECT_FACE_OR_SUBFACE = |
    FACE
    SUBFACE .
SHELL_LOGICAL_STRUCTURE-FLAG = logical.
SHELL_LOGICAL_STRUCTURE-SHELL_ELEMENT = entity_reference | SHELL.
SHELL_LOGICAL_STRUCTURE = "("
                   SHELL_LOGICAL_STRUCTURE-FLAG ","
                   SHELL_LOGICAL_STRUCTURE-SHELL_ELEMENT ")").

SPHERICAL_SURFACE-POSITION__ = entity_reference | AXIS2_PLACEMENT.
SPHERICAL_SURFACE-RADIUS = real.
SPHERICAL_SURFACE = "("
                   SPHERICAL_SURFACE-POSITION__ ","
                   SPHERICAL_SURFACE-RADIUS ")").

SUBFACE-BOUNDS = list.
SUBFACE-OUTER_BOUND = entity_reference | LOOP_LOGICAL_STRUCTURE.
SUBFACE-TRIMMING = entity_reference | SELECT_FACE_OR_SUBFACE.
SUBFACE = "("
          SUBFACE-BOUNDS ","
          SUBFACE-OUTER_BOUND ","
          SUBFACE-TRIMMING ")").

SURFACE = RECTANGULAR_COMPOSITE_SURFACE
        OFFSET_SURFACE
        BOUNDED_SURFACE
        SWEPt_SURFACE
        ELEMENTARY_SURFACE.

SURFACE_CURVE-CURVE_1 = entity_reference | CURVE.
SURFACE_CURVE-MASTER = "." string ".".
SURFACE_CURVE-PCURVE_S1 = entity_reference | PCURVE.
SURFACE_CURVE-SURFACE_1 = entity_reference | SURFACE.
SURFACE_CURVE = "("
                  SURFACE_CURVE-CURVE_1 ","
                  [ SURFACE_CURVE-MASTER ] ","
                  [ SURFACE_CURVE-PCURVE_S1 ] ","
                  SURFACE_CURVE-SURFACE_1 ")").

SURFACE_LOGICAL_STRUCTURE-FLAG = logical.
SURFACE_LOGICAL_STRUCTURE-SURFACE_ELEMENT = logical.
SURFACE_LOGICAL_STRUCTURE = "("
                      SURFACE_LOGICAL_STRUCTURE-FLAG ","
                      SURFACE_LOGICAL_STRUCTURE-SURFACE_ELEMENT ")").

SURFACE_OF_LINEAR_EXTRUSION-AXIS = entity_reference | DIRECTION.
SURFACE_OF_LINEAR_EXTRUSION-EXTRUDED_CURVE = entity_reference | CURVE.
SURFACE_OF_LINEAR_EXTRUSION = "("
                       SURFACE_OF_LINEAR_EXTRUSION-AXIS ","
                       SURFACE_OF_LINEAR_EXTRUSION-EXTRUDED_CURVE ")").
**SURFACE_OF_REVOLUTION-AXIS** = entity_reference | DIRECTION.
**SURFACE_OF_REVOLUTION-AXIS_POINT** = entity_reference | CARTESIAN_POINT.
**SURFACE_OF_REVOLUTION-REVOLVED_CURVE** = entity_reference | CURVE.
**SURFACE_OF_REVOLUTION** = "(
  **SURFACE_OF_REVOLUTION-AXIS** ",
  **SURFACE_OF_REVOLUTION-AXIS_POINT** ",
  **SURFACE_OF_REVOLUTION-REVOLVED_CURVE"
)".

**SWEPT_SURFACE** = **SURFACE_OF_LINEAR_EXTRUSION**
| **SURFACE_OF_REVOLUTION**.

**THREE_SPACE_DIRECTION-X** = real.
**THREE_SPACE_DIRECTION-Y** = real.
**THREE_SPACE_DIRECTION-Z** = real.
**THREE_SPACE_DIRECTION** = "(
  **THREE_SPACE_DIRECTION-X** ",
  **THREE_SPACE_DIRECTION-Y** ",
  **THREE_SPACE_DIRECTION-Z"
)".

**TOPOLOGY** = REGION
| SHELL
| SURFACE
| FACE
| LOOP
| PATH
| EDGE
| VERTEX.

**TOROIDAL_SURFACE-MAJOR_RADIUS** = real.
**TOROIDAL_SURFACE-MINOR_RADIUS** = real.
**TOROIDAL_SURFACE-POSITION** = entity_reference | AXIS2_PLACEMENT.
**TOROIDAL_SURFACE** = "(
  **TOROIDAL_SURFACE-MAJOR_RADIUS** ",
  **TOROIDAL_SURFACE-MINOR_RADIUS** ",
  **TOROIDAL_SURFACE-POSITION**
)".

**TRANSFORMATION-AXIS1** = entity_reference | DIRECTION.
**TRANSFORMATION-AXIS2** = entity_reference | THREE_SPACE_DIRECTION.
**TRANSFORMATION-AXIS3** = entity_reference | THREE_SPACE_DIRECTION.
**TRANSFORMATION-LOCAL_ORIGIN** = entity_reference | CARTESIAN_POINT.
**TRANSFORMATION-SCALE** = real.
**TRANSFORMATION** = "(
  [ **TRANSFORMATION-AXIS1** ] ",
  [ **TRANSFORMATION-AXIS2** ] ",
  [ **TRANSFORMATION-AXIS3** ] ",
  [ **TRANSFORMATION-LOCAL_ORIGIN** ] ",
  [ **TRANSFORMATION-SCALE** ]
)"."
TRIMMED_CURVE-BASIS_CURVE = integer.
TRIMMED_CURVE-PARAMETER_1 = real.
TRIMMED_CURVE-PARAMETER_2 = real.
TRIMMED_CURVE-POINT_1 = entity_reference | CARTESIAN_POINT.
TRIMMED_CURVE-POINT_2 = entity_reference | CARTESIAN_POINT.
TRIMMED_CURVE-SENSE = logical.
TRIMMED_CURVE = "("
  TRIMMED_CURVE-BASIS_CURVE ","
  [ TRIMMED_CURVE-PARAMETER_1 ] ","
  [ TRIMMED_CURVE-PARAMETER_2 ] ","
  [ TRIMMED_CURVE-POINT_1 ] ","
  [ TRIMMED_CURVE-POINT_2 ] ","
  TRIMMED_CURVE-SENSE
)").

TWO_SPACE_DIRECTION-X = real.
TWO_SPACE_DIRECTION-Y = real.
TWO_SPACE_DIRECTION = "("
  TWO_SPACE_DIRECTION-X ","
  TWO_SPACE_DIRECTION-Y
)").

VECTOR = VECTOR_WITH_MAGNITUDE |
  DIRECTION .

VECTOR_WITH_MAGNITUDE-MAGNITUDE = real.
VECTOR_WITH_MAGNITUDE-ORIENTATION = entity_reference | DIRECTION.
VECTOR_WITH_MAGNITUDE = "("
  VECTOR_WITH_MAGNITUDE-MAGNITUDE ","
  VECTOR_WITH_MAGNITUDE-ORIENTATION
")" .

VERTEX-VERTEX_POINT = entity_reference | POINT.
VERTEX = "("
  [ VERTEX-VERTEX_POINT ]
)").

VERTEX_LOOP-LOOP_VERTEX = entity_reference | VERTEX.
VERTEX_LOOP = "("
  VERTEX_LOOP-LOOP_VERTEX
)").

VERTEX_SHELL-VERTEX_SHELL_BOUNDARY = entity_reference | VERTEX_LOOP.
VERTEX_SHELL = "("
  VERTEX_SHELL-VERTEX_SHELL_BOUNDARY
)").

WIRE_SHELL-WIRE_SHELL_BOUNDARY = list.
WIRE_SHELL = "("
  WIRE_SHELL-WIRE_SHELL_BOUNDARY
)").

ENTITY_DEF = 

C-12
<table>
<thead>
<tr>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIS1 PLACEMENT</td>
<td>AXIS1_PLACEMENT</td>
</tr>
<tr>
<td>AXIS2 PLACEMENT</td>
<td>AXIS2_PLACEMENT</td>
</tr>
<tr>
<td>BEZIER CURVE</td>
<td>BEZIER_CURVE</td>
</tr>
<tr>
<td>BEZIER SURFACE</td>
<td>BEZIER_SURFACE</td>
</tr>
<tr>
<td>B SPLINE CURVE</td>
<td>B_SPLINE_CURVE</td>
</tr>
<tr>
<td>B SPLINE SURFACE</td>
<td>B_SPLINE_SURFACE</td>
</tr>
<tr>
<td>CARTESIAN THREE COORDINATE</td>
<td>CARTESIAN_THREE_COORDINATE</td>
</tr>
<tr>
<td>CARTESIAN TWO COORDINATE</td>
<td>CARTESIAN_TWO_COORDINATE</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>CIRCLE</td>
</tr>
<tr>
<td>CLOSED SHELL</td>
<td>CLOSED_SHELL</td>
</tr>
<tr>
<td>COMPOSITE CURVE</td>
<td>COMPOSITE_CURVE</td>
</tr>
<tr>
<td>COMPOSITE CURVE ON SURFACE</td>
<td>COMPOSITE_CURVE_ON_SURFACE</td>
</tr>
<tr>
<td>CONICAL SURFACE</td>
<td>CONICAL_SURFACE</td>
</tr>
<tr>
<td>CURVE BOUNDED SURFACE</td>
<td>CURVE_BOUNDED_SURFACE</td>
</tr>
<tr>
<td>CURVE LOGICAL STRUCTURE</td>
<td>CURVE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>CYLINDRICAL SURFACE</td>
<td>CYLINDRICAL_SURFACE</td>
</tr>
<tr>
<td>D2 OFFSET CURVE</td>
<td>D2_OFFSET_CURVE</td>
</tr>
<tr>
<td>D3 OFFSET CURVE</td>
<td>D3_OFFSET_CURVE</td>
</tr>
<tr>
<td>EDGE</td>
<td>EDGE</td>
</tr>
<tr>
<td>EDGE LOGICAL STRUCTURE</td>
<td>EDGE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>EDGE LOOP</td>
<td>EDGE_LOOP</td>
</tr>
<tr>
<td>ELLIPSE</td>
<td>ELLIPSE</td>
</tr>
<tr>
<td>FACE</td>
<td>FACE</td>
</tr>
<tr>
<td>FACE LOGICAL STRUCTURE</td>
<td>FACE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>HYPERBOLA</td>
<td>HYPERBOLA</td>
</tr>
<tr>
<td>INTERSECTION CURVE</td>
<td>INTERSECTION_CURVE</td>
</tr>
<tr>
<td>LINE</td>
<td>LINE</td>
</tr>
<tr>
<td>LINE SEGMENT</td>
<td>LINE_SEGMENT</td>
</tr>
<tr>
<td>LOOP LOGICAL STRUCTURE</td>
<td>LOOP_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>OFFSET SURFACE</td>
<td>OFFSET_SURFACE</td>
</tr>
<tr>
<td>OPEN SHELL</td>
<td>OPEN_SHELL</td>
</tr>
<tr>
<td>PARABOLA</td>
<td>PARABOLA</td>
</tr>
<tr>
<td>PATH</td>
<td>PATH</td>
</tr>
<tr>
<td>PCURVE</td>
<td>PCURVE</td>
</tr>
<tr>
<td>PLANE</td>
<td>PLANE</td>
</tr>
<tr>
<td>POINT ON CURVE</td>
<td>POINT_ON_CURVE</td>
</tr>
<tr>
<td>POINT ON SURFACE</td>
<td>POINT_ON_SURFACE</td>
</tr>
<tr>
<td>POLY LOOP</td>
<td>POLY_LOOP</td>
</tr>
<tr>
<td>RECTANGULAR COMPOSITE SURFACE</td>
<td>RECTANGULAR_COMPOSITE_SURFAC</td>
</tr>
<tr>
<td>RECTANGULAR TRIMMED SURFACE</td>
<td>RECTANGULAR_TRIMMED_SURFAC</td>
</tr>
<tr>
<td>REGION</td>
<td>REGION</td>
</tr>
<tr>
<td>SHELL LOGICAL STRUCTURE</td>
<td>SHELL_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>SPHERICAL SURFACE</td>
<td>SPHERICAL_SURFACE</td>
</tr>
<tr>
<td>SURFACE</td>
<td>SURFACE</td>
</tr>
<tr>
<td>SURFACE CURVE</td>
<td>SURFACE_CURVE</td>
</tr>
<tr>
<td>SURFACE LOGICAL STRUCTURE</td>
<td>SURFACE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>SURFACE OF LINEAR EXTRUSION</td>
<td>SURFACE_OF_LINEAR_EXTRUSION</td>
</tr>
<tr>
<td>SURFACE OF REVOLUTION</td>
<td>SURFACE_OF_REVOLUTION</td>
</tr>
<tr>
<td>THREE SPACE DIRECTION</td>
<td>THREE_SPACE_DIREC</td>
</tr>
<tr>
<td>TOROIDAL SURFACE</td>
<td>TOROIDAL_SURFACE</td>
</tr>
<tr>
<td>TRANSFORMATION</td>
<td>TRANSFORMATION</td>
</tr>
<tr>
<td>TRIMMED CURVE</td>
<td>TRIMMED_CURVE</td>
</tr>
<tr>
<td>TWO SPACE DIRECTION</td>
<td>TWO_SPACE_DIRECTION</td>
</tr>
</tbody>
</table>
VECTOR_WITH_MAGNITUDEalias VECTOR_WITH_MAGNITUDE  
VERTEXalias VERTEX  
VERTEX_LOOPalias VERTEX_LOOP  
VERTEX_SHELLalias VERTEX_SHELL  
WIRE_SHELLalias WIRE_SHELL  

AXIS1_PLACEMENTalias = "AXIS1_PLACEMENT" | "AX1".  
AXIS2_PLACEMENTalias = "AXIS2_PLACEMENT" | "AX2".  
BEZIER_CURVEalias = "BEZIER_CURVE" | "CVZ".  
BEZIER_SURFACEalias = "BEZIER_SURFACE" | "SFZ".  
B_SPLINE_CURVEalias = "B_SPLINE_CURVE" | "CVB".  
B_SPLINE_SURFACEalias = "B_SPLINE_SURFACE" | "SFB".  
CARTESIAN_THREE_COORDINATEalias = "CARTESIAN_THREE_COORDINATE" | "PT3".  
CARTESIAN_TWO_COORDINATEalias = "CARTESIAN_TWO_COORDINATE" | "PT2".  
CIRCLEalias = "CIRCLE" | "CI".  
CLOSED_SHELLalias = "CLOSED_SHELL" | "SHC".  
COMPOSITE_CURVEalias = "COMPOSITE_CURVE" | "CVC".  
COMPOSITE_CURVE_ON_SURFACEalias = "COMPOSITE_CURVE_ON_SURFACE" | "CVCS".  
CONICAL_SURFACEalias = "CONICAL_SURFACE" | "SFCN".  
CURVE_BOUNDED_SURFACEalias = "CURVE_BOUNDED_SURFACE" | "SFCD".  
CURVE_LOGICAL_STRUCTUREalias = "CURVE_LOGICAL_STRUCTURE" | "CVD".  
CYLINDRICAL_SURFACEalias = "CYLINDRICAL_SURFACE" | "SFCY".  
D2_OFFSET_CURVEalias = "D2_OFFSET_CURVE" | "CVF2".  
D3_OFFSET_CURVEalias = "D3_OFFSET_CURVE" | "CVF3".  
EDGEalias = "EDGE" | "ED".  
EDGE_LOGICAL_STRUCTUREalias = "EDGE_LOGICAL_STRUCTURE" | "EGD".  
EDGE_LOOPalias = "EDGE_LOOP" | "LPE".  
ELLIPSEalias = "ELLIPSE" | "EL".  
FACEalias = "FACE" | "FL".  
FACE_LOGICAL_STRUCTUREalias = "FACE_LOGICAL_STRUCTURE" | "FCD".  
HYPERBOL alias = "HYPERBOLA" | "HP".  
INTERSECTION_CURVEalias = "INTERSECTION_CURVE" | "CVX".  
LINEalias = "LINE" | "LN".  
LINE_SEGMENTalias = "LINE_SEGMENT" | "LS".  
LOOP_LOGICAL_STRUCTUREalias = "LOOP_LOGICAL_STRUCTURE" | "LPD".  
OFFSET_SURFACEalias = "OFFSET_SURFACE" | "SFF".  
OPEN_SHELLalias = "OPEN_SHELL" | "SHO".  
PARABOL alias = "PARABOLA" | "PB".  
PATHalias = "PATH" | "PA".  
PCURVEalias = "PCURVE" | "CVP".  
PLANEalias = "PLANE" | "PL".  
POINT_ON_CURVEalias = "POINT_ON_CURVE" | "PTC".  
POINT_ON_SURFACEalias = "POINT_ON_SURFACE" | "PTS".  
POLY_LOOPalias = "POLY_LOOP" | "LP".  
RECTANGULAR_COMPOSITE_SURFACEalias = "RECTANGULAR_COMPOSITE_SURFACE" | "SFRC".  
RECTANGULAR_TRIMMED_SURFACEalias = "RECTANGULAR_TRIMMED_SURFACE" | "SFR".  
REGIONalias = "REGION" | "RG".  
SHELL_LOGICAL_STRUCTUREalias = "SHELL_LOGICAL_STRUCTURE" | "SHD".  
SPHERICAL_SURFACEalias = "SPHERICAL_SURFACE" | "SFSP".  
SURFACE_CURVEalias = "SURFACE_CURVE" | "CVS".  
SURFACE_LOGICAL_STRUCTUREalias = "SURFACE_LOGICAL_STRUCTURE" | "SFD".  

C-14
SURFACE_OF_LINEAR_EXTRUSION = "SURFACE_OF_LINEAR_EXTRUSION" | "SFLE".
SURFACE_OF_REVOLUTION = "SURFACE_OF_REVOLUTION" | "SFRV".
THREE_SPACE_DIRECTION = "THREE_SPACE_DIRECTION" | "D31".
TOROIDAL_SURFACE = "TOROIDAL_SURFACE" | "SFT".
TRANSFORMATION = "TRANSFORMATION" | "TM".
TRIMMED_CURVE = "TRIMMED_CURVE" | "CVT".
TWO_SPACE_DIRECTION = "TWO_SPACE_DIRECTION" | "DI2".
VECTOR_WITH_MAGNITUDE = "VECTOR_WITH_MAGNITUDE" | "VCM".
VERTEX = "VERTEX" | "VX".
VERTEX_LOOP = "VERTEX_LOOP" | "LPV".
VERTEX_SHELL = "VERTEX_SHELL" | "SHV".
WIRE_SHELL = "WIRE_SHELL" | "SHW".

ENTITY_OCCURRENCE = entity_identifier "=" ENTITY_DEF ";".
APPENDIX D - PHYSICAL EXCHANGE MEDIUM FORMATS

D.0 PHYSICAL EXCHANGE MEDIUM FORMATS

D.1 MAGNETIC TAPE

When a STEP file is to be transported via magnetic tape, the magnetic tape should have the following characteristics:

- STANDARD
- UNLABELED
- 9 TRACK
- 1600 BPI
- PHYSICAL BLOCKSIZE = 800 BYTES
- 1/2-INCH TAPE
- DATA ENCODING ACCORDING TO ISO 6937 (See Appendix B for more detail).

The above characteristics are in accord with ISO 3788 "Information Processing Information Interchange Via Magnetic Tape".

D.2 TELECOMMUNICATION FORMAT

To be determined.
Title: Mapping from EXPRESS to Physical File

Owner: Jeff Altemueller

Date: September 1988

Corresponding ISO Document Number: N280
MAPPING FROM EXPRESS TO PHYSICAL FILE STRUCTURE

DOCUMENT NUMBER: 4.2.2

TITLE: MAPPING FROM EXPRESS TO PHYSICAL FILE STRUCTURE

ABSTRACT:

This document presents the mapping from the Data Specification Language EXPRESS to the STEP Physical File Structure.

KEY WORDS:

DATE: September 1988
OWNER: Jeff Altemueller
PHONE NUMBER: (314) 234-5272
ISO REPRESENTATIVE: Jeff Altemueller
STATUS: Working Paper
# Table of Contents

1.0 INTRODUCTION ............................................. 1-1

2.0 MAPPING FROM EXPRESS TO PHYSICAL FILE STRUCTURE.
   2.1 Quick Reference Mapping Table ................. 2-1

3.0 DATA TYPE MAPPINGS .......................................... 3-1
   3.1 Base Data Types ........................................ 3-1
      3.1.1 Integer ..................................... 3-1
      3.1.2 String ..................................... 3-1
      3.1.3 Logical .................................. 3-1
      3.1.4 Real ..................................... 3-1
      3.1.5 Mapping Example ......................... 3-2
   3.2 List ............................................... 3-3
   3.3 Array ............................................ 3-4
   3.4 Set .............................................. 3-6
   3.5 Defined Type .................................... 3-7

4.0 ENTITY MAPPING ............................................... 4-1
   4.1 Entity With Explicit Attributes ................ 4-1
   4.2 Entity With Optional Attributes .............. 4-4
   4.3 Entity With Derived Attributes ............... 4-6
   4.4 Entity With Entity Attributes ............... 4-8
      4.4.1 Entities As Internal Attributes ........ 4-8
      4.4.2 Entities As External Attributes ....... 4-9
      4.4.3 Entities as Dynamic Attributes ....... 4-11
   4.5 Entities as Subtypes ............................. 4-12
   4.6 Entities as Supertypes ........................... 4-14
   4.7 Entity With WHERE Rules ....................... 4-14

5.0 SCHEMA .................................................. 7-1

6.0 REMARKS ................................................ 8-1
CHAPTER 1 - INTRODUCTION

1.0 INTRODUCTION

This document identifies the process through which Product Data entities defined in EXPRESS are mapped to the physical file format of PDES/STEP. It is assumed that the audience of this document is familiar with the EXPRESS language (refer to STEP Document 4.1.1, The Use Of Formal Language) and the PDES/STEP exchange format (refer to STEP Document 4.2.1, The STEP File Structure).

This document presents the simpler concept of data type mapping first and then discusses the more complex concept of entity mapping. Mapping is performed strictly from EXPRESS to the physical file structure, specifically to the Data section of the exchange format. Therefore, the headings used throughout this document utilize the same terms used within STEP Document 4.1.1, The EXPRESS User Guide. As a convenience to the reader of this document, EXPRESS constructs are written in bold upper case letters, while the file structure constructs are written in bold, lower case, italicized letters.

The EXPRESS language is an extremely powerful data modeling language that contains declarative statements, executable statements, and algorithms. For the purposes of exchange, only the declarative statements are actually mapped to the exchange format. The executable statements and algorithms of EXPRESS are used to clarify and refine the definition of and to impose constraints upon the attributes of Product Data entities. The executable statements are not mapped to the exchange format.

Several mapping examples have been provided throughout this document. Blank spaces and blank lines have been inserted into the exchange format of several examples to aid readability. The reader should note that these blank spaces and blank lines need not appear in an actual exchange format file.
CHAPTER 2 - MAPPING FROM EXPRESS TO PHYSICAL FILE STRUCTURE

2.0 MAPPING FROM EXPRESS TO PHYSICAL FILE STRUCTURE

Table 2-1 provides a quick reference of the mappings between the Conceptual Schema and the Physical File Structure. Specifically, these are the mappings from the elements or constructs of EXPRESS onto their counterparts in the DATA section of the STEP file structure. This table has been organized in alphabetical order while the content of this document has been organized in increasing order of complexity. A more detailed discussion of the individual mappings is presented in chapters 3 and 4 of this document.

2.1 QUICK REFERENCE MAPPING TABLE

Table 2-1 Conceptual Schema to Physical File Structure Mapping

<table>
<thead>
<tr>
<th>EXPRESS CONSTRUCTS</th>
<th>DATA SECTION CONSTRUCTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>list</td>
</tr>
<tr>
<td>DERIVED ATTRIBUTE</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>ENTITY</td>
<td>entity or embedded entity</td>
</tr>
<tr>
<td>ENTITY AS ATTRIBUTE</td>
<td>embedded entity or entity reference</td>
</tr>
<tr>
<td>ENTITY AS SUPERTYPES</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>ENUMERATION</td>
<td>enumeration</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>INTEGER</td>
<td>integer</td>
</tr>
<tr>
<td>LIST</td>
<td>list</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>enumeration</td>
</tr>
<tr>
<td>MAP</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>REAL</td>
<td>real</td>
</tr>
<tr>
<td>REMARKS</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>RULE</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>SELECT</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>SET</td>
<td>list</td>
</tr>
<tr>
<td>STRING</td>
<td>string</td>
</tr>
<tr>
<td>TYPE</td>
<td>NO INSTANTIATION</td>
</tr>
<tr>
<td>WHERE RULES</td>
<td>NO INSTANTIATION</td>
</tr>
</tbody>
</table>
CHAPTER 3 - DATA TYPE MAPPINGS

3.0 DATA TYPE MAPPINGS

3.1 BASE DATA TYPES

3.1.1 INTEGER

The EXPRESS construct of INTEGER will map syntactically to the DATA section as an integer data type. Any restrictions on the domain of the data type will be validated at a level higher than the file structure. See the mapping example in Section 3.1.5.

3.1.2 STRING

The EXPRESS construct of STRING will map syntactically to the DATA section as a string data type. Any restrictions on the domain of the data type will be validated at a level higher than the file structure. See the mapping example in Section 3.1.5.

3.1.3 LOGICAL

The EXPRESS construct of LOGICAL will map syntactically to the DATA section as an enumeration data type. See the mapping example in Section 3.1.5.

The EXPRESS construct of LOGICAL is treated as a predefined enumerated data type with the values of .T., .F., .U..

3.1.4 REAL

The EXPRESS construct of REAL will map syntactically to the DATA section as a real data type. Any restrictions on the domain of the data type will be validated at a higher level than the file structure. See the mapping example in Section 3.1.5.
3.1.5 MAPPING EXAMPLE

Entity Definition in EXPRESS

ENTITY WIDGET;
  i1: INTEGER(2); <-------------------------- A
  i2: INTEGER; <-------------------------- B
  s1: STRING(3); <-------------------------- C
  s2: STRING; <-------------------------- D
  1: LOGICAL; <-------------------------- E
  r1: REAL(4); <-------------------------- F
  r2: REAL; <-------------------------- G
END_ENTITY;

Sample Entity Occurrence in DATA Section

@12=WIDGET(99,99999,'ABC','ABCDEFG','.T.,9.0,1.2345);

A i1 had a value of 99 in this entity occurrence. The number 2 in parenthesis indicates the number of decimal digits allowed in values for this attribute. In this case, the value falls within the specified range (-99..99) for this attribute. The value is syntactically and semantically legal.

B i2 had a value of 99999 in this entity occurrence.

C s1 had a value of 'ABC' in this entity occurrence. This value falls within the range (3 characters) specified for this attribute. The value is syntactically and semantically legal.

D s2 had a value of 'ABCDEFG' in this entity occurrence.

E 1 had a value of TRUE in this entity occurrence.

F r1 had a value of 9.0 in this entity occurrence. This value falls within the range (4 decimal digits in the fractional part of a number) specified for this attribute. The value is syntactically and semantically legal.

G r2 had a value of 1.2345 in this entity occurrence.
3.2 LIST

The EXPRESS construct of LIST will map syntactically to the DATA section as a list data type. If a list is optional and does not exist, it is treated as a defaulted field in the DATA section (refer to Chapter 7 of STEP Document ANNEX B Normative, The STEP File Structure). If the list is empty, it will appear in the DATA section as an open parenthesis immediately followed by a close parenthesis.

3.2.1 Mapping Example

Entity Definition In EXPRESS

```
ENTITY WIDGET;
  attribute1: LIST [0 : #] OF INTEGER; <-       A
  attribute2: LIST [1 : #] OF INTEGER; <-       B
  attribute3: OPTIONAL LIST [1 : #] OF INTEGER; <- C
  attribute4: REAL;
END_ENTITY;
```

Sample Entity Occurrence In DATA Section

```
@24 = WIDGET( (), (1.2.4), , 2.56);
```

- Attribute 1 was an empty list.
- Attribute 2 contained three elements in this occurrence.
- Attribute 3 did not exist in this occurrence.
3.3 ARRAY

The EXPRESS construct of ARRAY will map syntactically in the DATA section as a List data type. The restrictions on the bounds of the array implied in the Conceptual Schema will be validated at a level higher than that of the file structure. An ARRAY [i:j] where j > i has j-i+1 elements.

3.3.1 Mapping Example

Entity Definition In EXPRESS

ENTITY WIDGET;
  attribute1: ARRAY [-1 : 3] OF INTEGER; <------A
  attribute2: ARRAY [1 : 5] OF OPTIONAL INTEGER; <------B
END_ENTITY;

Sample Entity Occurrence In DATA Section

@30 = WIDGET((1,2,3,4,5) , (1,2,3,,5)) , ((1,2,3),(4,5,6)));

  A   B   C

A attribute1 contained the following values
  attribute1 [-1] = 1
  attribute1 [0] = 2
  attribute1 [1] = 3
  attribute1 [2] = 4
  attribute1 [3] = 5

B attribute2 contained the following values. The meaning of a missing value must be defined in the conceptual schema document.
  attribute2 [1] = 1
  attribute2 [2] = 2
  attribute2 [3] = 3
  attribute2 [4] = MISSING
  attribute2 [5] = 5
Mapping Example (continued)

attribute3 contained the following values:

\[
\begin{align*}
\text{attribute3} \{1,1\} &= 1 \\
\text{attribute3} \{1,2\} &= 2 \\
\text{attribute3} \{1,3\} &= 3 \\
\text{attribute3} \{2,1\} &= 4 \\
\text{attribute3} \{2,2\} &= 5 \\
\text{attribute3} \{2,3\} &= 6
\end{align*}
\]
3.4 SET

The EXPRESS construct of SET will map syntactically in the DATA section as a list data type. The constraint that no two elements of a SET may have the same value will be validated at a higher level than the file structure. The constraint that a SET may not have missing members will be validated at a higher level than the file structure.

3.4.1. Mapping Example

Entity Definition In EXPRESS

```
ENTITY WIDGET;
  a_number: SET OF INTEGER; <-A
END_ENTITY;
```

Sample Occurrence In DATA Section

@22 = WIDGET((0,1,2));

A The attribute a_number was defined by the set of numbers 0, 1, 2 in this occurrence. It is semantically and syntactically correct.

@23 = WIDGET((0,2));

A In this occurrence the attribute is semantically incorrect according to the definition of a SET in EXPRESS. In EXPRESS a SET may not have missing or defaulted members.

@24 = WIDGET((0,0,2));

A In this occurrence the attribute is semantically incorrect according to the definition of a SET in EXPRESS. In EXPRESS a SET may not have duplicate values.
3.5 DEFINED TYPE

The EXPRESS construct of TYPE has no instantiation of its own within the DATA section. The type is treated as the element(s) it defines. For example, if a type is defined as a Real, a real number is put into the DATA section. If a type is defined to be a List, a list is put into the DATA section. A mapping example is provided below. (see 3.6.1) Two other important kinds of defined types, theENUMERATION and the SELECT are discussed separately.

3.5.1 Mapping Example

Entity Definition In EXPRESS

```express
TYPE
  type1 = INTEGER;
  type2 = LIST [1 : 2] of REAL;
END_TYPE;
ENTITY WIDGET;
  attribute1: LOGICAL;                   \ A
  attribute2: TYPE1;                     \ B
  attribute3: TYPE2;                     \ C
END_ENTITY;
```

Sample Occurrence In DATA Section

```express
@14 = WIDGET( .T., 256, (1.0,0.0));
```

3.5.2 ENUMERATION

The EXPRESS construct of ENUMERATION will map syntactically to the DATA section as an enumeration. The actual value in an occurrence of the enumeration will be one of the original enumerated values enclosed by periods.
3.5.2.1 Mapping Example

Entity Definition In EXPRESS

TYPE
  primary_color = ENUMERATION OF (RED, GREEN, BLUE);
END_TYPE;
ENTITY WIDGET;
  color: PRIMARY_COLOR; <- \(A\)
END_ENTITY;

Sample Occurrence In DATA Section

@22 = WIDGET(.RED.);

\(A\) The color attribute of the entity occurrence contained a value of RED.

3.5.3 SELECT

The EXPRESS construct of SELECT has no instantiation of its own within the DATA section. The type is treated as a reference to or an occurrence of one of the entity types in its select-from list. A mapping example is provided below.
3.5.3.1 Mapping Example

TYPE
  AAA = SELECT (XXX, YYY, ZZZ);
END_TYPE

ENTITY WIDGET;
  attr1: LIST [1:#] OF AAA; <--A
END_ENTITY;

Sample Entity Occurrence in DATA Section

@11 = XXX (0.0,0.0,0.0, 1.0,1.0,1.0);
@12 = YYY(0.0,0.0,0.0, 1.0,2.0,0.0, 3.0,2.0,0.0);
@13 = WIDGET((#11,#12,XXX(3.0,2.0,0.0,5.0,4.0,0.0)));

A The attribute attr1 in this entity occurrence had values of a reference to a XXX, a reference to an YYY, and an embedded XXX entity occurrence.
CHAPTER 4 - ENTITY MAPPING

4.0 ENTITY MAPPING

The EXPRESS construct of ENTITY will map syntactically to the DATA section as an entity. Due to the possible complexity of entity definitions in EXPRESS, a number of mapping examples are provided. These examples will address the following entity definitions:

- Entities defined with simple, explicit attributes
- Entities defined with optional attributes
- Entities defined with derived attributes
- Entities defined with other entities as attributes
- Entities defined with WHERE Rules

4.1 ENTITY WITH EXPLICIT ATTRIBUTES

Entities with explicit attributes map in a straightforward manner to the representative entity occurrences in the DATA section of the exchange format file structure. The mapping of entities with explicit attributes is illustrated in the example below and described on the following page.
4.1.1 Mapping Example

Entity Definition In EXPRESS

TYPE
  primary_color_abbreviation = (R,G,B);
END_TYPE;

ENTITY WIDGET;
  <attribute1: INTEGER; <attribute2: STRING; <attribute3: LOGICAL; <attribute4: REAL; <attribute5: LIST [1 : 2] of LOGICAL; <attribute6: ARRAY [-1:3] of INTEGER; <attribute7: PRIMARY_COLOR_ABBREVIATION; END_ENTITY

Sample Entity Occurrence In DATA Section

@1 = WIDGET( 1, 'A', .T., 1.0, (.T...F.), (1.0,1.2,3), .R.);
The EXPRESS ENTITY-NAME "WIDGET" is mapped to the entity-type keyword of the DATA section entity.

NOTE: In order to optimize file size yet ensure human readability, it is expected that a table will be created to relate the actual EXPRESS ENTITY-NAME to an abbreviated exchange format entity-type keyword.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>attribute 1 had a value of 1 in this entity occurrence.</td>
</tr>
<tr>
<td>C</td>
<td>attribute 2 had a value of 'A' in this entity occurrence.</td>
</tr>
<tr>
<td>D</td>
<td>attribute 3 had a value of .T. in this entity occurrence.</td>
</tr>
<tr>
<td>E</td>
<td>attribute 4 had a value of 1.0 in this entity occurrence.</td>
</tr>
<tr>
<td>F</td>
<td>attribute 5 was a list of logicals in this entity occurrence. The list values were:</td>
</tr>
<tr>
<td>G</td>
<td>attribute 6 was an array of integers in this entity occurrence. The array values were:</td>
</tr>
<tr>
<td>H</td>
<td>Attribute 7 was an enumeration. The entity occurrence contained a value of R.</td>
</tr>
</tbody>
</table>

The attribute order of EXPRESS entity definitions is interpreted such that the first entity attribute below the EXPRESS ENTITY-NAME is considered to be the first entity attribute. The second entity attribute below the EXPRESS ENTITY-NAME is considered to be the second attribute, and so on.

The EXPRESS entity attributes are mapped one-to-one in the order indicated above onto DATA section entity occurrences. The attribute order of DATA section entity occurrences is defined so that the first data value that follows sequentially from the entity-type keyword is considered to be the value of the first attribute. The second value that follows the entity-type keyword is considered to be the value of the second attribute, and so on.

The basic data types of EXPRESS (e.g. INTEGER, REAL, etc.) are always assumed to be INTERNAL attributes to the entities in which they appear. It is not permitted to refer to a simple data type in the file structure. Therefore all references in the file structure must refer to other entities.
4.2 ENTITY WITH OPTIONAL ATTRIBUTES

Entities with explicit but optional attributes (assuming the optional attribute value is supplied) follow the same mapping algorithm as entities with explicit and required attributes. (Refer to Section 4.1, "Entity With Explicit Attributes".) When the optional value is not supplied in an entity occurrence, the delimiter that would have appeared (e.g., "," or "") is still placed into the entity occurrence. The semantics of missing attribute values must be defined by the Application Area that was responsible for defining the entity.

4.2.1 Mapping Example

Entity Definition In EXPRESS

ENTITY XXX;
  attribute1: REAL;
  attribute2: REAL;
END_ENTITY;

ENTITY YYY;
  attribute1: OPTIONAL LOGICAL;
  attribute2: EXTERNAL POINT;
  attribute3: EXTERNAL POINT;
  attribute4: OPTIONAL INTEGER;
  attribute5: OPTIONAL REAL;
END_ENTITY;

Sample Entity Occurrences In DATA Section

@1=XXX(1.0,2.0);
@2=XXX(3.0,4.0);

@22=YYY(#1,#2,#4,);
The EXPRESS ENTITY-NAME "YYY" is mapped to the *entity-type keyword* of the DATA Section entity.

attribute 1 does not have a value in this entity occurrence.

attribute 2 is a reference to the XXX entity with entity identifier 2.

attribute 3 is a reference to the XXX entity with entity identifier 1.

attribute 4 does not have a value in this entity occurrence.

attribute 5 does not have a value in this entity occurrence.
4.3 ENTITY WITH DERIVED ATTRIBUTES

Derived attributes contain information about an entity in addition to containing those attributes that define the entity explicitly and completely. The derived attributes carry redundant information since they can be calculated from the explicit attributes.

Derived attributes do not map to the DATA section: only explicit attributes are mapped. In the example below, note that the derived attributes (attrib5 and attrib4) do not appear in the entity occurrence.

4.3.1 Mapping Example

Entity Definition In EXPRESS

ENTITY XXX; <----------------------------------- A
  p0: YYY; <----------------------------------- B
  p1: YYY; <----------------------------------- C
  p2: YYY; <----------------------------------- D
  DERIVE 
    attrib5 = FUNC_NORMAL (P0,P1,P2); <-- E
    attrib4 = FUNC_DIAMETER (PO,PI,P2);<------ F
END_ENTITY;

Sample Entity Occurrences In DATA Section

@9 = YYY( 0.0, 0.0, 0.0);
@10 = YYY( 1.0, 2.0, 3.0);
@11 = INT( 4.0, 5.0, 6.0);
@12 = XXX( #9, #10, #11);

A B C D
(A) The EXPRESS ENTITY-NAME "XXX" is mapped to the entity-type keyword of the DATA Section entity.

(B) p0 is a reference to the YYY entity with an entity identifier of 9.

(C) p1 is a reference to the YYY entity with an entity identifier of 10.

(D) p3 is a reference to the YYY entity with an entity identifier of 11.

(E) attrib5 does not map to the entity occurrence.

(F) attrib4 does not map to the entity occurrence.
4.4 ENTITY WITH ENTITY ATTRIBUTES

When an entity is specified as an attribute of a second (parent) entity, the first (child) entity will be mapped to the DATA section as either an entity reference or an embedded entity. The exact mapping is controlled by the presence or absence of specific keywords within the attribute declaration: INTERNAL and EXTERNAL.

If the keyword INTERNAL is present, the entity will map to an embedded entity. If the keyword EXTERNAL is present, the entity will map to an entity reference. If neither keyword is present, or if the keyword DYNAMIC appears, the entity may legally map to either an embedded entity or an entity reference.

4.4.1 Entities As Internal Attributes

When the keyword INTERNAL is present, an entity declared as an attribute will map to an embedded entity. The syntax of an embedded entity can be found in STEP Document 4.2.1, The STEP File Structure. If the child entity is not a SUPERTYPE, then the entity type keyword is optional. Since the embedded entity must always be of the same type, the entity type keyword is not required for unambiguous processing of the exchange file. If the child entity is a SUPERTYPE, then the entity type keyword is required because of the select from functionality of EXPRESS's SUPERTYPE construct.

The significance of an embedded entity is that the sending CAD system does not intend for the receiving CAD system to create the embedded entity as an independent entity within the database of the receiving system. The embedded entity is present only for the purposes of constructing the parent entity. If the parent entity cannot be constructed within the receiving database, the embedded entity should not be constructed.

4.4.1.1 Mapping Example

Entity Definition In EXPRESS

ENTITY YYY;
x: REAL;
y: REAL;
z: REAL;
END_ENTITY;

ENTITY XXX; <--------- (A)
p0: INTERNAL YYY ; <------ (B)
p1: INTERNAL YYY ; <--------- (C)
END_ENTITY;
Sample Entity Occurrence In DATA Section

@30 = XXX( YYY( 0.0, 1.0, 2.0 ), YYY( 1.0, 2.0, 3.0 ) );

(A) The EXPRESS ENTITY-NAME "XXX" is mapped to the entity-type keyword of the DATA Section entity.
(B) p0 is mapped to an embedded YYY entity in this entity occurrence.
(C) p1 is mapped to an embedded YYY entity in this entity occurrence.

Another Sample Entity Occurrence In DATA Section

@31 = XXX( ( 0.0, 1.0, 2.0 ), ( 1.0, 2.0, 3.0 ) );

(D) The EXPRESS ENTITY-NAME "XXX" is mapped to the entity-type keyword of the DATA Section entity.
(E) p0 is mapped to an embedded YYY entity in this entity occurrence.
(F) p1 is mapped to an embedded YYY entity in this entity occurrence.

4.4.2 Entities As External Attributes

When the keyword EXTERNAL is present, an entity declared as an attribute will map to an entity reference. The syntax of an entity reference can be found in STEP Document Normative ANNEX B, "The STEP File Structure". If the keyword EXTERNAL is used, it is illegal for an entity attribute to be embedded in a parent entity.

The significance of an entity reference is that the sending CAD system intends for the receiving CAD system to create the referred entity as an independent, sharable entity within the database of the receiving system.
4.4.2.1 Mapping Example

Entity Definition In EXPRESS

```express
ENTITY YYY;
x: REAL;
y: REAL;
z: REAL;
END_ENTITY;

ENTITY XXX;
  p0: EXTERNAL YYY;
  p1: EXTERNAL YYY;
END_ENTITY;
```

Sample Entity Occurrences In DATA Section

@9 = YYY(4.0, 5.0, 6.0);
@10 = YYY(1.0, 2.0, 3.0);
@11 = XXX(#9, #10);

- **A** The EXPRESS ENTITY-NAME "XXX" is mapped to the entity-type keyword of the DATA Section entity.
- **B** p0 is a reference to a YYY entity with an entity identifier of 9.
- **C** p1 is a reference to a YYY entity with an entity identifier of 10.
4.4.3 Entities as Dynamic Attributes

When the keyword DYNAMIC is present or when neither the keywords INTERNAL nor EXTERNAL is present, an entity declared as an attribute may legally map to an embedded entity or an entity reference. The syntax of embedded entities and entity references can be found in STEP Document Normative ANNEX B, "The STEP File Structure".

If the entity that has been declared as an attribute is not a SUPERTYPE, then the entity type keyword is optional. Since the embedded entity must always be of the same type, the entity type keyword is not required for unambiguous processing of the exchange file. If the entity that has been declared as an attribute is a SUPERTYPE, then the entity type keyword is required because of the select from functionality of EXPRESS's SUPERTYPE construct.

The significance of the 2 mappings is specified in Chapters 4.4.1 and 4.4.2.

4.4.3.1 Mapping Example

**Entity Definition in EXPRESS**

```
ENTITY YYY;
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;

ENTITY XXX <---------- A
  pO : DYNAMIC YYY <------ B
  pl : DYNAMIC YYY <-------- C
END_ENTITY;
```

**Sample Entity Occurrence in DATA Section**

```
07 = YYY(3., 4., 5.)
09 = XXX(YYY(1., 2., 3.), #7);
```

- The EXPRESS ENTITY NAME "XXX" is mapped to the entity type keyword of the DATA section entity.
- The attribute pO is mapped to an embedded YYY entity in this occurrence.
- The attribute pl is mapped to an entity reference in this entity occurrence.
4.5 ENTITIES AS SUBTYPES

When an EXPRESS entity is labeled as a SUBTYPE of some other entity(s) it will inherit the attributes of the SUPERTYPE entity(s). The order of the attribute inheritance is as follows:

0 All inherited attributes will appear sequentially prior to the defining attributes of any entity.

0 The attributes of a SUPERTYPE entity are inherited in the order they appear in the SUPERTYPE entity itself.

0 If the SUPERTYPE entity is itself a SUBTYPE of another entity, then the attributes of the higher SUPERTYPE entity are inherited first.

0 When multiple SUPERTYPE entities are listed, the list is processed left to right.

4.5.1 Mapping Example

Entity Definition in EXPRESS

ENTITY AAA SUPERTYPE OF (BBB); ------- A
   tm: EXTERNAL ZZZ; ------------------ B
END_ENTITY;

ENTITY BBB SUBTYPE OF (AAA) SUPERTYPE OF (XXX); C
   p0: EXTERNAL YYY; ---------------- D
   pl: EXTERNAL YYY; ---------------- E
END_ENTITY;

ENTITY CCC SUPERTYPE OF (XXX); ------- F
   parameterization : REAL; ------------ G
END_ENTITY;

ENTITY XXX SUBTYPE OF (BBB, CCC); ------- H
   midpt: EXTERNAL YYY; -------------- I
END_ENTITY;
Sample Entity Occurrence in DATA Section

@1 = ZZZ(1., 0., 0., 0., 1., 0., 0., 0., 1., 0.);
@2 = YYY(1.0, 2.0, 0.0)
@3 = YYY(2.0, 2.0, 0.0)
@4 = YYY(1.5, 2.5, 0.0)
@5 = XXX(#1, #2, #3, 1.0, #4);

Since entity AAA is a SUPERTYPE, it will not map to the DATA Section of the exchange format file.

The attribute tm will map to the DATA Section as an inherited attribute in an entity that is directly or indirectly subtyped to the AAA entity. In this case, tm is represented by the ZZZ entity with the entity identifier of "@1".

Since entity BBB is a SUPERTYPE, it will not map to the DATA Section of the exchange format file.

The attribute p0 will map to the DATA Section as an inherited attribute in an entity that is directly or indirectly subtyped to the AAA entity. In this case, p0 is represented by the YYY entity with the entity identifier "@2".

The attribute p1 will map to the DATA Section as an inherited attribute in an entity that is directly or indirectly subtyped to the AAA entity. In this case, p1 is represented by the YYY entity with the entity identifier "@3".

Since entity CCC is a SUPERTYPE, it will not map to the DATA Section of the exchange format file.

The attribute parameterization will map to the DATA Section as an inherited attribute in an entity that is directly or indirectly subtyped to the AAA entity. In this case, the parameterization value is 1.0.

The EXPRESS ENTITY NAME "XXX" is mapped to the entity type keyword of the DATA Section entity.

Attribute midpt is represented by the YYY entity with the entity identifier of "@4".
4.6 ENTITIES AS SUPERTYPES

Entities that are declared as supertypes of other entities will not map directly to the DATA Section of the exchange format file. The attributes of those supertype entities will be migrated into the subtyped entities. See Section 4.5 for a detailed description of the attribute migration.
4.7 ENTITY WITH WHERE RULES

Only attributes that are explicitly declared are mapped to the DATA section. Therefore, WHERE rules do not map to the DATA section. In the example below, note that the WHERE rules are not instantiated in the entity occurrence.

The information contained within the WHERE rules can be used to validate the information within the entity occurrence. However, the WHERE rules themselves are not represented in the exchange format file.

4.7.1 Mapping Example

Entity Definition In EXPRESS

ENTITY WIDGET;  <-------------- A
a: REAL; <----------------- B
b: REAL; <----------------- C
WHERE
a ** 2 + b ** 2 + c ** 2 = 3.0; <--- E
END_ENTITY;

Sample Entity Occurrence In DATA Section

@22 = WIDGET( 1.0, 1.0, 2.0);

A B C D

A The EXPRESS ENTITY NAME "WIDGET" is mapped to the entity type keyword of the DATA section entity.
B Attribute a had a value of 1.0 in the entity occurrence.
C Attribute b had a value of 1.0 in the entity occurrence.
D Attribute c had a value of 2.0 in the entity occurrence.
E The WHERE rule did not map to the DATA section. In this case, the entity is syntactically correct, but semantically incorrect.
5.0 SCHEMA

The EXPRESS construct of SCHEMA does not map to the DATA Section.
6.0 REMARKS

Remarks do not map to the DATA section.
Title: Integrated Product Data Semantic Model and Topical Reference Models

Owner: Anthony Day

Date: 31 October 1988

Corresponding ISO Document Number: N288
Section 1: INTEGRATION CORE MODEL

1.1 Purpose

The purpose of the data model described herein is to provide a fundamental data structure for the shape definition of a product item and its various geometric representations. The data structure is independent of the type of product. Whether it be strictly mechanical, electrical, architectural, construction, or of some other type, the model can satisfy the requirements to provide the shape definition and shape geometric representations.

1.2 Scope

The scope of the Integration Core Model evolves as more PDES topical models become ready to be integrated. The initial scope of the model is focused on models that contain shareable information, such as geometry, Product Structure Configuration Management, Form Feature, Tolerance, etc. The model does not contain information that is specific to an application area, such as Mechanical Product, Electric Product, Architectural Product, etc. The scope of the integration core model will expand to contain integration information necessary for the purpose to indicate how the shareable data is used for an application area.

At any point of time, the Integration Core Model represents a consolidated viewpoint of all models that are addressed. Eventually, the Integration Core Model will represent the viewpoint of PDES total scope.

1.3 Fundamental Concepts and Assumptions

The Integration Core Model was based on a fundamental conceptual idea—that is, the elements of shape that are used to reference the form of a real or conceived object are independent from any specific geometric and topological elements (e.g., face, vertex, point, etc.) that are associated with specific rules of a representation method. A clear distinction was made between referenceable shape elements on a real or conceived object and the mechanism used to represent that shape element.

The structure of the Integration Core Model is very extendible. The separation of the referenceable shape elements from their representations allows expansion of additional representations. The establishment of the referenceable shape elements provides a framework for elements of the descriptive identification of the physical form of any product item. It also allows for future expansion when requirements arise.
Section 1: INTEGRATION CORE MODEL

For each identified shape element, relationships were established to all of its known possible representations. The following table was used to capture the relationships and guide the development of the Integration Core Model:

Referenceable Shape Element and Representation Geometric Model Relationship Tables

### Dimensionality-3 Shape Element Representation Table

<table>
<thead>
<tr>
<th>Model</th>
<th>Object</th>
<th>Voidless Volume</th>
<th>Object Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brep</td>
<td>Entire model</td>
<td>Shell</td>
<td>Entire model</td>
</tr>
<tr>
<td>Facetted Brep</td>
<td>Entire model</td>
<td>Shell</td>
<td>Entire model</td>
</tr>
<tr>
<td>CSG Solid</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>CSG Primitive</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>Surface</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>Half Space</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>Wireframe</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>SOR (Complete)</td>
<td>Entire model</td>
<td>Loop log. struct.</td>
<td>Entire model</td>
</tr>
<tr>
<td>SOR (Partial)</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>SOLE</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
<tr>
<td>Geom. Assembly</td>
<td>Entire model</td>
<td>Entire model</td>
<td>Entire model</td>
</tr>
</tbody>
</table>

### Dimensionality-2 Shape Element Representation Table

<table>
<thead>
<tr>
<th>Model</th>
<th>Area</th>
<th>Maximal Area</th>
<th>Nonmaximal Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brep</td>
<td>Face</td>
<td></td>
<td>Subface</td>
</tr>
<tr>
<td>Facetted Brep</td>
<td>Face</td>
<td></td>
<td>Subface</td>
</tr>
<tr>
<td>CSG Solid</td>
<td>Face</td>
<td></td>
<td>Subface</td>
</tr>
<tr>
<td>CSG Primitive</td>
<td>Face</td>
<td></td>
<td>Subface</td>
</tr>
<tr>
<td>Surface</td>
<td>Face</td>
<td></td>
<td>Subface</td>
</tr>
<tr>
<td>Half Space</td>
<td>Loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wireframe</td>
<td>Edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOR (Complete)</td>
<td>Edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOR (Partial)</td>
<td>Edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLE</td>
<td>Edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unstruct. Geom.</td>
<td>Geometry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geom. Assembly</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Edge on subface.
Section 1: INTEGRATION CORE MODEL

<table>
<thead>
<tr>
<th>Model</th>
<th>Seam</th>
<th>Perimeter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brep</td>
<td>Edge</td>
<td>Loop</td>
</tr>
<tr>
<td>Facetted Brep</td>
<td></td>
<td>Polyloop</td>
</tr>
<tr>
<td>CSG Solid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSG Primitive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surface</td>
<td>Edge</td>
<td>Loop</td>
</tr>
<tr>
<td>Half Space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wireframe</td>
<td>Edge</td>
<td>Loop</td>
</tr>
<tr>
<td>SOR (Complete)</td>
<td>Vertex</td>
<td>Vertex</td>
</tr>
<tr>
<td>SOR (Partial)</td>
<td>Edge, Vertex</td>
<td>Loop, Edge</td>
</tr>
<tr>
<td>SOLE</td>
<td>Edge, Vertex</td>
<td>Loop, Edge</td>
</tr>
<tr>
<td>Unstruct. Geom.</td>
<td>Geometry</td>
<td>Geometry</td>
</tr>
<tr>
<td>Geom. Assembly</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dimensionality-0 Shape Element Representation Table**

<table>
<thead>
<tr>
<th>Model</th>
<th>Corner</th>
<th>Boundary Location</th>
<th>Interior Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brep</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>Facetted Brep</td>
<td>Point</td>
<td>Point (2)</td>
<td>Point (2)</td>
</tr>
<tr>
<td>CSG Solid</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>CSG Primitive</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>Surface</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>Half Space</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>Wireframe</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>SOR (Complete)</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>SOR (Partial)</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>SOLE</td>
<td>Vertex</td>
<td>Vertex (1)</td>
<td>Vertex (1)</td>
</tr>
<tr>
<td>Geom. Assembly</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Vertex on nonmax face.
Note 2: Point on nonmax polyloop.
Section 1: INTEGRATION CORE MODEL

Assumptions were made that the PDES Integrated Product Data Model (IPDM) must address the following issues:

(1) PDES must include various types of geometric models, i.e., Boundary Representation, CSG, Surface, Extrusion, Wireframe, Solid of Revolution, Facetted Boundary Representation, Geometry, and a mixture of several representations.

(2) To describe the same physical form/shape of a product item, many possible representations can be and will be used.

(3) It is necessary to identify the correspondences between geometric entities that belong to different geometric models when they represent the same physical form/shape of a product item.

(4) PDES will allow geometry to be exchanged independent of its association with a product item, e.g., standard shape.

1.4 Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this document.

Brep  Boundary Representation
DIM-0  Dimensionality-0
DIM-1  Dimensionality-1
DIM-2  Dimensionality-2
DIM-3  Dimensionality-3
PDES  Product Data Exchange Specification
PI  Product Item
PIV  Product Item Version
PSCM  Product Structure Configuration Management
SE  Shape Element
SML  Structured Modeling Language
SOLE  Solids of Linear Extrusion
SOR  Solids of Revolution
2. INTEGRATION PLANNING MODEL

This section describes the planning model for the Integration model. This planning model serves as a high-level overview of the scope of the model and as a starting point for the development of the detailed reference model.

2.1 Entity Pool

The Integration planning model consists of the following entities, which are defined on subsequent pages:

<table>
<thead>
<tr>
<th>Entity No.</th>
<th>Entity Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT-1</td>
<td>Shape</td>
</tr>
<tr>
<td>INT-2</td>
<td>Shape Element</td>
</tr>
</tbody>
</table>

The Integration planning model also contains the following entities, which are (or need to be) defined in other PDES reference models:

<table>
<thead>
<tr>
<th>Entity No.</th>
<th>Entity Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSCM-1</td>
<td>Product Item</td>
</tr>
<tr>
<td>PSCM-2</td>
<td>Product Item Version</td>
</tr>
<tr>
<td>GEO-84</td>
<td>CSG Primitive Model</td>
</tr>
<tr>
<td>GEO-85</td>
<td>CSG Solid Model</td>
</tr>
<tr>
<td>GEO-86</td>
<td>Facetted Brep Model</td>
</tr>
<tr>
<td>GEO-87</td>
<td>Half Space Model</td>
</tr>
<tr>
<td>GEO-88</td>
<td>Manifold Solid Brep Model</td>
</tr>
<tr>
<td>GEO-106</td>
<td>Solid of Linear Extrusion Model</td>
</tr>
<tr>
<td>GEO-107</td>
<td>Solid of Revolution Model</td>
</tr>
</tbody>
</table>

Surface Model  
Wireframe Model  
Unstructured Geometry Model  
Geometric Assembly Model
Section 1: INTEGRATION CORE MODEL

2.2 Planning Model Diagram

planning model goes here
2.3 **Entity Glossary**

**Entity Name:** CSG Primitive Model  
**Entity Number:** GEO-84  
**Entity Definition:** Defined in the PDES Geometry reference model.  
**Business Rules:** Every "CSG Primitive Model":  
* represents zero, one, or more "Shape Elements".

**Entity Name:** CSG Solid Model  
**Entity Number:** GEO-85  
**Entity Definition:** Defined in the PDES Geometry reference model.  
**Business Rules:** Every "CSG Solid Model":  
* represents zero, one, or more "Shape Elements".

**Entity Name:** Facetted Brep Model  
**Entity Number:** GEO-86  
**Entity Definition:** Defined in the PDES Geometry reference model.  
**Business Rules:** Every "Facetted Brep Model":  
* represents zero, one, or more "Shape Elements".

**Entity Name:** Geometric Assembly Model  
**Entity Number:** none  
**Entity Definition:** Needs to be defined in some other PDES reference model.  
**Business Rules:** Every "Geometric Assembly Model":  
* represents zero, one, or more "Shape Elements".
Section 1: INTEGRATION CORE MODEL

Entity Name: Half Space Model
Entity Number: GEO-87
Entity Definition: Defined in the PDES Geometry reference model.
Business Rules: Every "Half Space Model":
  • represents zero, one, or more "Shape Elements".

Entity Name: Manifold Solid Brep Model
Entity Number: GEO-88
Entity Definition: Defined in the PDES Geometry reference model.
Business Rules: Every "Manifold Solid Brep Model":
  • represents zero, one, or more "Shape Elements".

Entity Name: Product Item
Entity Number: PSCM-1
Entity Definition: Defined in the PDES PSCM reference model.
Business Rules: Every "Product Item":
  • has one or more "Product Item Versions".

Entity Name: Product Item Version
Entity Number: PSCM-2
Entity Definition: Defined in the PDES PSCM reference model.
Business Rules: Every "Product Item Version":
  • is for one, and only one, "Product Item".
  • has a primary shape of zero, one, or more "Shapes".
Section 1: INTEGRATION CORE MODEL

Entity Name: Shape

Entity Number: INT-1

Entity Definition: The size, spatial configuration, and proportions of a real or conceived thing, independent of whether or how it is represented.

Business Rules: Every "Shape":
   - is an aggregation zero, one, or more "Shape Elements".
   - is the primary shape for zero, one, more "Product Item Versions".

Entity Name: Shape Element

Entity Number: INT-2

Entity Definition: A distinguishable aspect of a "Shape".

Business Rules: Every "Shape Element":
   - is part of one, and only one, "Shape".
   - is represented by zero, one, or more "CSG Primitive Models".
   - is represented by zero, one, or more "CSG Solid Models".
   - is represented by zero, one, or more "Facetted Brep Models".
   - is represented by zero, one, or more "Geometric Assembly Models".
   - is represented by zero, one, or more "Half Space Models".
   - is represented by zero, one, or more "Manifold Solid Brep Models".
   - is represented by zero, one, or more "Solid of Linear Extrusion Models".
   - is represented by zero, one, or more "Solid of Revolution Models".
   - is represented by zero, one, or more "Surface Models".
   - is represented by zero, one, or more "Unstructured Geometry Models".
   - is represented by zero, one, or more "Wireframe Models".
Section 1: INTEGRATION CORE MODEL

Entity Name: Solid of Linear Extrusion Model
Entity Number: GEO-106
Entity Definition: Defined in the PDES Geometry reference model.
Business Rules: Every "Solid of Linear Extrusion Model":
• represents zero, one, or more "Shape Elements".

Entity Name: Solid of Revolution Model
Entity Number: GEO-107
Entity Definition: Defined in the PDES Geometry reference model.
Business Rules: Every "Solid of Revolution Model":
• represents zero, one, or more "Shape Elements".

Entity Name: Surface Model
Entity Number: none
Entity Definition: Needs to be defined in some other PDES reference model.
Business Rules: Every "Surface Model":
• represents zero, one, or more "Shape Elements".

Entity Name: Unstructured Geometry Model
Entity Number: none
Entity Definition: Needs to be defined in some other PDES reference model.
Business Rules: Every "Unstructured Geometry Model":
• represents zero, one, or more "Shape Elements".
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Model

Entity Number: none

Entity Definition: Needs to be defined in some other PDES reference model.

Business Rules: Every "Wireframe Model":
• represents zero, one, or more "Shape Elements".
3. INTEGRATION REFERENCE MODEL

This section describes the reference model for the Integration model. This reference model is depicted in 15 individual views:

1. Shape
2. Geometric Models
3. Dimensionality-3
4. Object
5. Voidless Volume
6. Object Assembly
7. Area
8. Maximal Area
9. Nonmaximal Area
10. Seam
11. Perimeter
12. Dimensionality-0
13. Corner
14. Boundary Location
15. Interior Location
3.1 Entity Pool

The Integration reference model consists of the following entities, which are defined on subsequent pages:

<table>
<thead>
<tr>
<th>Entity No.</th>
<th>Entity Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT-1</td>
<td>Shape</td>
</tr>
<tr>
<td>INT-2</td>
<td>Shape Element</td>
</tr>
<tr>
<td>INT-3</td>
<td>Dimensionality 3 Shape Element</td>
</tr>
<tr>
<td>INT-4</td>
<td>Object Shape Element</td>
</tr>
<tr>
<td>INT-5</td>
<td>Voidless Volume Shape Element</td>
</tr>
<tr>
<td>INT-6</td>
<td>Object Assembly Shape Element</td>
</tr>
<tr>
<td>INT-7</td>
<td>Dimensionality 2 Shape Element</td>
</tr>
<tr>
<td>INT-8</td>
<td>Area Shape Element</td>
</tr>
<tr>
<td>INT-9</td>
<td>Maximal Area Shape Element</td>
</tr>
<tr>
<td>INT-10</td>
<td>Nonmaximal Area Shape Element</td>
</tr>
<tr>
<td>INT-11</td>
<td>Zone Shape Element</td>
</tr>
<tr>
<td>INT-13</td>
<td>Dimensionality 1 Shape Element</td>
</tr>
<tr>
<td>INT-14</td>
<td>Seam Shape Element</td>
</tr>
<tr>
<td>INT-15</td>
<td>Edge Shape Element</td>
</tr>
<tr>
<td>INT-16</td>
<td>Subedge Shape Element</td>
</tr>
<tr>
<td>INT-17</td>
<td>Interior Seam Shape Element</td>
</tr>
<tr>
<td>INT-18</td>
<td>Perimeter Shape Element</td>
</tr>
<tr>
<td>INT-19</td>
<td>Dimensionality 0 Shape Element</td>
</tr>
<tr>
<td>INT-20</td>
<td>Corner Shape Element</td>
</tr>
<tr>
<td>INT-21</td>
<td>Boundary Location Shape Element</td>
</tr>
<tr>
<td>INT-22</td>
<td>Interior Location Shape Element</td>
</tr>
<tr>
<td>INT-23</td>
<td>Geometric Model</td>
</tr>
<tr>
<td>INT-24</td>
<td>Pair of Equivalent Geometric Models</td>
</tr>
<tr>
<td>INT-25</td>
<td>Dim 3 Shape Element Representation</td>
</tr>
<tr>
<td>INT-26</td>
<td>CSG Solid Dim 3 SE Rep</td>
</tr>
<tr>
<td>INT-27</td>
<td>Surface Dim 3 SE Rep</td>
</tr>
<tr>
<td>INT-28</td>
<td>Wireframe Dim 3 SE Rep</td>
</tr>
<tr>
<td>INT-29</td>
<td>Unstruct Geometry Dim 3 SE Rep</td>
</tr>
<tr>
<td>INT-30</td>
<td>Object Representation</td>
</tr>
<tr>
<td>INT-31</td>
<td>Brep Object Rep</td>
</tr>
<tr>
<td>INT-32</td>
<td>Facetted Brep Object Rep</td>
</tr>
<tr>
<td>INT-33</td>
<td>Full Rev SOR Object Rep</td>
</tr>
<tr>
<td>INT-34</td>
<td>Voidless Volume Representation</td>
</tr>
<tr>
<td>INT-35</td>
<td>Brep Volume Rep</td>
</tr>
<tr>
<td>INT-36</td>
<td>CSG Primitive Volume Rep</td>
</tr>
<tr>
<td>INT-37</td>
<td>Half Space Volume Rep</td>
</tr>
<tr>
<td>INT-38</td>
<td>Partial Rev SOR Volume Rep</td>
</tr>
<tr>
<td>INT-39</td>
<td>Full Rev SOR Volume Rep</td>
</tr>
</tbody>
</table>

Page D3.1.1-15
## Section 1: INTEGRATION CORE MODEL

| INT-40 | SOLE Volume Rep |
| INT-41 | Object Assembly Representation |
| INT-42 | Area Representation |
| INT-43 | Brep Area Rep |
| INT-44 | Facetted Brep Area Rep |
| INT-45 | Surface Area Rep |
| INT-46 | Partial Rev SOR Max Area Rep |
| INT-47 | SOLE Max Area Rep |
| INT-48 | Wireframe Area Rep |
| INT-49 | SOR Edge Area Rep |
| INT-50 | SOLE Edge Area Rep |
| INT-51 | Unstruct Geometry Area Rep |
| INT-52 | Nonmaximal Area Representation |
| INT-53 | Brep NM Area Rep |
| INT-54 | Facetted Brep NM Area Rep |
| INT-55 | Surface NM Area Rep |
| INT-56 | Partial Rev SOR Subface NM Area Rep |
| INT-57 | SOLE Subface NM Area Rep |
| INT-59 | SOR Edge NM Area Rep |
| INT-60 | SOLE Edge NM Area Rep |
| INT-63 | Seam Representation |
| INT-64 | Brep Seam Rep |
| INT-65 | Surface Seam Rep |
| INT-66 | Wireframe Seam Rep |
| INT-67 | Partial Rev SOR Edge Seam Rep |
| INT-68 | SOLE Edge Seam Rep |
| INT-70 | SOR Vertex Seam Rep |
| INT-71 | SOLE Vertex Seam Rep |
| INT-72 | Unstruct Geometry Seam Rep |
| INT-73 | Perimeter Representation |
| INT-74 | Brep Perimeter Rep |
| INT-75 | Surface Perimeter Rep |
| INT-76 | Wireframe Perimeter Rep |
| INT-77 | Partial Rev SOR Loop Perimeter Rep |
| INT-78 | SOLE Loop Perimeter Rep |
| INT-79 | Facetted Brep Perimeter Rep |
| INT-80 | Full Rev SOR Vertex Perimeter Rep |
| INT-81 | Partial Rev SOR Edge Perimeter Rep |
| INT-82 | SOLE Edge Perimeter Rep |
| INT-83 | Unstruct Geometry Perimeter Rep |
| INT-84 | Dim 0 Shape Element Representation |
| INT-85 | Brep Dim 0 SE Rep |
| INT-86 | Surface Dim 0 SE Rep |
| INT-87 | Wireframe Dim 0 SE Rep |
| INT-88 | SOLE Dim 0 SE Rep |
Section 1: INTEGRATION CORE MODEL

The Integration reference model also contains the following entities, which are (or need to be) defined in other PDES reference models:

<table>
<thead>
<tr>
<th>Entity No.</th>
<th>Entity Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF-001</td>
<td>Form Feature</td>
</tr>
<tr>
<td>TOP-2</td>
<td>Vertex</td>
</tr>
<tr>
<td>TOP-3</td>
<td>Edge</td>
</tr>
<tr>
<td>TOP-4</td>
<td>Loop</td>
</tr>
<tr>
<td>TOP-5</td>
<td>Face</td>
</tr>
<tr>
<td>TOP-6</td>
<td>Shell</td>
</tr>
<tr>
<td>TOP-8</td>
<td>Subface</td>
</tr>
<tr>
<td>TOP-11</td>
<td>Polyloop</td>
</tr>
<tr>
<td>TOP-25</td>
<td>Loop Logical Structure</td>
</tr>
<tr>
<td>GEO-1</td>
<td>Geometry</td>
</tr>
<tr>
<td>GEO-2</td>
<td>Point</td>
</tr>
<tr>
<td>GEO-84</td>
<td>CSG Primitive Model</td>
</tr>
<tr>
<td>GEO-85</td>
<td>CSG Solid Model</td>
</tr>
<tr>
<td>GEO-86</td>
<td>Facetted Brep Model</td>
</tr>
<tr>
<td>GEO-87</td>
<td>Half Space Model</td>
</tr>
<tr>
<td>GEO-88</td>
<td>Manifold Solid Brep Model</td>
</tr>
<tr>
<td>GEO-106</td>
<td>Solid of Linear Extrusion Model</td>
</tr>
<tr>
<td>GEO-107</td>
<td>Solid of Revolution Model</td>
</tr>
<tr>
<td></td>
<td>Surface Model</td>
</tr>
<tr>
<td></td>
<td>Wireframe Model</td>
</tr>
<tr>
<td></td>
<td>Unstructured Geometry Model</td>
</tr>
<tr>
<td></td>
<td>Geometric Assembly Model</td>
</tr>
</tbody>
</table>
### Section 1: INTEGRATION CORE MODEL

<table>
<thead>
<tr>
<th>Product Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSCM-1</td>
<td>Product Item</td>
</tr>
<tr>
<td>PSCM-2</td>
<td>Product Item Version</td>
</tr>
<tr>
<td>PSCM-4</td>
<td>Product Item Version Functional Definition</td>
</tr>
<tr>
<td>PSCM-27</td>
<td>Product Item Version Definition Shape</td>
</tr>
<tr>
<td>FEM-1</td>
<td>FEM</td>
</tr>
<tr>
<td>FEM-2</td>
<td>FEM Product Item Version Definition</td>
</tr>
</tbody>
</table>
Section 1: INTEGRATION CORE MODEL

3.2 Reference Model Diagrams
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL
Section 1: INTEGRATION CORE MODEL

[Diagram of integration core model]
Section 1: INTEGRATION CORE MODEL
3.3 Entity and Attribute Glossary

**Entity Name:** Area Representation

**Entity Number:** INT-42

**Entity Definition:** The symbolic description of an "Area Shape Element" in a "Geometric Model".

**Business Rules:** Every "Area Representation":
- represents one, and only one, "Area Shape Element".
- must be one, and only one, of the following:
  - "Brep Area Rep"
  - "Facetted Brep Area Rep"
  - "SOLE Edge Area Rep"
  - "SOLE Max Area Rep"
  - "SOR Edge Area Rep"
  - "Partial Rev SOR Max Area Rep"
  - "Surface Area Rep"
  - "Unstruct Geometry Area Rep"
  - "Wireframe Area Rep"

**Primary Key Attributes:**

- **Shape ID** (FK)
- **Shape Elem ID** (FK)
- **Area Rep ID**

The means of distinguishing each "Area Representation" from all the others for the same "Area Shape Element".

**Other Attributes:**

**Area Rep Type**

The means of determining whether an "Area Representation" is a(n):
- "Brep Area Rep"
- "Facetted Brep Area Rep"
- "SOLE Edge Area Rep"
- "SOLE Max Area Rep"
- "SOR Edge Area Rep"
- "Partial Rev SOR Max Area Rep"
- "Surface Area Rep"
- "Unstruct Geometry Area Rep", or
- "Wireframe Area Rep"
Section 1: INTEGRATION CORE MODEL

SML:

ENTITY Area-Representation
  KEY
    Area-Shape-Element "is represented by"
    Area-Rep-ID
  ENDK
  Area-Rep-Type NONULL
ENDE

EXPRESS:

ENTITY Area_Representation
  SUPERTYPE OF ( Brep_Area_Rep
    OR Facetted_Brep_Area_Rep
    OR SOLE_Edge_Area_Rep
    OR SOLE_Max_Area_Rep
    OR SOR_Edge_Area_Rep
    OR Partial_Rev_SOR_Max_Area_Rep
    OR Surface_Area_Rep
    OR Unstruct_Geometry_Area_Rep
    OR Wireframe_Area_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Area Shape Element
Entity Number: INT-8
Entity Definition: A "Dimensionality 2 Shape Element" that is arcwise connected and has a uniform underlying mathematical surface.

Business Rules: Every "Area Shape Element":
- is a "Dimensionality 2 Shape Element".
- must be one, and only one, of the following:
  "Maximal Area Shape Element"
  "Nonmaximal Area Shape Element"
- is represented by zero, one, or more "Area Representations".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area SE Type</td>
<td>The means of determining whether an &quot;Area Shape Element&quot; is a(n):</td>
</tr>
<tr>
<td></td>
<td>&quot;Maximal Area Shape Element&quot; or</td>
</tr>
<tr>
<td></td>
<td>&quot;Nonmaximal Area Shape Element&quot;</td>
</tr>
</tbody>
</table>

SML:

```
ENTITY Area-Shape-Element
  CATEGORY BY Dim-2-SE-Type OF Dimensionality-2-Shape-Element
  Area-SE-Type NONULL
ENDE
```

EXpress:

```
ENTITY Area_Shape_Element
  SUPERTYPE OF ( Maximal_Area_Shape_Element
               OR Nonmaximal_Area_Shape-Element )
  SUBTYPE OF ( Dimensionality_2_Shape_Element );
  Representations : LIST OF [ 0 : # ] OF Area_Representation;
END_ENTITY;
```
Section 1: INTEGRATION CORE MODEL

Entity Name: Boundary Location Shape Element

Entity Number: INT-21

Entity Definition: A "Dimensionality 0 Shape Element" that is on an "Edge Shape Element", but that is not an end point.

Business Rules: Every "Boundary Location Shape Element":
- is a "Dimensionality 0 Shape Element".
- is represented by zero, one, or more "Partial Rev SOR Bdry Loc Reps".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Boundary-Location-Shape-Element
   CATEGORY BY Dim-0-SE-Type OF Dimensionality-0-Shape-Element
ENDE

EXPRESSION:

ENTITY Boundary_Location_Shape_Element
   SUBTYPE OF ( Dimensionality_0_Shape_Element );
   Representations : LIST OF [ 0 : # ] OF Partial_Rev_SOR_Bdry_Loc_Rep;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep Area Rep
Entity Number: INT-43
Entity Definition: The use of a "Face" in a "Manifold Solid Brep Model" to represent an "Area Shape Element". In a manifold solid Brep, a face represents a portion of a solid's surface area that (i) has a uniform underlying surface and (ii) is arcwise connected. If there is an adjacent face with the same underlying surface, the area is nonmaximal; otherwise, it is maximal.

Business Rules: Every "Brep Area Rep":
- is an "Area Representation".
- is in one, and only one, "Manifold Solid Brep Model".
- is one, and only one, "Face".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Area Rep ID (FK)

Other Attributes:
- Brep Model ID (FK)
- Face ID (FK)

SML:

ENTITY Brep-Area-Rep
  CATEGORY BY Area-Rep-Type OF Area-Representation
  Face "is used as" NONULL
  Manifold-Solid-Brep-Model "is context of" NONULL
END_ENTITY;

EXPRESSION:

ENTITY Brep_Area_Rep
  SUBTYPE OF ( Area_Representation );
  Definition: Face;
  Context: Manifold_Solid_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep Dim 0 SE Rep

Entity Number: INT-85

Entity Definition: The use of a "Vertex" in a "Manifold Solid Brep Model" to represent a "Dimensionality 0 Shape Element". In a manifold solid Brep, vertices are employed in maximal faces, nonmaximal faces, subfaces, and vertex loops. In the first three cases, the edges that meet at the vertex may have the same underlying curve. So, a vertex may represent any of the three categories of "Dimensionality 0 Shape Element" -- corner, boundary location, or interior location.

Business Rules: Every "Brep Dim 0 SE Rep":
- is a "Dim 0 Shape Element Representation".
- is in one, and only one, "Manifold Solid Brep Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Dim 0 Rep ID (FK)

Other Attributes:
- Brep Model ID (FK)
- Vertex ID (FK)

SML:

ENTITY Brep-Dim-0-SE-Rep
 CATEGORY BY Dim-0-Rep-Type OF Dim-0-Shape-Element-Representation
   Vertex "is used as" NONULL
   Manifold-Solid-Brep-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY Brep_Dim_0_SE_Rep
   SUBTYPE OF ( Dim_0_Shape_Element_Representation );
   Definition : Vertex;
   Context : Manifold_Solid_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep NM Area Rep

Entity Number: INT-53

Entity Definition: The use of a "Subface" in a "Manifold Solid Brep Model" to represent a "Nonmaximal Area Shape Element". In a manifold solid Brep, a subface represents a portion of surface area that is necessarily nonmaximal because the subface is a subset of a face with a uniform underlying surface. (Note: The IPIM does not require that a subface be a proper subset of the containing face, but it seems pointless to have a subface identical to its containing face. For this reason, no provision is made here for representing a "Maximal Area Shape Element" with a subface.)

Business Rules: Every "Brep NM Area Rep":
- is a "Nonmaximal Area Representation".
- is in one, and only one, "Manifold Solid Brep Model".
- is one, and only one, "Subface".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Nonmax Area Rep ID (FK)

Other Attributes:

- Brep Model ID (FK)
- Subface ID (FK)

SML:

ENTITY Brep-NM-Area-Rep
  CATEGORY BY Nonmax-Area-Rep-Type
  OF Nonmaximal-Area-Representation
  Subface "is used as" NONULL
  Manifold-Solid-Brep-Model "is context of" NONULL
ENDE
ENTITY Brep_NM_Area_Rep
  SUBTYPE OF ( Nonmaximal_Area_Representation );
  Definition : Subface;
  Context : Manifold_Solid_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep Object Rep
Entity Number: INT-31

Entity Definition: The use of a "Manifold Solid Brep Model" to represent an "Object Shape Element". A manifold solid Brep represents a volume that has an arcwise-connected interior, but that may include voids.

Business Rules: Every "Brep Object Rep":
- is an "Object Representation".
- is one, and only one, "Manifold Solid Brep Model".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Object Rep ID (FK)

Other Attributes:
- Brep Model ID (FK)

SML:

ENTITY Brep-Object-Rep
    CATEGORY BY Object-Rep-Type OF Object-Representation
    Manifold-Solid-Brep-Model "is used as" NONULL
END_ENTITY

EXPRESS:

ENTITY Brep_Object_Rep
    SUBTYPE OF ( Object_Representation );
    Definition : Manifold_Solid_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep Perimeter Rep

Entity Number: INT-74

Entity Definition: The use of a "Loop" in a "Manifold Solid Brep Model" to represent a "Perimeter Shape Element". In a manifold solid Brep, a loop occurs as a boundary of a maximal face. Hence, a loop may represent the perimeter of an "Area Shape Element".

Business Rules: Every "Brep Perimeter Rep":
• is a "Perimeter Representation".
• is in one, and only one, "Manifold Solid Brep Model".
• is one, and only one, "Loop".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

Other Attributes:
- Brep Model ID (FK)
- Loop ID (FK)

SML:

ENTITY Brep-Perimeter-Rep
   CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
   Loop "is used as" NONULL
   Manifold-Solid-Brep-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY Brep_Perimeter_Rep
   SUBTYPE OF ( Perimeter_Representation );
   Definition : Loop;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep Seam Rep

Entity Number: INT-64

Entity Definition: The use of an "Edge" in a "Manifold Solid Brep Model" to represent a "Seam Shape Element". In a manifold solid Brep, an edge may occur as a boundary element of a maximal or nonmaximal face and/or as an edge of a subface of a maximal or nonmaximal face. Even if the parent face is maximal, the edge may be nonmaximal; i.e., the edge may have an adjacent edge with the same underlying curve. Hence, an edge may represent a full "natural" edge, a portion of a full "natural" edge, or a curve in the interior of a "natural" face. So, an edge may represent any of the three categories of "Seam Shape Element" -- edge, subedge, or interior seam.

Business Rules: Every "Brep Seam Rep":
- is a "Seam Representation".
- is in one, and only one, "Manifold Solid Brep Model".
- is one, and only one, "Edge".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Seam Rep ID (FK)

Other Attributes:

Brep Model ID (FK)
Edge ID (FK)

SML:

ENTITY Brep-Seam-Rep
CATEGORY BY Seam-Rep-Type OF Seam-Representation
Edge "is used as" NONULL
Manifold-Solid-Brep-Model "is context of" NONULL
ENDE
ENTITY Brep_Seam_Rep
   SUBTYPE OF ( Seam_Representation );
   Definition : Edge;
   Context : Manifold_Solid_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Brep Volume Rep

Entity Number: INT-35

Entity Definition: The use of a "Shell" in a "Manifold Solid Brep Model" to represent a "Voidless Volume Shape Element". In a manifold solid Brep, a shell represents a voidless volume. This may be the peripheral volume of the represented solid or a void within the solid.

Business Rules: Every "Brep Volume Rep":
- is a "Voidless Volume Representation".
- is in one, and only one, "Manifold Solid Brep Model".
- is one, and only one, "Shell".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Volume Rep ID (FK)

Other Attributes:

Brep Model ID (FK)
Shell ID (FK)

SML:

ENTITY Brep-Volume-Rep
  CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation
  Shell "is used as" NONULL
  Manifold-Solid-Brep-Model "is context of" NONULL
END_ENTITY;

EXPRESS:

ENTITY Brep_Volume_Rep
  SUBTYPE OF ( Voidless_Volume_Representation );
  Definition : Shell;
  Context : Manifold_Solid_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Corner Representation

Entity Number: INT-91

Entity Definition: The symbolic description of a "Corner Shape Element" in a "Geometric Model".

Business Rules: Every "Corner Representation":
- represents one, and only one, "Corner Shape Element".
- must be one, and only one, of the following:
  "Full Rev SOR Corner Rep"
  "Partial Rev SOR Corner Rep"

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Corner Rep ID

The means of distinguishing each "Corner Representation" from all the others for the same "Corner Shape Element".

Other Attributes:
- Corner Rep Type

The means of determining whether a "Corner Representation" is a(n):
- "Full Rev SOR Corner Rep" or
- "Partial Rev SOR Corner Rep".

SML:

ENTITY Corner-Representation
  KEY Corner-Shape-Element "is represented by" Corner-Rep-ID
ENDK
  Corner-Rep-Type NONULL
ENDE

EXPRESS:

ENTITY Corner_Representation
  SUPERTYPE OF ( Full_Rev_SOR_Corner_Rep
                     OR Partial_Rev_SOR_Corner_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

**Entity Name:** Corner Shape Element

**Entity Number:** INT-20

**Entity Definition:** A "Dimensionality 0 Shape Element" that is the intersection of three or more "Area Shape Elements" or that is the apex of a cone-like shape.

**Business Rules:** Every "Corner Shape Element":
- is a "Dimensionality 0 Shape Element".
- is represented by zero, one, or more "Corner Representations".

**Primary Key Attributes:**
- Shape ID (FK)
- Shape Elem ID (FK)

**Other Attributes:** none

**SML:**

```
ENTITY Corner-Shape-Element
    CATEGORY BY Dim-0-SE-Type OF Dimensionality-0-Shape-Element
END_ENTITY;
```

**EXPRESS:**

```
ENTITY Corner_Shape_Element
    SUBTYPE OF ( Dimensionality_0_Shape_Element );
    Representations : LIST OF [ 0 : # ] OF Corner_Representation;
END_ENTITY;
```
Section 1: INTEGRATION CORE MODEL

Entity Name: CSG Primitive Model

Entity Number: GEO-84

Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "CSG Primitive Model":
- is a "Geometric Model".
- is used as zero, one, or more "CSG Primitive Volume Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: CSG Primitive Volume Rep
Entity Number: INT-36
Entity Definition: The use of a "CSG Primitive Model" to represent a "Voidless Volume Shape Element".

Business Rules: Every "CSG Primitive Volume Rep":
- is a "Voidless Volume Representation".
- is one, and only one, "CSG Primitive Model".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Volume Rep ID (FK)

Other Attributes:
- CSG Primitive Model ID (FK)

SML:

ENTITY CSG-Primitive-Volume-Rep
   CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation
   CSG-Primitive-Model "is used as" NONULL
END_ENTITY;

EXPRESS:

ENTITY CSG_Primitive_Volume_Rep
   SUBTYPE OF ( Volume_Representation );
   Definition : CSG_Primitive_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: CSG Solid Dim 3 SE Rep

Entity Number: INT-26

Entity Definition: The use of a "CSG Solid Model" to represent a "Dimensionality 3 Shape Element".

Business Rules: Every "CSG Solid Dim 3 SE Rep":
- is a "Dim 3 Shape Element Representation".
- is one, and only one, "CSG Solid Model".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Dim 3 Rep ID (FK)

Other Attributes:
- CSG Solid Model ID (FK)

SML:

ENTITY CSG-Solid-Dim-3-SE-Rep
  CATEGORY BY Dim-3-Rep-Type OF Dim-3-Shape-Element-Representation
  CSG-Solid-Model "is used as" NONULL
END_ENTITY;

EXPRESS:

ENTITY CSG_Solid_Dim_3_SE_Rep
  SUBTYPE OF ( Dim_3_Shape_Element_Representation );
  Definition : CSG_Solid_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: CSG Solid Model

Entity Number: GEO-85

Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "CSG Solid Model":
- is a "Geometric Model".
- is used as zero, one, or more "CSG Solid Dim 3 SE Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Dim 0 Shape Element Representation

Entity Number: INT-84

Entity Definition: The symbolic description of a "Dimensionality 0 Shape Element" in a "Geometric Model".

Business Rules: Every "Dim 0 Shape Element Representation":
- represents one, and only one, "Dimensionality 0 Shape Element".
- must be one, and only one, of the following:
  - "Brep Dim 0 SE Rep"
  - "Facetted Brep Dim 0 SE Rep"
  - "SOLE Dim 0 SE Rep"
  - "Surface Dim 0 SE Rep"
  - "Unstruct Geometry Dim 0 SE Rep"
  - "Wireframe Dim 0 SE Rep"

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Dim 0 Rep ID

The means of distinguishing each "Dim 0 Shape Element Representation" from all the others for the same "Dimensionality 0 Shape Element".

Other Attributes:

Dim 0 Rep Type

The means of determining whether a "Dim 0 Shape Element Representation" is a(n):
- "Brep Dim 0 SE Rep",
- "Facetted Brep Dim 0 SE Rep",
- "SOLE Dim 0 SE Rep",
- "Surface Dim 0 SE Rep",
- "Unstruct Geometry Dim 0 SE Rep", or
- "Wireframe Dim 0 SE Rep".
Section 1: INTEGRATION CORE MODEL

SML:

ENTITY Dim-0-Shape-Element-Representation
  KEY
    Dimensionality-0-Shape-Element "is represented by"
    Dim-0-Rep-ID
  ENDK
  Dim-0-Rep-Type NONULL
ENDE

EXPRESS:

ENTITY Dim_0_Shape_Element_Representation
  SUPERTYPE OF ( Brep_Dim_0_SE_Rep
    OR Facetted_Brep_Dim_0_SE_Rep
    OR SOLE_Dim_0_SE_Rep
    OR Surface_Dim_0_SE_Rep
    OR Unstruct_Geometry_Dim_0_SE_Rep
    OR Wireframe_Dim_0_SE_Rep )
END_ENTITY;
Entity Name: Dim 3 Shape Element Representation

Entity Number: INT-25

Entity Definition: The symbolic description of a "Dimensionality 3 Shape Element" in a "Geometric Model".

Business Rules: Every "Dim 3 Shape Element Representation":
- represents one, and only one, "Dimensionality 3 Shape Element".
- must be one, and only one, of the following:
  "CSG Solid Dim 3 SE Rep"
  "Surface Dim 3 SE Rep"
  "Unstruct Geometry Dim 3 SE Rep"
  "Wireframe Dim 3 SE Rep"

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Dim 3 Rep ID

Primary Key Attributes:
- Dim 3 Rep Type

Other Attributes:
- The means of distinguishing each "Dim 3 Shape Element Representation" from all the others for the same "Dimensionality 3 Shape Element".

SML:

```
ENTITY Dim-3-Shape-Element-Representation
  KEY
    Dimensionality-3-Shape-Element "is represented by"
    Dim-3-Rep-ID
  ENDK
  Dim-3-Rep-Type NONULL
ENDE
```
ENTITY Dim_3_Shape_Element_Representation
  SUPERTYPE OF ( CSG_Solid_Dim_3_SE_Rep
  OR Surface_Dim_3_SE_Rep
  OR Unstruct_Geometry_Dim_3_SE_Rep
  OR Wireframe_Dim_3_SE_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Dimensionality 0 Shape Element

Entity Number: INT-19

Entity Definition: A "Shape Element" that has dimensionality zero, i.e., that is a location on the surface area of a shape.

Business Rules: Every "Dimensionality 0 Shape Element":
- is a "Shape Element".
- must be one, and only one, of the following:
  "Boundary Location Shape Element"
  "Corner Shape Element"
  "Interior Location Shape Element"
- is represented by zero, one, or more "Dim 0 Shape Element Representations".

Primary Key Attributes:
- Shape ID
- Shape Elem ID

Other Attributes:
- Dim 0 SE Type

The means of determining whether a "Dimensionality 0 Shape Element" is a(n):
- "Boundary Location Shape Element",
- "Corner Shape Element", or
- "Interior Location Shape Element".

SML:

ENTITY Dimensionality-0-Shape-Element
  CATEGORY BY Shape-Elem-Type OF Shape-Element
  Dim-0-SE-Type NONULL
ENDE
EXPRESS:

ENTITY Dimensionality_0_Shape_Element
   SUPERTYPE OF ( Corner_Shape_Element
   OR Boundary_Location_Shape_Element
   OR Interior_Location_Shape_Element )
   SUBTYPE OF ( Shape_Element );
   Representations : LIST OF [ 0 : # ] OF
      Dim_0_Shape_Element_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Dimensionality 1 Shape Element

Entity Number: INT-13

Entity Definition: A "Shape Element" that has no discontinuity and has dimensionality one, i.e., that is a path on the surface area of a shape.

Business Rules: Every "Dimensionality 1 Shape Element":
- is a "Shape Element".
- must be one, and only one, of the following:
  "Perimeter Shape Element"
  "Seam Shape Element"

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)

Other Attributes:

Dim 1 SE Type

The means of determining whether a "Dimensionality 1 Shape Element" is a(n):
"Perimeter Shape Element" or "Seam Shape Element".

SML:

ENTITY Dimensionality-1-Shape-Element
  CATEGORY BY Shape-Elem-Type OF Shape-Element
  Dim-1-SE-Type NONULL
END

EXPRESS:

ENTITY Dimensionality_1_Shape_Element
  SUPERTYPE OF ( Seam_Shape_Element
                  OR Perimeter_Shape_Element )
  SUBTYPE OF ( Shape_Element );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Dimensionality 2 Shape Element

Entity Number: INT-7

Entity Definition: A "Shape Element" that has dimensionality two, i.e., that is a portion of the surface area of a shape.

Business Rules: Every "Dimensionality 2 Shape Element":
- is a "Shape Element".
- must be one, and only one, of the following:
  "Area Shape Element"
  "Zone Shape Element"
- is zero or one "Form Feature".
- is used as zero, one, or more "Zone Shape Element Components".

Primary Key Attributes:

- Shape ID
- Shape Elem ID (FK)

Other Attributes:

- Dim 2 SE Type
  The means of determining whether a "Dimensionality 2 Shape Element" is a(n):
  "Area Shape Element" or "Zone Shape Element".

SML:

ENTITY Dimensionality-2-Shape-Element
  CATEGORY BY Shape-Elem-Type OF Shape-Element
  Dim-2-SE-Type NONULL
END_ENTITY

EXPRESS:

ENTITY Dimensionality_2_Shape_Element
  SUPERTYPE OF ( Area_Shape_Element
      OR Zone_Shape_Element )
  SUBTYPE OF ( Shape_Element );
  Form_Feature_Shape_Element : OPTIONAL Form_Feature;
  Use_As_Zone_Comp : LIST OF [ 0 : # ] Zone_Shape_Element_Component;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Dimensionality 3 Shape Element

Entity Number: INT-3

Entity Definition: A "Shape Element" that has dimensionality three, i.e., that is a volume.

Business Rules: Every "Dimensionality 3 Shape Element":
- is a "Shape Element".
- must be one, and only one, of the following: "Object Assembly Shape Element" or "Object Shape Element".
- is represented by zero, one, or more "Dim 3 Shape Element Representations".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)

Other Attributes:
- Dim 3 SE Type: The means of determining whether a "Dimensionality 3 Shape Element" is a(n): "Object Assembly Shape Element" or "Object Shape Element".

SML:

ENTITY Dimensionality-3-Shape-Element
  CATEGORY BY Shape-Elem-Type OF Shape-Element
  Dim-3-SE-Type NONULL
ENDER

EXPRESS:

ENTITY Dimensionality_3_Shape_Element
  SUPERTYPE OF ( Object_Shape_Element
                     OR Object_Assembly_Shape_Element )
  SUBTYPE OF ( Shape_Element );
  Representations : LIST OF [ 0 : # ] OF
                     Dim_3_Shape_Element_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Edge

Entity Number: TOP-3

Entity Definition: Defined in the PDES Topology reference model.

Business Rules: Every "Edge":

- is used as zero, one, or more "Brep Seam Reps".
- is used as zero, one, or more "SOLE Edge Area Reps".
- is used as zero, one, or more "SOLE Edge NM Area Reps".
- is used as zero, one, or more "SOLE Edge Perimeter Reps".
- is used as zero, one, or more "SOLE Edge Seam Reps".
- is used as zero, one, or more "SOR Edge Area Reps".
- is used as zero, one, or more "SOR Edge NM Area Reps".
- is used as zero, one, or more "Partial Rev SOR Edge Perimeter Reps".
- is used as zero, one, or more "Partial Rev SOR Edge Seam Reps".
- is used as zero, one, or more "Surface Seam Reps".
- is used as zero, one, or more "Wireframe Seam Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Edge Shape Element

Entity Number: INT-15

Entity Definition: A "Seam Shape Element" that is the intersection of two "Maximal Area Shape Elements". Due to the continuity requirement on "Dimensionality 1 Shape Elements", a "Seam Shape Element" does not have to be the entire intersection of the pair of "Maximal Area Shape Elements", but it must be a maximal continuous portion of their intersection.

Business Rules: Every "Edge Shape Element" is a "Seam Shape Element".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>FK</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>FK</td>
</tr>
</tbody>
</table>

Other Attributes: none

SML:

ENTITY Edge-Shape-Element
  CATEGORY BY Seam-SE-Type OF Seam-Shape-Element
END ENTITY;

EXPRESS:

ENTITY Edge_Shape_Element
  SUBTYPE OF ( Seam_Shape_Element );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Face

Entity Number: TOP-5

Entity Definition: Defined in the PDES Topology reference model.

Business Rules: Every "Face":
- is the face revolved by zero, one, or more "Solid of Revolution Models".
- is bound by one or more "Loop Logical Structures".
- is used as zero, one, or more "Brep Area Reps".
- is used as zero, one, or more "Facetted Brep Area Reps".
- is used as zero, one, or more "Surface Area Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Facetted Brep Area Rep

Entity Number: INT-44

Entity Definition: The use of a "Face" in a "Facetted Brep Model" to represent an "Area Shape Element".

Business Rules: Every "Facetted Brep Area Rep":
- is an "Area Representation".
- is in one, and only one, "Facetted Brep Model".
- is one, and only one, "Face".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Area Rep ID (FK)

Other Attributes:

- Facetted Brep Model ID (FK)
- Face ID (FK)

SML:

ENTITY Facetted-Brep-Area-Rep
  CATEGORY BY Area-Rep-Type OF Area-Representation
  Face "is used as" NONULL
  Facetted-Brep-Model "is context of" NONULL
ENDE

EXpress:

ENTITY Facetted_Brep_Area_Rep
  SUBTYPE OF ( Area_Representation );
  Definition : Face;
  Context : Facetted_Brep_Model;
END_ENTITY;
### Section 1: INTEGRATION CORE MODEL

**Entity Name:** Facetted Brep Dim 0 SE Rep

**Entity Number:** INT-89

**Entity Definition:** The use of a "Point" in a "Facetted Brep Model" to represent a "Dimensionality 0 Shape Element".

**Business Rules:**
- Every "Facetted Brep Dim 0 SE Rep"
  - is a "Dim 0 Shape Element Representation".
  - is in one, and only one, "Facetted Brep Model".
  - is one, and only one, "Point".

**Primary Key Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td></td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td></td>
</tr>
<tr>
<td>Dim 0 Rep ID</td>
<td></td>
</tr>
</tbody>
</table>

**Other Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facetted Brep Model ID</td>
<td></td>
</tr>
<tr>
<td>Point ID</td>
<td></td>
</tr>
</tbody>
</table>

**SML:**

```xml
ENTITY Facetted-Brep-Dim-0-SE-Rep
  CATEGORY BY Dim-0-Rep-Type OF Dim-0-Shape-Element-Representation
  Point "is used as" NONULL
  Facetted-Brep-Model "is context of" NONULL
ENDE
```

**EXPRESS:**

```xml
ENTITY Facetted_Brep_Dim_0_SE_Rep
  SUBTYPE OF ( Dim_0_Shape_Element_Representation );
  Definition : Point;
  Context : Facetted_Brep_Model;
END_ENTITY;
```
Section 1: INTEGRATION CORE MODEL

Entity Name: Facetted Brep Model
Entity Number: GEO-86
Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "Facetted Brep Model":
- is a "Geometric Model".
- is the context for zero, one, or more "Facetted Brep Area Reps".
- is the context for zero, one, or more "Facetted Brep Dim 0 SE Reps".
- is the context for zero, one, or more "Facetted Brep NM Area Reps".
- is used as zero, one, or more "Facetted Brep Object Reps".
- is the context for zero, one, or more "Facetted Brep Perimeter Reps".
- is the context of zero, one, or more "Facetted Brep Volume Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Facetted Brep NM Area Rep

Entity Number: INT-54

Entity Definition: The use of a "Subface" in a "Facetted Brep Model" to represent a "Nonmaximal Area Shape Element".

Business Rules: Every "Facetted Brep NM Area Rep":
- is a "Nonmaximal Area Representation".
- is in one, and only one, "Facetted Brep Model".
- is one, and only one, "Subface".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Nonmax Area Rep ID (FK)

Other Attributes:

Facetted Brep Model ID (FK)
Subface ID (FK)

SML:

ENTITY Facetted-Brep-NM-Area-Rep
  CATEGORY BY Nonmax-Area-Rep-Type
  OF Nonmaximal-Area-Representation
  Subface "is used as" NONULL
  Facetted-Brep-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY Facetted_Brep_NM_Area_Rep
  SUBTYPE OF ( Nonmaximal_Area_Representation );
  Definition : Subface;
  Context : Facetted_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

**Entity Name:** Facetted Brep Object Rep

**Entity Number:** INT-32

**Entity Definition:** The use of a "Facetted Brep Model" to represent an "Object Shape Element".

**Business Rules:** Every "Facetted Brep Object Rep":
- is an "Object Representation".
- is one, and only one, "Facetted Brep Model".

**Primary Key Attributes:**
- Shape ID (FK)
- Shape Elem ID (FK)
- Object Rep ID (FK)

**Other Attributes:**
- Facetted Brep Model ID (FK)

**SML:**

```xml
ENTITY Facetted-Brep-Object-Rep
    CATEGORY BY Object-Rep-Type OF Object-Representation
    Facetted-Brep-Model "is used as" NONULL
END.
```

**EXPRESS:**

```express
ENTITY Facetted_Brep_Object_Rep
    SUBTYPE OF ( Object_Representation );
    Definition : Facetted_Brep_Model;
END_ENTITY;
```
Section 1: INTEGRATION CORE MODEL

Entity Name: Facetted Brep Perimeter Rep

Entity Number: INT-79

Entity Definition: The use of a "Polyloop" in a "Facetted Brep Model" to represent a "Perimeter Shape Element".

Business Rules: Every "Facetted Brep Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Facetted Brep Model".
- is one, and only one, "Polyloop".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

Other Attributes:

- Facetted Brep Model ID (FK)
- Polyloop ID (FK)

SML:

ENTITY Facetted-Brep-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Polyloop "is used as" NONULL
  Facetted-Brep-Model "is context of" NONULL
END_ENTITY;

EXPRESS:

ENTITY Facetted_Brep_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Polyloop;
  Context : Facetted_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Facetted Brep Volume Rep
Entity Number: INT-104
Entity Definition: The use of a "Shell" in a "Facetted Brep Model" to represent a "Voidless Volume Shape Element".

Business Rules:
Every "Facetted Brep Volume Rep":
• is a "Voidless Volume Representation".
• is in one, and only one, "Facetted Brep Model".
• is one, and only one, "Shell".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Volume Rep ID (FK)

Other Attributes:
- Facetted Brep Model ID (FK)
- Shell ID (FK)

SML:
ENTITY Facetted-Brep-Volume-Rep
  CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation
  Shell "is used as" NONULL
  Facetted-Brep-Model "is context of" NONULL
ENDE

EXPRESS:
ENTITY Facetted_Brep_Volume_Rep
  SUBTYPE OF ( Voidless_Volume_Representation );
  Definition : Shell;
  Context : Facetted_Brep_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: FEM

Entity Number: FEM-1

Entity Definition: Defined in the PDES FEM reference model.

Business Rules: Every "FEM" models zero, one, or more "FEM Product Item Version Definitions".
Section 1: INTEGRATION CORE MODEL

Entity Name: FEM Product Item Version Definition

Entity Number: FEM-2

Entity Definition: Defined in the PDES FEM reference model.

Business Rules: Every "FEM Product Item Version Definition":
   • is modeled by one, and only one, "FEM".
   • models one, and only one, "Product Item Version Functional Definition".
**Section 1: INTEGRATION CORE MODEL**

<table>
<thead>
<tr>
<th><strong>Entity Name:</strong></th>
<th>Form Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity Number:</strong></td>
<td>FF-001</td>
</tr>
<tr>
<td><strong>Entity Definition:</strong></td>
<td>Defined in the PDES Form Feature reference model.</td>
</tr>
<tr>
<td><strong>Business Rules:</strong></td>
<td>Every &quot;Form Feature&quot; is a &quot;Dimensionality 2 Shape Element&quot;.</td>
</tr>
</tbody>
</table>
Section 1: INTEGRATION CORE MODEL

Entity Name: Full Rev SOR Corner Rep

Entity Number: INT-92

Entity Definition: The use of a "Vertex" in a "Solid of Revolution Model" that is revolved through 360° to represent a "Corner Shape Element".

Business Rules: Every "Full Rev SOR Corner Rep":
- is a "Corner Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Corner Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)

SML:

ENTITY Full-Rev-SOR-Corner-Rep
  CATEGORY BY Corner-Rep-Type OF Corner-Representation
  Vertex "is used as" NONULL
  Solid-of-Revolution-Model "is context of" NONULL
END.ENTITY;

EXPRESS:

ENTITY Full_Rev_SOR_Corner_Rep
  SUBTYPE OF (Corner_Representation);
  Definition : Vertex;
  Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Number: INT-100

Entity Definition: The use of a "Vertex" in a "Solid of Revolution Model" that is revolved through 360° to represent an "Interior Location Shape Element".

Business Rules: Every "Full Rev SOR Int Loc Rep":
- is a "Interior Location Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Int Loc Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)

SML:

ENTITY Full-Rev-SOR-Int-Loc-Rep
  CATEGORY BY Int-Loc-Rep-Type OF Interior-Location-Representation
  Vertex "is used as" NONULL
  Solid-of-Revolution-Model "is context of" NONULL
END.

EXPRESSION:

ENTITY Full_Rev_SOR_Int_Loc_Rep
  SUBTYPE OF (Interior_Location_Representation);
  Definition : Vertex;
  Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Full Rev SOR Object Rep

Entity Number: INT-33

Entity Definition: The use of a "Solid of Revolution Model" that is revolved through 360° to represent an "Object Shape Element".

Business Rules: Every "Full Rev SOR Object Rep":
- is an "Object Representation".
- is one, and only one, "Solid of Revolution Model".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Object Rep ID (FK)

Other Attributes:

SOR Model ID (FK)

SML:

ENTITY Full-Rev-SOR-Object-Rep
    CATEGORY BY Object-Rep-Type OF Object-Representation
    Solid-of-Revolution-Model "is used as" NONULL
END

EXPRESS:

ENTITY Full_Rev_SOR_Object_Rep
    SUBTYPE OF ( Object_Representation );
    Definition : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Full Rev SOR Vertex Perimeter Rep

Entity Number: INT-80

Entity Definition: The use of a "Vertex" in a complete "Solid of Revolution Model" that is revolved through 360° to represent a "Perimeter Shape Element".

Business Rules: Every "Full Rev SOR Vertex Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)

SML:

ENTITY Full-Rev-SOR-Vertex-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Vertex "is used as" NONULL
  Solid-of-Revolution-Model "is context of" NONULL
END_ENTITY;

EXPRESS:

ENTITY Full_Rev_SOR_Vertex_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Vertex;
  Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Full Rev SOR Volume Rep

Entity Number: INT-39

Entity Definition: The use of a "Loop Logical Structure" in a "Solid of Revolution Model" that is revolved through 360° to represent a "Voidless Volume Shape Element".

Business Rules: Every "Full Rev SOR Volume Rep":
- is a "Voidless Volume Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Loop Logical Structure".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Volume Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Face ID (FK)
- Loop ID (FK)

SML:

ENTITY Full-Rev-SOR-Volume-Rep
   CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation
   Loop-Logical-Structure "is used as" (1) NONULL
   Solid-of-Revolution-Model "is context of" (1) NONULL
END_ENTITY;

EXPRESS:

ENTITY Full_Rev_SOR_Volume_Rep
   SUBTYPE OF ( Voidless_Volume_Representation );
   Definition : Loop_Logical_Structure;
   Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Geometric Assembly Model

Entity Number: none

Entity Definition: Needs to be defined in some other PDES reference model.

Business Rules: Every "Geometric Assembly Model"
  • is a "Geometric Model".
  • is used as zero, one, or more "Object Assembly Representations".
Section 1: INTEGRATION CORE MODEL

Entity Name: Geometric Model

Entity Number: INT-23

Entity Definition: A formal logical/mathematical representation of a shape.

Business Rules: Every "Geometric Model":
- must be one, and only one, of the following:
  "CSG Primitive Model"
  "CSG Solid Model"
  "Facetted Brep Model"
  "Geometric Assembly Model"
  "Half Space Model"
  "Manifold Solid Brep Model"
  "Solid of Linear Extrusion Model"
  "Solid of Revolution Model"
  "Surface Model"
  "Unstructured Geometry Model"
  "Wireframe Model"
- is one in zero, one, or more "Pairs of Equivalent Geometric Models".
- is the other in zero, one, or more "Pairs of Equivalent Geometric Models".

Primary Key Attributes:

Geom Model ID The means of distinguishing each "Geometric Model" from all others.

Other Attributes:

Geom Model Type The means of determining whether a "Geometric Model" is a(n):
  "CSG Primitive Model",
  "CSG Solid Model",
  "Facetted Brep Model",
  "Geometric Assembly Model",
  "Half Space Model",
  "Manifold Solid Brep Model",
  "Solid of Linear Extrusion Model",
  "Solid of Revolution Model",
  "Surface Model",
  "Unstructured Geometry Model", or
  "Wireframe Model".
SML:

ENTITY Geometric-Model
  KEY
  Geom-Model-ID
  ENDK
  Geom-Model-Type NONULL
  ENDE

EXPRESS:

ENTITY Geometric_Model
  SUPERTYPE OF ( CSG_Primitive_Model
  OR CSG_Solid_Model
  OR Facetted_Brep_Model
  OR Geometric_Assembly_Model
  OR Half_Space_Model
  OR Manifold_Solid_Brep_Model
  OR Solid_of_Linear_Extrusion_Model
  OR Solid_of_Revolution_Model
  OR Surface_Model
  OR Unstructured_Geometry_Model
  OR Wireframe_Model );

END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Geometry

Entity Number: GEO-1

Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "Geometry":
- is used as zero, one, or more "Unstruct Geometry Area Reps".
- is used as zero, one, or more "Unstruct Geometry Dim 0 SE Reps".
- is used as zero, one, or more "Unstruct Geometry NM Area Reps".
- is used as zero, one, or more "Unstruct Geometry Perimeter Reps".
- is used as zero, one, or more "Unstruct Geometry Seam Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Half Space Model

Entity Number: GEO-87

Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "Half Space Model":
- is a "Geometric Model".
- is used as zero, one, or more "Half Space Volume Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Half Space Volume Rep

Entity Number: INT-37

Entity Definition: The use of a "Half Space Model" to represent a "Voidless Volume Shape Element".

Business Rules: Every "Half Space Volume Rep":
- is a "Voidless Volume Representation".
- is one, and only one, "Half Space Model".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td></td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Volume Rep ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half Space Model ID</td>
<td></td>
</tr>
</tbody>
</table>

SML:

ENTITY Half-Space-Volume-Rep
    CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation
    Half-Space-Model "is used as" NONULL
END

EXPRESS:

ENTITY Half_Space_Volume_Rep
    SUBTYPE OF ( Voidless_Volume_Representation );
    Definition : Half_Space_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Interior Location Representation

Entity Number: INT-99

Entity Definition: The symbolic description of an "Interior Location Shape Element" in a "Geometric Model".

Business Rules: Every "Interior Location Representation":
- represents one, and only one, "Interior Location Shape Element".
- must be one, and only one, of the following:
  "Full Rev SOR Int Loc Rep"
  "Partial Rev SOR Int Loc Rep"

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Int Loc Rep ID

The means of distinguishing each "Interior Location Representation" from all the others for the same "Interior Location Shape Element".

Other Attributes:

Int Loc Rep Type

The means of determining whether an "Interior Location Representation" is a(n):
  "Full Rev SOR Int Loc Rep" or
  "Partial Rev SOR Int Loc Rep".

SML:

ENTITY Interior-Location-Representation
  KEY
    Interior-Location-Shape-Element "is represented by"
    Int-Loc-Rep-ID
  ENDK
  Int-Loc-Rep-Type NONULL
ENDE

EXPRESS:

ENTITY Interior_Location_Representation
  SUPERTYPE OF ( Full_Rev_SOR_Int_Loc_Rep
                     OR Partial_Rev_SOR_Int_Loc_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Interior Location Shape Element

Entity Number: INT-22

Entity Definition: A "Dimensionality 0 Shape Element" that is in the interior of an "Area Shape Element".

Business Rules: Every "Interior Location Shape Element":
  - is a "Dimensionality 0 Shape Element".
  - is represented by zero, one, or more "Interior Location Representations".

Primary Key Attributes:

  Shape ID  (FK)
  Shape Elem ID  (FK)

Other Attributes: none

SML:

ENTITY Interior-Location-Shape-Element
  CATEGORY BY Dim-0-SE-Type OF Dimensionality-0-Shape-Element
END

EXPRESSION:

ENTITY Interior_Location_Shape_Element
  SUBTYPE OF (Dimensionality_0_Shape_Element);
  Representations : LIST OF [ 0 : # ] OF Interior_Location_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Interior Seam Shape Element

Entity Number: INT-17

Entity Definition: A "Seam Shape Element" that, with the possible exception of one or both of its end points, lies in the interior of an "Area Shape Element".

Business Rules: Every "Interior Seam Shape Element" is a "Seam Shape Element".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Interior-Seam-Shape-Element
   CATEGORY BY Seam-SE-Type OF Seam-Shape-Element
ENDE

EXPRESS:

ENTITY Interior_Seam_Shape_Element
   SUBTYPE OF ( Seam_Shape_Element );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Loop

Entity Number: TOP-4

Entity Definition: Defined in the PDES Topology reference model.

Business Rules: Every "Loop":
- bounds zero, one, or more "Loop Logical Structures".
- is used as zero, one, or more "Brep Perimeter Reps".
- is used as zero, one, or more "SOLE Loop Perimeter Reps".
- is used as zero, one, or more "Partial Rev SOR Loop Perimeter Reps".
- is used as zero, one, or more "Surface Perimeter Reps".
- is used as zero, one, or more "Wireframe Area Reps".
- is used as zero, one, or more "Wireframe NM Area Reps".
- is used as zero, one, or more "Wireframe Perimeter Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Loop Logical Structure

Entity Number: TOP-25

Entity Definition: Defined in the PDES Topology reference model.

Business Rules: Every "Loop Logical Structure":
- is used as zero, one, or more "Full Rev SOR Volume Reps".
- is bound by one, and only one, "Loop".
- bounds one, and only one, "Face".
Section 1: INTEGRATION CORE MODEL

Entity Name: Manifold Solid Brep Model
Entity Number: GEO-88
Entity Definition: Defined in the PDES Geometry reference model.
Business Rules: Every "Manifold Solid Brep Model":
- is a "Geometric Model".
- is the context of zero, one, or more "Brep Area Reps".
- is the context of zero, one, or more "Brep Dim 0 SE Reps".
- is the context of zero, one, or more "Brep NM Area Reps".
- is used as zero, one, or more "Brep Object Reps".
- is the context of zero, one, or more "Brep Perimeter Reps".
- is the context of zero, one, or more "Brep Seam Reps".
- is the context of zero, one, or more "Brep Volume Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Maximal Area Representation

Entity Number: INT-105

Entity Definition: The symbolic description of a "Maximal Area Shape Element" in a "Geometric Model".

Business Rules: Every "Maximal Area Representation":

- represents one, and only one, "Maximal Area Shape Element".
- must be one, and only one, of the following:
  - "SOLE Max Area Rep"
  - "Partial Rev SOR Max Area Rep"

Primary Key Attributes:

- Shape ID
- Shape Elem ID
- Max Area Rep ID

Other Attributes:

- Maximal Area Rep Type

SML:

ENTITY Maximal-Area-Representation
  KEY
    Maximal-Area-Shape-Element "is represented by"
    Max-Area-Rep-ID
  ENDK
  Max-Area-Rep-Type NONULL
ENDENTITY

EXPRESS:

ENTITY Maximal_Area_Representation
  SUPERTYPE OF ( SOLE_Max_Area_Rep
                 OR Partial_Rev_SOR_Max_Area_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Maximal Area Shape Element

Entity Number: INT-9

Entity Definition: An "Area Shape Element" that has no adjacent "Area Shape Element" with the same underlying mathematical surface.

Business Rules: Every "Maximal Area Shape Element" is an "Area Shape Element".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Maximal-Area-Shape-Element
   CATEGORY BY Area-SE-Type OF Area-Shape-Element
END

EXPRESS:

ENTITY Maximal_Area_Shape_Element
   SUBTYPE OF ( Area_Shape_Element );
END_ENTITY;
**Section 1: INTEGRATION CORE MODEL**

**Entity Name:** Nonmaximal Area Representation

**Entity Number:** INT-52

**Entity Definition:** The symbolic description of a "Nonmaximal Area Shape Element" in a "Geometric Model".

**Business Rules:** Every "Nonmaximal Area Representation":
- represents one, and only one, "Nonmaximal Area Shape Element".
- must be one, and only one, of the following:
  - "Brep NM Area Rep"
  - "Facetted Brep NM Area Rep"
  - "SOLE Edge NM Area Rep"
  - "SOLE Subface NM Area Rep"
  - "SOR Edge NM Area Rep"
  - "Partial Rev SOR Subface NM Area Rep"
  - "Surface NM Area Rep"
  - "Unstruct Geometry NM Area Rep"
  - "Wireframe NM Area Rep"

**Primary Key Attributes:**
- Shape ID (FK)
- Shape Elem ID (FK)
- Nonmax Area Rep ID
  - The means of distinguishing each "Nonmaximal Area Representation" from all the others for the same "Nonmaximal Area Shape Element".

**Other Attributes:**
- Nonmax Area Rep Type
  - The means of determining whether a "Nonmaximal Area Representation" is a(n):
    - "Brep NM Area Rep",
    - "Facetted Brep NM Area Rep",
    - "SOLE Edge NM Area Rep",
    - "SOLE Subface NM Area Rep",
    - "SOR Edge NM Area Rep",
    - "Partial Rev SOR Subface NM Area Rep",
    - "Surface NM Area Rep",
    - "Unstruct Geometry NM Area Rep", or
    - "Wireframe NM Area Rep".

Page D3.1.1-94
Section 1: INTEGRATION CORE MODEL

SML:

ENTITY Nonmaximal-Area-Representation
  KEY
    Nonmaximal-Area-Shape-Element "is represented by"
    Nonmax-Area-Rep-ID
  ENDK
  Nonmax-Area-Rep-Type NONULL
END Entity

EXPRESS:

ENTITY Nonmaximal_Area_Representation
  SUPERTYPE OF ( Brep_NM_Area_Rep
                  OR Facetted_Brep_NM_Area_Rep
                  OR SOLE_Edge_NM_Area_Rep
                  OR SOLE_Subface_NM_Area_Rep
                  OR SOR_Edge_NM_Area_Rep
                  OR Partial_Rev_SOR_Subface_NM_Area_Rep
                  OR Surface_NM_Area_Rep
                  OR Unstruct_Geometry_NM_Area_Rep
                  OR Wireframe_NM_Area_Rep );

END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Nonmaximal Area Shape Element

Entity Number: INT-10

Entity Definition: An "Area Shape Element" that has at least one adjacent "Area Shape Element" with the same underlying mathematical surface.

Business Rules: Every "Nonmaximal Area Shape Element":
• is an "Area Shape Element".
• is represented by zero, one, or more "Nonmaximal Area Representations".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Nonmaximal-Area-Shape-Element
 CATEGORY BY Area-SE-Type OF Area-Shape-Element
 ENDE

EXPRESS:

ENTITY Nonmaximal_Area_Shape_Element
 SUBTYPE OF ( Area_Shape_Element );
 Representations : LIST OF [ 0 : # ] OF Nonmaximal_Area_Representation;
 END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Object Assembly Representation

Entity Number: INT-41

Entity Definition: The symbolic description of an "Object Assembly Shape Element" in a "Geometric Model".

Business Rules: Every "Object Assembly Representation":
- represents one, and only one, "Object Assembly Shape Element".
- is one, and only one, "Geometric Assembly Model".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Obj Assem Rep ID

The means of distinguishing each "Object Assembly Representation" from all the others for the same "Object Assembly Shape Element".

Other Attributes:

- Geom Assem Model ID (FK)

SML:

ENTITY Object-Assembly-Representation
  KEY
    Object-Assembly-Shape-Element "is represented by"
    Obj-Assem-Rep-ID
  ENDK
  Geometric-Assembly-Model "is used as" NONULL
ENDE

EXPRESS:

ENTITY Object_Assembly_Representation;
  Definition : Geometric_Assembly_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Object Assembly Shape Element

Entity Number: INT-6

Entity Definition: A "Dimensionality 3 Shape Element" that is a collection of two or more positioned "Object Shape Elements".

Business Rules: Every "Object Assembly Shape Element":
  • is a "Dimensionality 3 Shape Element".
  • is represented by zero, one, or more "Object Assembly Representations".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Object-Assembly-Shape-Element
  CATEGORY BY Dim-3-SE-Type OF Dimensionality-3-Shape-Element
END

EXPRESS:

ENTITY Object_Assembly_Shape_Element
  SUBTYPE OF ( Dimensionality_3_Shape_Element );
  Representations : LIST OF [ 0 : # ] OF Object_Assembly_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Object Representation

Entity Number: INT-30

Entity Definition: The symbolic description of an "Object Shape Element" in a "Geometric Model".

Business Rules: Every "Object Representation":
  • represents one, and only one, "Object Shape Element".
  • must be one, and only one, of the following:
    "Brep Object Rep"
    "Full Rev SOR Object Rep"
    "Facetted Brep Object Rep"

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Object Rep ID

Other Attributes:
- Object Rep Type

The means of distinguishing each "Object Representation" from all the others for the same "Object Shape Element".

Object Rep Type

The means of determining whether an "Object Representation" is a(n):
  "Brep Object Rep",
  "Full Rev SOR Object Rep", or
  "Facetted Brep Object Rep".

SML:

ENTITY Object-Representation
  KEY Object-Shape-Element "is represented by"
     Object-Rep-ID
  ENDK
  Object-Rep-Type NONULL
ENDE
Section 1: INTEGRATION CORE MODEL

ENTITY Object_Representation
  SUPERTYPE OF ( Brep_Object_Rep
  OR Full_Rev_SOR_Object_Rep
  OR Facetted_Brep_Object_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Object Shape Element
Entity Number: INT-4
Entity Definition: A "Dimensionality 3 Shape Element" that is arcwise connected. It may have interior voids.

Business Rules: Every "Object Shape Element":
• is a "Dimensionality 3 Shape Element".
• is zero or one "Voidless Volume Shape Element".
• is represented by zero, one, or more "Object Representations".

Primary Key Attributes:

- Shape ID
- Shape Elem ID

Other Attributes:

- Object SE Type

The means of determining whether an "Object Shape Element" is a "Voidless Volume Shape Element" or not.

SML:

ENTITY Object-Shape-Element

CATEGORY BY Dim-3-SE-Type OF Dimensionality-3-Shape-Element
Object-SE-Type

ENDE

EXPRESS:

ENTITY Object_Shape_Element

SUBTYPE OF ( Dimensionality_3_Shape_Element );
Voidless_Volume : OPTIONAL Voidless_Volume_Shape_Element;
Representations : LIST OF [ 0 : # ] OF Object_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Pair of Equivalent Geometric Models

Entity Number: INT-24

Entity Definition: Identifies alternative representations of shape.

Business Rules: Every "Pair of Equivalent Geometric Models":
• includes one, and only one, first "Geometric Model".
• includes one, and only one, second "Geometric Model".

Primary Key Attributes:

Geom Model 1 ID

(FK) – The role name for the Geom Model ID of one "Geometric Model" in the pair of equivalent models.

Geom Model 2 ID

(FK) – The role name for the Geom Model ID of the other "Geometric Model" in the pair of equivalent models.

Other Attributes:

none

SML:

ENTITY Pair-of-Equivalent-Geometric-Models
  KEY
    Geometric-Model "is one in"
    ROLE Geom-Model-1-ID FOR Geom-Model-ID
    Geometric-Model "is other in"
    ROLE Geom-Model-2-ID FOR Geom-Model-ID
  ENDK
ENDE

EXPRESS:

ENTITY Pair_of_Equivalent_Geometric_Models;
  First_Model : Geometric_Model;
  Second_Model : Geometric_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Bdry Loc Rep

Entity Number: INT-97

Entity Definition: The use of a "Vertex" in a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Boundary Location Shape Element".

Business Rules: Every "Partial Rev SOR Bdry Loc Rep":
- represents one, and only one, "Boundary Location Shape Element".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Bdry Loc Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)
- Bdry Loc Start or End Face Indicates whether the "Partial Rev SOR Bdry Loc Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Bdry-Loc-Rep
   KEY Boundary-Location-Shape-Element "is represented by"
      Bdry-Loc-Rep-ID
   ENDK
   Vertex "is used as" NONULL
   Solid-of-Revolution-Model "is context of" NONULL
   Bdry-Loc-Start-or-End-Face NONULL
ENDE
EXPRESS:

TYPE

Partial_Rev_SOR_Face_Types =
  ENUMERATION OF ( Starting_SOR_Face, Ending_SOR_Face );
END_TYPE;

ENTITY Partial_Rev_SOR_Bdry_Loc_Rep;
  Definition : Vertex;
  Context : Solid_of_Revolution_Model;
  Face : Partial_Rev_SOR_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Corner Rep

Entity Number: INT-93

Entity Definition: The use of a "Vertex" in a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Corner Shape Element".

Business Rules: Every "Partial Rev SOR Corner Rep":
- is a "Corner Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Corner Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)
- Corner Start or End Face Indicates whether the "Partial Rev SOR Corner Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Corner-Rep
   CATEGORY BY Corner-Rep-Type OF Corner-Representation
   Vertex "is used as" NONULL
   Solid-of-Revolution-Model "is context of" NONULL
   Corner-Start-or-End-Face NONULL
END_ENTITY;

EXPRESS:

ENTITY Partial_Rev_SOR_Corner_Rep
   SUBTYPE OF ( Corner_Representation );
   Definition : Vertex;
   Context : Solid_of_Revolution_Model;
   Face : Partial_Rev_SOR_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Edge Perimeter Rep

Entity Number: INT-81

Entity Definition: The use of an "Edge" in a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Perimeter Shape Element".

Business Rules: Every "Partial Rev SOR Edge Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Edge".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

Other Attributes:

- SOR Model ID (FK)
- Edge ID (FK)

SML:

ENTITY Partial-Rev-SOR-Edge-Perimeter-Rep
    CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
    Edge "is used as" NONULL
    Solid-of-Revolution-Model "is context of" NONULL
END

EXPRESSION:

ENTITY Partial_Rev_SOR_Edge_Perimeter_Rep
    SUBTYPE OF ( Perimeter_Representation );
    Definition : Edge;
    Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Edge Seam Rep

Entity Number: INT-67

Entity Definition: The use of an "Edge" in a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Seam Shape Element".

Business Rules: Every "Partial Rev SOR Edge Seam Rep":
- is a "Seam Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Edge".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Seam Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Edge ID (FK)
- SOR Seam Start or End Face
  Indicates whether the "Partial Rev SOR Edge Seam Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Edge-Seam-Rep  
CATEGORY BY Seam-Rep-Type OF Seam-Representation  
Edge "is used as" NONULL  
Solid-of-Revolution-Model "is context of" NONULL  
SOR-Seam-Start-or-End-Face NONULL  
ENDE

EXPRESSION:

ENTITY Partial_Rev_SOR_Edge_Seam_Rep  
SUBTYPE OF ( Seam_Representation );  
Definition : Edge;  
Context : Solid_of_Revolution_Model;  
Face : Partial_Rev_SOR_Face_Types;  
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Int Loc Rep

Entity Number: INT-101

Entity Definition: The use of a "Vertex" in a "Solid of Revolution Model" that is revolved through less than 360° to represent an "Interior Location Shape Element".

Business Rules: Every "Partial Rev SOR Int Loc Rep":
- is a "Interior Location Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Int Loc Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)
- Int Loc Start or End Face Indicates whether the "Partial Rev SOR Int Loc Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Int-Loc-Rep
  CATEGORY BY Int-Loc-Rep-Type OF Interior-Location-Representation
  Vertex "is used as" NONULL
  Solid-of-Revolution-Model "is context of" NONULL
  Int-Loc-Start-or-End-Face NONULL
ENDE

EXPRESS:

ENTITY Partial_Rev_SOR_Int_Loc_Rep
  SUBTYPE OF ( Interior_Location_Representation );
  Definition : Vertex;
  Context : Solid_of_Revolution_Model;
  Face : Partial_Rev_SOR_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Loop Perimeter Rep

Entity Number: INT-77

Entity Definition: The use of a "Loop" in a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Perimeter Shape Element".

Business Rules: Every "Partial Rev SOR Loop Perimeter Rep":
  • is a "Perimeter Representation".
  • is in one, and only one, "Solid of Revolution Model".
  • is one, and only one, "Loop".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Perim Rep ID (FK)

Other Attributes:

SOR Model ID (FK)
Loop ID (FK)
SOR Perim Start or End Face
  Indicates whether the "Partial Rev SOR Loop Perimeter Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Loop-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Loop "is used as" NONULL
  Solid-of-Revolution-Model "is context of" NONULL
  SOR-Perim-Start-or-End-Face NONULL
ENDE
Section 1: INTEGRATION CORE MODEL

EXPRESS:

ENTITY Partial_Rev_SOR_Loop_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Loop;
  Context : Solid_of_Revolution_Model;
  Face : Partial_Rev_SOR_Face Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Max Area Rep

Entity Number: INT-46

Entity Definition: The use of a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Maximal Area Shape Element".

Business Rules: Every "Partial Rev SOR Max Area Rep":
• is an "Maximal Area Representation".
• is one, and only one, "Solid of Revolution Model".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Area Rep ID (FK)

Other Attributes:

SOR Model ID (FK)
SOR Max Area Start or End Face
Indicates whether the "Partial Rev SOR Max Area Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Max-Area-Rep
  CATEGORY BY Max-Area-Rep-Type OF Maximal-Area-Representation
  Solid-of-Revolution-Model "is used as" NONULL
  SOR-Max-Area-Start-or-End-Face NONULL
END

EXPRESS:

ENTITY Partial_Rev_SOR_Max_Area_Rep
  SUBTYPE OF ( Maximal_Area_Representation );
  Definition : Solid_of_Revolution_Model;
  Face : Partial_Rev_SOR_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Subface NM Area Rep

Entity Number: INT-56

Entity Definition: The use of a "Subface" in a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Nonmaximal Area Shape Element".

Business Rules: Every "Partial Rev SOR Subface NM Area Rep":
- is a "Nonmaximal Area Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Subface".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Nonmax Area Rep ID (FK)

Other Attributes:

SOR Model ID (FK)
Subface ID (FK)
SOR Nonmax Area Start or End Face Indicates whether the "Partial Rev SOR Subface NM Area Rep" is on the starting or ending face in the partially revolved "Solid of Revolution Model".

SML:

ENTITY Partial-Rev-SOR-Subface-NM-Area-Rep
CATEGORY BY Nonmax-Area-Rep-Type
    OF Nonmaximal-Area-Representation
Subface "is used as" NONULL
Solid-of-Revolution-Model "is context of" NONULL
SOR-Nonmax-Area-Start-or-End-Face NONULL
ENDE
Section 1: INTEGRATION CORE MODEL

EXPRESS:

ENTITY Partial_Rev_SOR_Subface_NM_Area_Rep
  SUBTYPE OF ( Nonmaximal_Area_Representation );
  Definition : Subface;
  Context : Solid_of_Revolution_Model;
  Face : Partial_Rev_SOR_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Partial Rev SOR Volume Rep

Entity Number: INT-38

Entity Definition: The use of a "Solid of Revolution Model" that is revolved through less than 360° to represent a "Voidless Volume Shape Element".

Business Rules: Every "Partial Rev SOR Volume Rep":
  • is a "Voidless Volume Representation".
  • is one, and only one, "Solid of Revolution Model".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Volume Rep ID (FK)

Other Attributes:

SOR Model ID (FK)

SML:

ENTITY Partial-Rev-SOR-Volume-Rep
  CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation
  Solid-of-Revolution-Model "is used as" NONULL
END

EXPRESS:

ENTITY Partial_Rev_SOR_Volume_Rep
  SUBTYPE OF ( Voidless_Volume_Representation );
  Definition : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Perimeter Representation

Entity Number: INT-73

Entity Definition: The symbolic description of a "Perimeter Shape Element" in a "Geometric Model".

Business Rules:

Every "Perimeter Representation":
- represents one, and only one, "Perimeter Shape Element".
- must be one, and only one, of the following:
  "Brep Perimeter Rep"
  "Facetted Brep Perimeter Rep"
  "SOLE Edge Perimeter Rep"
  "SOLE Loop Perimeter Rep"
  "Partial Rev SOR Edge Perimeter Rep"
  "Partial Rev SOR Loop Perimeter Rep"
  "Full Rev SOR Vertex Perimeter Rep"
  "Surface Perimeter Rep"
  "Unstruct Geometry Perimeter Rep"
  "Wireframe Perimeter Rep"

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID

The means of distinguishing each "Perimeter Representation" from all the others for the same "Perimeter Shape Element".

Other Attributes:

- Perim Rep Type

The means of determining whether a "Perimeter Representation" is a(n):
  "Brep Perimeter Rep",
  "Facetted Brep Perimeter Rep",
  "SOLE Edge Perimeter Rep",
  "SOLE Loop Perimeter Rep",
  "Partial Rev SOR Edge Perimeter Rep",
  "Partial Rev SOR Loop Perimeter Rep",
  "Full Rev SOR Vertex Perimeter Rep",
  "Surface Perimeter Rep",
  "Unstruct Geometry Perimeter Rep", or
  "Wireframe Perimeter Rep".
Section 1: INTEGRATION CORE MODEL

SML:

ENTITY Perimeter-Representation
    KEY
    Perimeter-Shape-Element "is represented by"
    Perim-Rep-ID
    ENDK
    Perim-Rep-Type NONULL
    ENDE

EXPRESS:

ENTITY Perimeter_Representation
    SUPERTYPE OF (Brep_Perimeter_Rep
    OR Facetted_Brep_Perimeter_Rep
    OR SOLE_Edge_Perimeter_Rep
    OR SOLE_Loop_Perimeter_Rep
    OR Partial_Rev_SOR_Edge_Perimeter_Rep
    OR Partial_Rev_SOR_Loop_Perimeter_Rep
    OR Full_Rev_SOR_Vertex_Perimeter_Rep
    OR Surface_Perimeter_Rep
    OR Unstruct_Geometry_Perimeter_Rep
    OR Wireframe_Perimeter_Rep )
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Perimeter Shape Element

Entity Number: INT-18

Entity Definition: A "Dimensionality 1 Shape Element" that is a closed boundary of a "Maximal Area Shape Element".

Business Rules: Every "Perimeter Shape Element":
• is a "Dimensionality 1 Shape Element".
• is represented by zero, one, or more "Perimeter Representations".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Perimeter-Shape-Element
   CATEGORY BY Dim-1-SE-Type OF Dimensionality-1-Shape-Element
ENDE

EXPRESS:

ENTITY Perimeter_Shape_Element
   SUBTYPE OF ( Dimensionality_1_Shape_Element );
   Representations : LIST OF [ 0 : # ] OF Perimeter_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Point

Entity Number: GEO-2

Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "Point" is used as zero, one, or more "Facetted Brep Dim 0 SE Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Polyloop
Entity Number: TOP-11
Entity Definition: Defined in the PDES Topology reference model.
Business Rules: Every "Polyloop" is used as zero, one, or more "Facetted Brep Perimeter Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Product Item

Entity Number: PSCM-1

Entity Definition: Defined in the PDES PSCM reference model.

Business Rules: Every "Product Item" has one or more "Product Item Versions".
Entity Name: Product Item Version

Entity Number: PSCM-2

Entity Definition: Defined in the PDES PSCM reference model.

Business Rules: Every "Product Item Version":
- is for one, and only one, "Product Item".
- is described by one or more "Product Item Version Functional Definitions".
Section 1: INTEGRATION CORE MODEL

Entity Name: Product Item Version Definition Shape
Entity Number: PSCM-27
Entity Definition: Defined in the PDES PSCM reference model.
Business Rules: Every "Product Item Version Definition Shape":
  • is the primary shape for one, and only one, "Product Item Version Functional Definition".
  • is one, and only one, "Shape".
Section 1: INTEGRATION CORE MODEL

Entity Name: Product Item Version Functional Definition

Entity Number: PSCM-4

Entity Definition: Defined in the PDES PSCM reference model.

Business Rules: Every "Product Item Version Functional Definition":
   • describes one, and only one, "Product Item Version".
   • has a primary shape of zero or one "Product Item Version Definition Shape".
   • is modeled by zero, one, or more "FEM Product Item Version Definitions".
Section 1: INTEGRATION CORE MODEL

Entity Name: Seam Representation

Entity Number: INT-63

Entity Definition: The symbolic description of a "Seam Shape Element" in a "Geometric Model".

Business Rules: Every "Seam Representation":
- represents one, and only one, "Seam Shape Element".
- must be one, and only one, of the following:
  - "Brep Seam Rep"
  - "SOLE Edge Seam Rep"
  - "SOLE Vertex Seam Rep"
  - "Partial Rev SOR Edge Seam Rep"
  - "SOR Vertex Seam Rep"
  - "Surface Seam Rep"
  - "Unstruct Geometry Seam Rep"
  - "Wireframe Seam Rep"

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Seam Rep ID

Other Attributes:

- Seam Rep Type
  - The means of determining whether a "Seam Representation" is a(n):
    - "Brep Seam Rep",
    - "SOLE Edge Seam Rep",
    - "SOLE Vertex Seam Rep",
    - "Partial Rev SOR Edge Seam Rep",
    - "SOR Vertex Seam Rep",
    - "Surface Seam Rep",
    - "Unstruct Geometry Seam Rep", or
    - "Wireframe Seam Rep".
Section 1: INTEGRATION CORE MODEL

SML:

ENTITY Seam-Representation
  KEY
    Seam-Shape-Element "is represented by"
    Seam-Rep-ID
  ENDK
  Seam-Rep-Type NONULL
ENDE

EXPRESS:

ENTITY Seam_Representation
  SUPERTYPE OF ( Brep_Seam_Rep
               OR SOLE_Edge_Seam_Rep
               OR SOLE_Vertex_Seam_Rep
               OR Partial_Rev_SOR_Edge_Seam_Rep
               OR SOR_Vertex_Seam_Rep
               OR Surface_Seam_Rep
               OR Unstruct_Geometry_Seam_Rep
               OR Wireframe_Seam_Rep );

END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Seam Shape Element

Entity Number: INT-14

Entity Definition: A "Dimensionality 1 Shape Element" that has a uniform underlying mathematical curve.

Business Rules: Every "Seam Shape Element":
- is a "Dimensionality 1 Shape Element".
- must be one, and only one, of the following:
  - "Edge Shape Element"
  - "Interior Seam Shape Element"
  - "Subedge Shape Element"
- is represented by zero, one, or more "Seam Representations".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)

Other Attributes:

Seam SE Type The means of determining whether a "Seam Shape Element" is a(n):
- "Edge Shape Element",
- "Interior Seam Shape Element", or
- "Subedge Shape Element".

SML:

ENTITY Seam-Shape-Element
  CATEGORY BY Dim-1-SE-Type OF Dimensionality-1-Shape-Element
  Seam-SE-Type NONULL
END

EXPRESS:

ENTITY Seam_Shape_Element
  SUPERTYPE OF ( Edge_Shape_Element
                 OR Subedge_Shape-Element
                 OR Interior_Seam_Shape-Element );
  SUBTYPE OF ( Dimensionality_1_Shape_Element );
  Representations : LIST OF [ 0 : # ] OF Seam_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Shape

Entity Number: INT-1

Entity Definition: The size, spatial configuration, and proportions of a real or conceived thing, independent of whether or how it is represented.

Business Rules: Every "Shape":
- is an aggregation zero, one, or more "Shape Elements".
- is the primary shape for zero, one, more "Product Item Version Definition Shapes".

Primary Key Attributes:

Shape ID

The means of distinguishing each "Shape" from all others.

Other Attributes: none

SML:

ENTITY Shape
   KEY
      Shape-ID
   ENDK
ENDE

EXPRESS:

ENTITY Shape;
   Elements : LIST OF [ 0 : # ] Shape_Element;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Shape Element

Entity Number: INT-2

Entity Definition: A distinguishable aspect of a "Shape".

Business Rules: Every "Shape Element":
- is part of one, and only one, "Shape".
- must be one, and only one, of the following:
  - "Dimensionality 0 Shape Element"
  - "Dimensionality 1 Shape Element"
  - "Dimensionality 2 Shape Element"
  - "Dimensionality 3 Shape Element"

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>(FK) The means of distinguishing each &quot;Shape Element&quot; from all the others belonging to the same &quot;Shape&quot;.</td>
</tr>
<tr>
<td>Shape Element ID</td>
<td></td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| Shape Element Type | The means of determining whether a "Shape Element" is a(n):
  - "Dimensionality 0 Shape Element",
  - "Dimensionality 1 Shape Element",
  - "Dimensionality 2 Shape Element",
  - "Dimensionality 3 Shape Element". |

SML:

```sml
ENTITY Shape-Element
  KEY
    Shape "is aggregation of"
    Shape-Elem-ID
  ENDK
  Shape-Elem-Type NONULL
ENDE
```
EXPRESS:

ENTITY Shape_Element
  SUPERTYPE OF ( Dimensionality_0_Shape_Element
         OR Dimensionality_1_Shape_Element
         OR Dimensionality_2_Shape_Element
         OR Dimensionality_3_Shape_Element );

END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Shell

Entity Number: TOP-6

Entity Definition: Defined in the PDES Topology reference model.

Business Rules: Every "Shell":
- is used as zero, one, or more "Brep Volume Reps".
- is used as zero, one, or more "Facetted Brep Volume Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Dim 0 SE Rep

Entity Number: INT-88

Entity Definition: The use of a "Vertex" in a "Solid of Linear Extrusion Model" to represent a "Dimensionality 0 Shape Element".

Business Rules: Every "SOLE Dim 0 SE Rep":
- is a "Dim 0 Shape Element Representation".
- is in one, and only one, "Solid of Linear Extrusion Model".
- is one, and only one, "Vertex".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td></td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Dim 0 Rep ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FK</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLE Model ID</td>
<td></td>
</tr>
<tr>
<td>Vertex ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Dim 0 Start or End Face</td>
<td>Indicates whether the &quot;SOLE Dim 0 SE Rep&quot; is on the starting or ending face in the &quot;Solid of Linear Extrusion Model&quot;.</td>
</tr>
</tbody>
</table>

SML:

ENTITY SOLE-Dim-0-SE-Rep
CATEGORY BY Dim-0-Rep-Type OF Dim-0-Shape-Element-Representation
Vertex "is used as" NONULL
Solid-of-Linear-Extrusion-Model "is context of" NONULL
Dim-0-Start-or-End-Face NONULL
ENDE
Section 1: INTEGRATION CORE MODEL

EXPRESS:

TYPE
    SOLE_Face_Types =
        ENUMERATION OF ( StartingSOLE_Face, EndingSOLE_Face );
END_TYPE;

ENTITY SOLE_Dim_0_SE_Rep
    SUBTYPE OF ( Dim_0_Shape_Element_Representation );
    Definition: Vertex;
    Context: Solid_of_Linear_Extrusion_Model;
    Face: SOLE_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Edge Area Rep

Entity Number: INT-50

Entity Definition: The use of an "Edge" in a "Solid of Linear Extrusion Model" to represent an "Area Shape Element". An edge of the swept face sweeps a surface area of the prismatic solid. If the edge has an adjacent edge with the same underlying curve, the area is nonmaximal; otherwise, the area is maximal.

Business Rules: Every "SOLE Edge Area Rep":
  • is an "Area Representation".
  • is in one, and only one, "Solid of Linear Extrusion Model".
  • is one, and only one, "Edge".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Area Rep ID (FK)

Other Attributes:

SOLE Model ID (FK)
Edge ID (FK)

SML:

ENTITY SOLE-Edge-Area-Rep
  CATEGORY BY Area-Rep-Type OF Area-Representation
  Edge "is used as" NONULL
  Solid-of-Linear-Extrusion-Model "is context of" NONULL
END_ENTITY

EXPRESS:

ENTITY SOLE_Edge_Area_Rep
  SUBTYPE OF ( Area_Representation );
  Definition : Edge;
  Context : Solid_of_Linear_Extrusion_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Edge NM Area Rep

Entity Number: INT-60

Entity Definition: The use of an "Edge" in a "Solid of Linear Extrusion Model" to represent a "Nonmaximal Area Shape Element". An edge of a subface of the face swept by an SOLE sweeps a portion of the surface area of the solid, provided that the edge is a portion of an edge of the swept face. If, in addition, the subface edge is a proper subset of the swept face edge, the edge sweeps a nonmaximal area. (If the subface edge is an entire edge of the swept face, it seems pointless to regard the subface edge as the representation of the swept area; the swept face edge is available and more natural. Therefore, it is considered that only a nonmaximal area can be represented by an edge of a subface of the swept face.)

Business Rules: Every "SOLE Edge NM Area Rep":
• is a "Nonmaximal Area Representation".
• is in one, and only one, "Solid of Linear Extrusion Model".
• is one, and only one, "Edge".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Nonmax Area Rep ID (FK)

Other Attributes:
- SOLE Model ID (FK)
- Edge ID (FK)

SML:

ENTITY SOLE-Edge-NM-Area-Rep
  CATEGORY BY Nonmax-Area-Rep-Type
  OF Nonmaximal-Area-Representation
  Edge "is used as" NONULL
  Solid-of-Linear-Extrusion-Model "is context of" NONULL
ENDE
EXPRESS:

ENTITY SOLE_Edge_NM_Area_Rep
  SUBTYPE OF ( Nonmaximal_Area_Representation );
  Definition : Edge;
  Context : Solid_of_Linear_Extrusion_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

**Entity Name:** SOLE Edge Perimeter Rep

**Entity Number:** INT-82

**Entity Definition:** The use of an "Edge" in a "Solid of Linear Extrusion Model" to represent a "Perimeter Shape Element".

**Business Rules:** Every "SOLE Edge Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Solid of Linear Extrusion Model".
- is one, and only one, "Edge".

**Primary Key Attributes:**
- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

**Other Attributes:**
- SOLE Model ID (FK)
- Edge ID (FK)

**SML:**

```xml
ENTITY SOLE-Edge-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Edge "is used as" NONULL
  Solid-of-Linear-Extrusion-Model "is context of" NONULL
ENDE
```

**EXPRESS:**

```express
ENTITY SOLE_Edge_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Edge;
  Context : Solid_of_Linear_Extrusion_Model;
END_ENTITY;
```

Page D3.1.1-136
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Edge Seam Rep

Entity Number: INT-68

Entity Definition: The use of an "Edge" in a "Solid of Linear Extrusion Model" to represent a "Seam Shape Element". An edge of the face swept by an SOLE corresponds to some or all of a seam between the "base" or "cap" of the prismatic solid and a side area of the solid. If the edge has no adjacent face with the same underlying curve, the seam is an "Edge Shape Element", i.e., the intersection of maximal areas. If there is an adjacent face with the same underlying curve, the seam is a "Subedge Shape Element". An "Interior Seam Shape Element" cannot be represented in this way.

Business Rules: Every "SOLE Edge Seam Rep":
- is a "Seam Representation".
- is in one, and only one, "Solid of Linear Extrusion Model".
- is one, and only one, "Edge".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Seam Rep ID (FK)

Other Attributes:
- SOLE Model ID (FK)
- Edge ID (FK)
- SOLE Seam Start or End Face
  Indicates whether the "SOLE Edge Seam Rep" is on the starting or ending face in the "Solid of Linear Extrusion Model".

SML:

ENTITY SOLE-Edge-Seam-Rep
  CATEGORY BY Seam-Rep-Type OF Seam-Representation
  Edge "is used as" NONULL
  Solid-of-Linear-Extrusion-Model "is context of" NONULL
  SOLE-Seam-Start-or-End-Face NONULL
END
ENTITY SOLE_Edge_Seam_Rep
    SUBTYPE OF ( Seam_Representation );
    Definition : Edge;
    Context : Solid_of_Linear_Extrusion_Model;
    Face : SOLE_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Loop Perimeter Rep

Entity Number: INT-78

Entity Definition: The use of a "Loop" in a "Solid of Linear Extrusion Model" to represent a "Perimeter Shape Element".

Business Rules: Every "SOLE Loop Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Solid of Linear Extrusion Model".
- is one, and only one, "Loop".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>(FK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td></td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td></td>
</tr>
<tr>
<td>Perim Rep ID</td>
<td></td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>(FK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLE Model ID</td>
<td></td>
</tr>
<tr>
<td>Loop ID</td>
<td></td>
</tr>
<tr>
<td>SOLE Perim Start or End Face</td>
<td>Indicates whether the &quot;SOLE Loop Perimeter Rep&quot; is on the starting or ending face in the &quot;Solid of Linear Extrusion Model&quot;.</td>
</tr>
</tbody>
</table>

SML:

ENTITY SOLE-Loop-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Loop "is used as" NONULL
  Solid-of-Linear-Extrusion-Model "is context of" NONULL
  SOLE-Perim-Start-or-End-Face NONULL
END_ENTITY;

EXPRESSION:

ENTITY SOLE_Loop_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Loop;
  Context : Solid_of_Linear_Extrusion_Model;
  Face : SOLE_Face TYPES;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Max Area Rep
Entity Number: INT-47

Entity Definition: The use of a "Solid of Linear Extrusion Model" to represent an "Maximal Area Shape Element". The swept face of an SOLE corresponds to two surface areas of the solid: the "base" and the "cap" of the prism. By the nature of an SOLE, these two areas are necessarily maximal.

Business Rules: Every "SOLE Max Area Rep":
• is a "Maximal Area Representation".
• is one, and only one, "Solid of Linear Extrusion Model".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Area Rep ID (FK)

Other Attributes:
- SOLE Model ID (FK)
- SOLE Max Area Start or End Face
  Indicates whether the "SOLE Max Area Rep" is on the starting or ending face in the "Solid of Linear Extrusion Model".

SML:

ENTITY SOLE-Max-Area-Rep
  CATEGORY BY Max-Area-Rep-Type OF Maximal-Area-Representation
  Solid-of-Linear-Extrusion-Model "is used as" NONULL
  SOLE-Max-Area-Start-or-End-Face NONULL
ENDER

EXRESS:

ENTITY SOLE_Max_Area_Rep
  SUBTYPE OF ( Maximal_Area_Representation );
  Definition : Solid_of_Linear_Extrusion_Model;
  Face : SOLE_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Subface NM Area Rep

Entity Number: INT-57

Entity Definition: The use of a "Subface" in a "Solid of Linear Extrusion Model" to represent a "Nonmaximal Area Shape Element". A subface of the face swept of an SOLE corresponds to two surface areas of the solid. These are nonmaximal areas: one being a subset of the "base" of the prismatic solid; the other, a subset of the "cap".

Business Rules: Every "SOLE Subface NM Area Rep":
- is a "Nonmaximal Area Representation".
- is in one, and only one, "Solid of Linear Extrusion Model".
- is one, and only one, "Subface".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Nonmax Area Rep ID (FK)

Other Attributes:

- SOLE Model ID (FK)
- Subface ID (FK)
- SOLE Nonmax Area Start or End Face Indicates whether the "SOLE Subface NM Area Rep" is on the starting or ending face in the "Solid of Linear Extrusion Model".

SML:

ENTITY SOLE-Subface-NM-Area-Rep
    CATEGORY BY Nonmax-Area-Rep-Type
    OF Nonmaximal-Area-Representation
    Subface "is used as" NONULL
    Solid-of-Linear-Extrusion-Model "is context of" NONULL
    SOLE-Nonmax-Area-Start-or-End-Face NONULL
ENDE

Page D3.1.1-141
Section 1: INTEGRATION CORE MODEL

EXPRESS:

ENTITY SOLE_Subface_NM_Area_Rep
  SUBTYPE OF ( Nonmaximal_Area_Representation );
  Definition: Subface;
  Context: Solid_of_Linear_Extrusion_Model;
  Face: SOLE_Face_Types;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Vertex Seam Rep

Entity Number: INT-71

Entity Definition: The use of a "Vertex" in a "Solid of Linear Extrusion Model" to represent a "Seam Shape Element".

Business Rules: Every "SOLE Vertex Seam Rep":
- is a "Seam Representation".
- is in one, and only one, "Solid of Linear Extrusion Model".
- is one, and only one, "Vertex".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>FK</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>FK</td>
</tr>
<tr>
<td>Seam Rep ID</td>
<td>FK</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLE Model ID</td>
<td>FK</td>
</tr>
<tr>
<td>Vertex ID</td>
<td>FK</td>
</tr>
</tbody>
</table>

SML:

ENTITY SOLE-Vertex-Seam-Rep
   CATEGORY BY Seam-Rep-Type OF Seam-Representation
   Vertex "is used as" NONULL
   Solid-of-Linear-Extrusion-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY SOLE_Vertex_Seam_Rep
   SUBTYPE OF ( Seam_Representation );
   Definition : Vertex;
   Context : Solid_of_Linear_Extrusion_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOLE Volume Rep

Entity Number: INT-40

Entity Definition: The use of a "Solid of Linear Extrusion Model" to represent a "Voidless Volume Shape Element". An SOLE is a solid model. Since an SOLE is a sweep of a single face, the solid must have a connected interior. The nature of the representation makes it impossible for the solid to have any voids.

Business Rules: Every "SOLE Volume Rep":
   • is a "Voidless Volume Representation".
   • is one, and only one, "Solid of Linear Extrusion Model".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Volume Rep ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLE Model ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

SML:

ENTITY SOLE-Volume-Rep
   CATEGORY BY Volume-Rep-Type OF Voidless-Volume-Representation Solid-of-Linear-Extrusion-Model "is used as" NONULL
END

EXPRESS:

ENTITY SOLE_Volume_Rep
   SUBTYPE OF ( Voidless_Volume_Representation );
   Definition : Solid_of_Linear_Extrusion_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Solid of Linear Extrusion Model
Entity Number: GEO-106
Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "Solid of Linear Extrusion Model":
- is a "Geometric Model".
- is the context of zero, one, or more "SOLE Dim 0 SE Reps".
- is the context of zero, one, or more "SOLE Edge Area Reps".
- is the context of zero, one, or more "SOLE Edge NM Area Reps".
- is the context of zero, one, or more "SOLE Edge Perimeter Reps".
- is the context of zero, one, or more "SOLE Edge Seam Reps".
- is used as zero, one, or more "SOLE Max Area Reps".
- is the context of zero, one, or more "SOLE Loop Perimeter Reps".
- is the context of zero, one, or more "SOLE Subface NM Area Reps".
- is the context of zero, one, or more "SOLE Vertex Seam Reps".
- is used as zero, one, or more "SOLE Volume Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Solid of Revolution Model

Entity Number: GEO-107

Entity Definition: Defined in the PDES Geometry reference model.

Business Rules: Every "Solid of Revolution Model":
- is a "Geometric Model".
- revolves one, and only one, "Face".
- is the context of zero, one, or more "Full Rev SOR Corner Reps".
- is the context of zero, one, or more "Full Rev SOR Int Loc Reps".
- is used as zero, one, or more "Full Rev SOR Object Reps".
- is the context of zero, one, or more "Full Rev SOR Volume Reps".
- is the context of zero, one, or more "Partial Rev SOR Bdry Loc Reps".
- is the context of zero, one, or more "Partial Rev SOR Corner Reps".
- is the context of zero, one, or more "Partial Rev SOR Int Loc Reps".
- is used as zero, one, or more "Partial Rev SOR Volume Reps".
- is the context of zero, one, or more "SOR Edge Area Reps".
- is the context of zero, one, or more "SOR Edge NM Area Reps".
- is the context of zero, one, or more "Partial Rev SOR Edge Perimeter Reps".
- is the context of zero, one, or more "Partial Rev SOR Edge Seam Reps".
- is used as zero, one, or more "Partial Rev SOR Max Area Reps".
- is the context of zero, one, or more "Partial Rev SOR Loop Perimeter Reps".
- is the context of zero, one, or more "Partial Rev SOR Subface NM Area Reps".
- is the context of zero, one, or more "Full Rev SOR Vertex Perimeter Reps".
- is the context of zero, one, or more "SOR Vertex Seam Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: SOR Edge Area Rep
Entity Number: INT-49
Entity Definition: The use of an "Edge" in a "Solid of Revolution Model" to represent an "Area Shape Element".

Business Rules: Every "SOR Edge Area Rep":
- is an "Area Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Edge".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Area Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Edge ID (FK)

SML:
ENTITY SOR-Edge-Area-Rep
CATEGORY BY Area-Rep-Type OF Area-Representation
Edge "is used as" NONULL
Solid-of-Revolution-Model "is context of" NONULL
ENDE

EXPRESS:
ENTITY SOR_Edge_Area_Rep
SUBTYPE OF ( Area_Representation );
Definition : Edge;
Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOR Edge NM Area Rep

Entity Number: INT-59

Entity Definition: The use of an "Edge" in a "Solid of Revolution Model" to represent a "Nonmaximal Area Shape Element".

Business Rules: Every "SOR Edge NM Area Rep":
- is a "Nonmaximal Area Representation".
- is in one, and only one, "Solid of Revolution Model".
- is one, and only one, "Edge".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Nonmax Area Rep ID (FK)

Other Attributes:

- SOR Model ID (FK)
- Edge ID (FK)

SML:

ENTITY SOR-Edge-NM-Area-Rep
   CATEGORY BY Nonmax-Area-Rep-Type
   OF Nonmaximal-Area-Representation
   Edge "is used as" NONULL
   Solid-of-Revolution-Model "is context of" NONULL
ENDE

EXPRESSION:

ENTITY SOR_Edge_NM_Area_Rep
   SUBTYPE OF ( Nonmaximal_Area_Representation );
   Definition : Edge;
   Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: SOR Vertex Seam Rep

Entity Number: INT-70

Entity Definition: The use of a "Vertex" in a "Solid of Revolution Model" to represent a "Seam Shape Element".

Business Rules: Every "SOR Vertex Seam Rep":
  • is a "Seam Representation".
  • is in one, and only one, "Solid of Revolution Model".
  • is one, and only one, "Vertex".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Seam Rep ID (FK)

Other Attributes:
- SOR Model ID (FK)
- Vertex ID (FK)

SML:

ENTITY SOR-Vertex-Seam-Rep
  CATEGORY BY Seam-Rep-Type OF Seam-Representation
  Vertex "is used as" NONULL
  Solid-of-Revolution-Model "is context of" NONULL
ENDE

EXpress:

ENTITY SOR_Vertex_Seam_Rep
  SUBTYPE OF ( Seam_Representation );
  Definition : Vertex;
  Context : Solid_of_Revolution_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Subedge Shape Element

Entity Number: INT-16

Entity Definition: A "Seam Shape Element" that is a proper subset of an "Edge Shape Element".

Business Rules: Every "Subedge Shape Element" is a "Seam Shape Element".

Primary Key Attributes:

  Shape ID         (FK)
  Shape Elem ID    (FK)

Other Attributes: none

SML:

ENTITY Subedge-Shape-Element
  CATEGORY BY Seam-SE-Type OF Seam-Shape-Element
ENDE

EXPRESS:

ENTITY Subedge_Shape_Element
  SUBTYPE OF ( Seam_Shape_Element );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Subface
Entity Number: TOP-8
Entity Definition: Defined in the PDES Topology reference model.

Business Rules: Every "Subface":
- is used as zero, one, or more "Brep NM Area Reps".
- is used as zero, one, or more "Facetted Brep NM Area Reps".
- is used as zero, one, or more "SOLE Subface NM Area Reps".
- is used as zero, one, or more "Partial Rev SOR Subface NM Area Reps".
- is used as zero, one, or more "Surface NM Area Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface Area Rep

Entity Number: INT-45

Entity Definition: The use of a "Face" in a "Surface Model" to represent an "Area Shape Element". In a surface model, a face is typically employed to represent a portion of surface area that (i) has a uniform underlying surface and (ii) is arcwise connected. If there is an adjacent face with the same underlying surface, the area is not maximal; otherwise, it is maximal.

Business Rules: Every "Surface Area Rep":
- is an "Area Representation".
- is in one, and only one, "Surface Model".
- is one, and only one, "Face".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Area Rep ID (FK)

Other Attributes:

- Surface Model ID (FK)
- Face ID (FK)

SML:

ENTITY Surface-Area-Rep
  CATEGORY BY Area-Rep-Type OF Area-Representation
  Face "is used as" NONULL
  Surface-Model "is context of" NONULL
END.ENTITY;

EXPRESSION:

ENTITY Surface_Area_Rep
  SUBTYPE OF ( Area_Representation );
  Definition: Face;
  Context: Surface_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface Dim 0 SE Rep

Entity Number: INT-86

Entity Definition: The use of a "Vertex" in a "Surface Model" to represent a "Dimensionality 0 Shape Element". In a surface model, vertices are employed in maximal faces, nonmaximal faces, subfaces, and vertex loops. In the first three cases, the edges that meet at the vertex may have the same underlying curve. So, a vertex may represent any of the three categories of "Dimensionality 0 Shape Element" — corner, boundary location, or interior location.

Business Rules: Every "Surface Dim 0 SE Rep":
- is a "Dim 0 Shape Element Representation".
- is in one, and only one, "Surface Model".
- is one, and only one, "Vertex".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Dim 0 Rep ID (FK)

Other Attributes:

Surface Model ID (FK)
Vertex ID (FK)

SML:

ENTITY Surface-Dim-0-SE-Rep
  CATEGORY BY Dim-0-Rep-Type OF Dim-0-Shape-Element-Representation
  Vertex "is used as" NONULL
  Surface-Model "is used as" NONULL
END

EXPRESS:

ENTITY Surface_Dim_0_SE_Rep
  SUBTYPE OF ( Dim_0_Shape_Element_Representation );
  Definition : Vertex;
  Context : Surface_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface Dim 3 SE Rep

Entity Number: INT-27

Entity Definition: The use of a "Surface Model" to represent a "Dimensionality 3 Shape Element". A surface model has no information for structuring its component shells into solids. So, representation of a "Dimensionality 3 Shape Element" by a surface model is a matter of human interpretation, rather than indigenous content, of the data. With this attitude, a surface model may represent any category of "Dimensionality 3 Shape Element".

Business Rules: Every "Surface Dim 3 SE Rep":
- is a "Dim 3 Shape Element Representation".
- is one, and only one, "Surface Model".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Dim 3 Rep ID (FK)

Other Attributes:

- Surface Model ID (FK)

SML:

ENTITY Surface-Dim-3-SE-Rep
   CATEGORY BY Dim-3-Rep-Type OF Dim-3-Shape-Element-Representation
   Surface-Model "is used as" NONULL
END_Ende

EXPRESS:

ENTITY Surface_Dim_3_SE_Rep
   SUBTYPE OF ( Dim_3_Shape_Element_Representation );
   Definition : Surface_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface Model

Entity Number: none

Entity Definition: Needs to be defined in some other PDES reference model.

Business Rules: Every "Surface Model":
- is a "Geometric Model".
- is the context of zero, one, or more "Surface Area Reps".
- is the context of zero, one, or more "Surface Dim 0 SE Reps".
- is used as zero, one, or more "Surface Dim 3 SE Reps".
- is the context of zero, one, or more "Surface NM Area Reps".
- is the context of zero, one, or more "Surface Perimeter Reps".
- is the context of zero, one, or more "Surface Seam Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface NM Area Rep

Entity Number: INT-55

Entity Definition: The use of a "Subface" in a "Surface Model" to represent a "Nonmaximal Area Shape Element". In a surface model, a subface represents a portion of surface area that is necessarily nonmaximal because the subface is a subset of a face with a uniform underlying surface. (Note: The IPIM does not require that a subface be a proper subset of the containing face, but it seems pointless to have a subface identical to its containing face. For this reason, no provision is made here for representing a "Maximal Area Shape Element" with a subface.)

Business Rules: Every "Surface NM Area Rep":
- is a "Nonmaximal Area Representation".
- is in one, and only one, "Surface Model".
- is one, and only one, "Subface".

Primary Key Attributes:
Shape ID (FK)
Shape Elem ID (FK)
Nonmax Area Rep ID (FK)

Other Attributes:
Surface Model ID (FK)
Subface ID (FK)

SML:

ENTITY Surface-NM-Area-Rep
CATEGORY BY Nonmax-Area-Rep-Type
OF Nonmaximal-Area-Representation
Subface "is used as" NONULL
Surface-Model "is context of" NONULL
ENDE
Section 1: INTEGRATION CORE MODEL

EXPRESS:

ENTITY Surface_NM_Area_Rep
  SUBTYPE OF ( Nonmaximal_Area_Representation );
  Definition : Subface;
  Context : Surface_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface Perimeter Rep

Entity Number: INT-75

Entity Definition: The use of a "Loop" in a "Surface Model" to represent a "Perimeter Shape Element". In a surface model, a loop is employed as a boundary of a maximal face. Hence, a loop may represent the perimeter of an "Area Shape Element".

Business Rules: Every "Surface Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Surface Model".
- is one, and only one, "Loop".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

Other Attributes:

- Surface Model ID (FK)
- Loop ID (FK)

SML:

ENTITY Surface-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Loop "is used as" NONULL
  Surface-Model "is context of" NONULL
END

EXPRESS:

ENTITY Surface_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Loop;
  Context : Surface_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Surface Seam Rep

Entity Number: INT-65

Entity Definition: The use of an "Edge" in a "Surface Model" to represent a "Seam Shape Element". In a surface model, an edge is employed as a boundary element of a maximal or nonmaximal face and/or as an edge of a subface of a maximal or nonmaximal face. Even if the parent face is maximal, the edge may be nonmaximal; i.e., the edge may have an adjacent edge with the same underlying curve. So, an edge may represent any of the three categories of "Seam Shape Element" — edge, subedge, or interior seam.

Business Rules: Every "Surface Seam Rep":
• is a "Seam Representation".
• is in one, and only one, "Surface Model".
• is one, and only one, "Edge".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Seam Rep ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surface Model ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Edge ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

SML:

ENTITY Surface-Seam-Rep
    CATEGORY BY Seam-Rep-Type OF Seam-Representation
    Edge "is used as" NONULL
    Surface-Model "is context of" NONULL
END_ENTITY;

EXPRESS:

ENTITY Surface_Seam_Rep
    SUBTYPE OF ( Seam_Representation );
    Definition : Edge;
    Context : Surface_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Unstruct Geometry Area Rep

Entity Number: INT-51

Entity Definition: The use of a "Geometry" in an "Unstructured Geometry Model" to represent an "Area Shape Element".

Business Rules: Every "Unstruct Geometry Area Rep":
- is an "Area Representation".
- is in one, and only one, "Unstructured Geometry Model".
- is one, and only one, "Geometry".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Area Rep ID (FK)

Other Attributes:
- Unstruct Geom Model ID (FK)
- Geometry ID (FK)

SML:

ENTITY Unstruct-Geometry-Area-Rep
  CATEGORY BY Area-Rep-Type OF Area-Representation
  Geometry "is used as" NONULL
  Unstructured-Geometry-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY Unstruct_Geometry_Area_Rep
  SUBTYPE OF ( Area_Representation );
  Definition : Geometry;
  Context : Unstructured_Geometry_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Unstruct Geometry Dim 0 SE Rep

Entity Number: INT-90

Entity Definition: The use of a "Geometry" in an "Unstructured Geometry Model" to represent a "Dimensionality 0 Shape Element".

Business Rules: Every "Unstruct Geometry Dim 0 SE Rep":
- is a "Dim 0 Shape Element Representation".
- is in one, and only one, "Unstructured Geometry Model".
- is one, and only one, "Geometry".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Dim 0 Rep ID (FK)

Other Attributes:

- Unstruct Geom Model ID (FK)
- Geometry ID (FK)

SML:

ENTITY Unstruct-Geometry-Dim-0-SE-Rep
  CATEGORY BY Dim-0-Rep-Type OF Dim-0-Shape-Element-Representation
  Geometry "is used as" NONULL
  Unstructured-Geometry-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY Unstruct_Geometry_Dim_0_SE_Rep
  SUBTYPE OF ( Dim_0_Shape_Element_Representation );
  Definition : Geometry;
  Context : Unstructured_Geometry_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Unstruct Geometry Dim 3 SE Rep

Entity Number: INT-29

Entity Definition: The use of a "Geometry" in an "Unstructured Geometry Model" to represent a "Dimensionality 3 Shape Element".

Business Rules: Every "Unstruct Geometry Dim 3 SE Rep":
- is a "Dim 3 Shape Element Representation".
- is one, and only one, "Unstructured Geometry Model".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Dim 3 Rep ID (FK)

Other Attributes:
- Unstruct Geom Model ID (FK)

SML:

ENTITY Unstruct-Geometry-Dim-3-SE-Rep
   CATEGORY BY Dim-3-Rep-Type OF Dim-3-Shape-Element-Representation
   Unstructured-Geometry-Model "is used as" NONULL
END

EXPRESS:

ENTITY Unstruct_Geometry_Dim_3_SE_Rep
   SUBTYPE OF ( Dim_3_Shape_Element_Representation );
   Definition : Unstructured_Geometry_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Unstruct Geometry Perimeter Rep

Entity Number: INT-83

Entity Definition: The use of a "Geometry" in an "Unstructured Geometry Model" to represent a "Perimeter Shape Element".

Business Rules: Every "Unstruct Geometry Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Unstructured Geometry Model".
- is one, and only one, "Geometry".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Perim Rep ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstruct Geom Model ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Geometry ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

SML:

ENTITY Unstruct-Geometry-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Geometry "is used as" NONULL
  Unstructured-Geometry-Model "is context of" NONULL
END;

EXPRESSION:

ENTITY Unstruct_Geometry_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Geometry;
  Context : Unstructured_Geometry_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Unstruct Geometry Seam Rep

Entity Number: INT-72

Entity Definition: The use of a "Geometry" in an "Unstructured Geometry Model" to represent a "Seam Shape Element".

Business Rules: Every "Unstruct Geometry Seam Rep":
- is a "Seam Representation".
- is in one, and only one, "Unstructured Geometry Model".
- is one, and only one, "Geometry".

Primary Key Attributes:
- Shape ID (FK)
- Shape Elem ID (FK)
- Seam Rep ID (FK)

Other Attributes:
- Unstruct Geom Model ID (FK)
- Geometry ID (FK)

SML:

ENTITY Unstruct-Geometry-Seam-Rep
    CATEGORY BY Seam-Rep-Type OF Seam-Representation
    Geometry "is used as" NONULL
    Unstructured-Geometry-Model "is context of" NONULL
ENDE

EXPRESSION:

ENTITY Unstruct_Geometry_Seam_Rep
    SUBTYPE OF ( Seam_Representation );
    Definition : Geometry;
    Context : Unstructured_Geometry_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Unstructured Geometry Model

Entity Number: none

Entity Definition: Needs to be defined in some other PDES reference model.

Business Rules: Every "Unstructured Geometry Model":
- is a "Geometric Model".
- is the context of zero, one, or more "Unstructured Geometry Area Reps".
- is the context of zero, one, or more "Unstructured Geometry Dim 0 SE Reps".
- is used as zero, one, or more "Unstructured Geometry Dim 3 SE Reps".
- is the context of zero, one, or more "Unstructured Geometry NM Area Reps".
- is the context of zero, one, or more "Unstructured Geometry Perimeter Reps".
- is the context of zero, one, or more "Unstructured Geometry Seam Reps".
**Entity Name:** Vertex

**Entity Number:** TOP-2

**Entity Definition:** Defined in the PDES Topology reference model.

**Business Rules:**

Every "Vertex":
- is used as zero, one, or more "Brep Dim 0 SE Reps".
- is used as zero, one, or more "Full Rev SOR Corner Reps".
- is used as zero, one, or more "Full Rev SOR Int Loc Reps".
- is used as zero, one, or more "Partial Rev SOR Bdry Loc Reps".
- is used as zero, one, or more "Partial Rev SOR Corner Reps".
- is used as zero, one, or more "Partial Rev SOR Int Loc Reps".
- is used as zero, one, or more "SOLE Dim 0 SE Reps".
- is used as zero, one, or more "SOLE Vertex Seam Reps".
- is used as zero, one, or more "Full Rev SOR Vertex Perimeter Reps".
- is used as zero, one, or more "SOR Vertex Seam Reps".
- is used as zero, one, or more "Surface Dim 0 SE Reps".
- is used as zero, one, or more "Wireframe Dim 0 SE Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Voidless Volume Representation

Entity Number: INT-34

Entity Definition: The symbolic description of a "Voidless Volume Shape Element" in a "Geometric Model".

Business Rules: Every "Voidless Volume Representation":
- represents one, and only one, "Voidless Volume Shape Element".
- must be one, and only one, of the following:
  "Brep Volume Rep"
  "Full Rev SOR Volume Rep"
  "CSG Primitive Volume Rep"
  "Facetted Brep Volume Rep"
  "Half Space Volume Rep"
  "Partial Rev SOR Volume Rep"
  "SOLE Volume Rep"

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Volume Rep ID

The means of distinguishing each "Voidless Volume Representation" from all the others for the same "Voidless Volume Shape Element".

Other Attributes:

Volume Rep Type

The means of determining whether a "Voidless Volume Representation" is a(n):
  "Brep Volume Rep",
  "Full Rev SOR Volume Rep",
  "CSG Primitive Volume Rep",
  "Facetted Brep Volume Rep",
  "Half Space Volume Rep",
  "Partial Rev SOR Volume Rep", or
  "SOLE Volume Rep".

Page D3.1.1-167
Section 1: INTEGRATION CORE MODEL

SML:

ENTITY Voidless-Volume-Representation
  KEY
    Voidless-Volume-Shape-Element "is represented by"
    Volume-Rep-ID
  ENDK
  Volume-Rep-Type NONULL
ENDE

EXPRESS:

ENTITY Voidless_Volume_Representation
  SUPERTYPE OF ( Brep_Volume_Rep
     OR Full_Rev_SOR_Volume_Rep
     OR CSG_Primitive_Volume_Rep
     OR Facetted_Brep_Volume_Rep
     OR Half_Space_Volume_Rep
     OR Partial_Rev_SOR_Volume_Rep
     OR SOLE_Volume_Rep );
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Voidless Volume Shape Element

Entity Number: INT-5

Entity Definition: An "Object Shape Element" that contains no voids.

Business Rules: Every "Voidless Volume Shape Element":
• is an "Object Shape Element".
• is represented by zero, one, or more "Voidless Volume Representations".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td>(FK)</td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td>(FK)</td>
</tr>
</tbody>
</table>

Other Attributes: none

SML:

ENTITY Voidless-Volume-Shape-Element
    CATEGORY BY Object-SE-Type OF Object-Shape-Element
ENDE

EXPRESSION:

ENTITY Voidless_Volume_Shape_Element;
    Representations : LIST OF [ 0 : # ] OF Voidless_Volume_Representation;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Area Rep

Entity Number: INT-48

Entity Definition: The use of a "Loop" in a "Wireframe Model" to represent an "Area Shape Element".

Business Rules: Every "Wireframe Area Rep":
• is an "Area Representation".
• is in one, and only one, "Wireframe Model".
• is one, and only one, "Loop".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Area Rep ID (FK)

Other Attributes:

Wireframe Model ID (FK)
Loop ID (FK)

SML:

ENTITY Wireframe_Area_Rep
  CATEGORY BY Area_Rep_Type OF Area_Representation
  Loop "is used as" NONULL
  Wireframe_Model "is context of" NONULL
END_ENTITY;

EXPRESS:

ENTITY Wireframe_Area_Rep
  SUBTYPE OF ( Area_Representation );
  Definition : Loop;
  Context : Wireframe_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Dim 0 SE Rep
Entity Number: INT-87
Entity Definition: The use of a "Vertex" in a "Wireframe Model" to represent a "Dimensionality 0 Shape Element".

Business Rules: Every "Wireframe Dim 0 SE Rep":
- is a "Dim 0 Shape Element Representation".
- is in one, and only one, "Wireframe Model".
- is one, and only one, "Vertex".

Primary Key Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>(FK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape ID</td>
<td></td>
</tr>
<tr>
<td>Shape Elem ID</td>
<td></td>
</tr>
<tr>
<td>Dim 0 Rep ID</td>
<td></td>
</tr>
</tbody>
</table>

Other Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>(FK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireframe Model ID</td>
<td></td>
</tr>
<tr>
<td>Vertex ID</td>
<td></td>
</tr>
</tbody>
</table>

SML:

ENTITY Wireframe-Dim-0-SE-Rep
   CATEGORY BY Dim-0-Rep-Type OF Dim-0-Shape-Element-Representation
   Vertex "is used as" NONULL
   Wireframe-Model "is context of" NONULL
END

EXPRESSION:

ENTITY Wireframe_Dim_0_SE_Rep
   SUBTYPE OF ( Dim_0_Shape_Element_Representation );
   Definition : Vertex;
   Context : Wireframe_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Dim 3 SE Rep

Entity Number: INT-28

Entity Definition: The use of a "Wireframe Model" to represent a "Dimensionality 3 Shape Element".

Business Rules: Every "Wireframe Dim 3 SE Rep":
- is a "Dim 3 Shape Element Representation".
- is one, and only one, "Wireframe Model".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Dim 3 Rep ID (FK)

Other Attributes:

Wireframe Model ID (FK)

SML:

ENTITY Wireframe-Dim-3-SE-Rep
   CATEGORY BY Dim-3-Rep-Type OF Dim-3-Shape-Element-Representation
   Wireframe-Model "is used as" NONULL
ENDE

EXPRESS:

ENTITY Wireframe_Dim_3_SE_Rep
   SUBTYPE OF ( Dim_3_Shape_Element_Representation );
   Definition : Wireframe_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Model

Entity Number: none

Entity Definition: Needs to be defined in some other PDES reference model.

Business Rules: Every "Wireframe Model":
- is a "Geometric Model".
- is the context of zero, one, or more "Wireframe Area Reps".
- is the context of zero, one, or more "Wireframe Dim 0 SE Reps".
- is used as zero, one, or more "Wireframe Dim 3 SE Reps".
- is the context of zero, one, or more "Wireframe NM Area Reps".
- is the context of zero, one, or more "Wireframe Perimeter Reps".
- is the context of zero, one, or more "Wireframe Seam Reps".
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Perimeter Rep

Entity Number: INT-76

Entity Definition: The use of a "Loop" in a "Wireframe Model" to represent a "Perimeter Shape Element".

Business Rules: Every "Wireframe Perimeter Rep":
- is a "Perimeter Representation".
- is in one, and only one, "Wireframe Model".
- is one, and only one, "Loop".

Primary Key Attributes:

- Shape ID (FK)
- Shape Elem ID (FK)
- Perim Rep ID (FK)

Other Attributes:

- Wireframe Model ID (FK)
- Loop ID (FK)

SML:

ENTITY Wireframe-Perimeter-Rep
  CATEGORY BY Perim-Rep-Type OF Perimeter-Representation
  Loop "is used as" NONULL
  Wireframe-Model "is context of" NONULL
END_ENTITY;

EXPRESS:

ENTITY Wireframe_Perimeter_Rep
  SUBTYPE OF ( Perimeter_Representation );
  Definition : Loop;
  Context : Wireframe_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Wireframe Seam Rep

Entity Number: INT-66

Entity Definition: The use of an "Edge" in a "Wireframe Model" to represent a "Seam Shape Element".

Business Rules: Every "Wireframe Seam Rep":
- is a "Seam Representation".
- is in one, and only one, "Wireframe Model".
- is one, and only one, "Edge".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)
Seam Rep ID (FK)

Other Attributes:

Wireframe Model ID (FK)
Edge ID (FK)

SML:

ENTITY Wireframe-Seam-Rep
CATEGORY BY Seam-Rep-Type OF Seam-Representation
   Edge "is used as" NONULL
   Wireframe-Model "is context of" NONULL
ENDE

EXPRESS:

ENTITY Wireframe_Seam_Rep
   SUBTYPE OF ( Seam_Representation );
   Definition : Edge;
   Context : Wireframe_Model;
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Zone Shape Element

Entity Number: INT-11

Entity Definition: A "Dimensionality 2 Shape Element" that is the union of other "Dimensionality 2 Shape Elements".

Business Rules: Every "Zone Shape Element":
• is a "Dimensionality 2 Shape Element".
• consists of one or more "Zone Shape Element Components".

Primary Key Attributes:

Shape ID (FK)
Shape Elem ID (FK)

Other Attributes: none

SML:

ENTITY Zone-Shape-Element  
   CATEGORY BY Dim-2-SE-Type OF Dimensionality-2-Shape-Element  
ENDE

EXPRESS:

ENTITY Zone_Shape_Element  
   SUBTYPE OF ( Dimensionality_2_Shape_Element );  
   Components : LIST OF [ 1 : # ] Zone_Shape_Element_Component;  
END_ENTITY;
Section 1: INTEGRATION CORE MODEL

Entity Name: Zone Shape Element Component

Entity Number: INT-103

Entity Definition: A "Dimensionality 2 Shape Element" that is part of a "Zone Shape Element".

Business Rules: Every "Zone Shape Element Component":
- is one, and only one, "Dimensionality 2 Shape Element".
- is part of one, and only one, "Zone Shape Element".

Primary Key Attributes:

Shape ID (FK)
Zone SE ID (FK) – The role name for the Shape Elem ID of the "Zone Shape Element" that includes the "Zone Shape Element Component".
Comp SE ID (FK) – The role name for the Shape Elem ID of the "Dimensionality 2 Shape Element" that is used as the "Zone Shape Element Component".

Other Attributes: none

SML:

ENTITY Zone-Shape-Element-Component
   KEY
      Zone-Shape-Element "consists of" P
      ROLE Zone-SE-ID FOR Shape-Elem-ID
   Dimensionality-2-Shape-Element "is used as"
      ROLE Comp-SE-ID FOR Shape-Elem-ID
ENDK
ENDE

EXPRESS:

ENTITY Zone_Shape_Element_Component;
END_ENTITY;
2 NOMINAL SHAPE INFORMATION MODEL

2.1 Purpose

This document presents a conceptual model of information used to represent nominal shape. This includes geometry, solids, and topology. Geometry includes points, curves, and surfaces. Solids is scope down to the two most popular solid representation methods: boundary representation (B-rep) and constructive solid geometry (CSG). Topology is required by B-rep, but is defined independently from B-rep. The intention is that the same topology entities used in B-rep can be used for other purposes as well.

2.2 Method

These IDEF1X models were developed from the IPIM, as neither the Curves & Surfaces Committee nor the Solids Committee did model development in IDEF1X. Care has been taken to maintain conformity between the IPIM and the IDEF1X by using the same names and words as much as possible. There are two major departures from this philosophy, however. The first is that common attributes are not migrated to the leaf entities as in the IPIM. Instead the common attributes appear in the appropriate supertype. The second is that “coordinated lists” are not used here. Instead new entities were created to bundle the coordinated elements together explicitly. For example SEGMENT/GEO-44 bundles the elements from the COMPOSITE.CURVE/GEO-33 attributes “segments” and “senses”.

2.3 GEOMETRY

Entity Name: GEOMETRY

Entity Number: GEO-1

This the supertype of all geometric entities.

Primary Key Attributes

GEOMETRY.ID

Other Attributes

GEOMETRY.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY geometry
SECTION 2: NOMINAL SHAPE INFORMATION MODEL

SUPERTYPE OF (point XOR
    vector XOR
    axis_placement XOR
    transformation_matrix XOR
    curve XOR
    surface)

END_ENTITY;

Entity Name: POINT
Entity Number: GEO-2

A point is a location in some coordinate space.

Primary Key Attributes
POINT.ID

Other Attributes
POINT.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY point
    SUPERTYPE OF (cartesian_point XOR
        point_on_curve XOR
        point_on_surface)
    SUBTYPE OF (geometry);
END_ENTITY;

Entity Name: VECTOR
Entity Number: GEO-3

Collects the two types of vector: DIRECTION/GEO-14 and
VECTOR.WITH.MAGNITUDE/GEO-17.

Primary Key Attributes
VECTOR.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

**Other Attributes**

VECTOR.TYPE (discriminator)

**Business Rules**

**EXPRESS Specification**

```express
ENTITY vector
  SUPERTYPE OF (direction XOR
               vector_with_magnitude)
  SUBTYPE OF (geometry);
END_ENTITY;
```

**Entity Name:** AXIS_PLACEMENT

**Entity Number:** GEO-4

The local environment for definition of a geometry entity. It locates the entity to be defined and gives its orientation.

**Primary Key Attributes**

AXIS_PLACEMENT.ID

**Other Attributes**

AXIS_PLACEMENT.TYPE (discriminator)

CARTESIAN POINT.ID (FK)
  The location.

AXIS.DIRECTION JD (FK)
  Direction of local Z axis.

**Business Rules**

**EXPRESS Specification**

```express
ENTITY axis_placement
  SUPERTYPE OF (axis1_placement XOR
                 axis2_placement)
  SUBTYPE OF (geometry);
END_ENTITY;
```
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: TRANSFORMATION_MATRIX

Entity Number: GEO-5

A general geometric transformation including translation, rotation, mirroring and scaling.

Primary Key Attributes
TRANSFORMATION_MATRIX.ID

Other Attributes

AXIS1.THREE_SPACE_DIRECTION.ID (FK)
The approximate direction of the required X axis.

AXIS2.THREE_SPACE_DIRECTION.ID (FK)
The approximate direction of the required Y axis.

AXIS3 THREE_SPACE_DIRECTION.ID (FK)
The exact direction of the required Z axis.

CARTESIAN_POINT.ID (FK)
The required translation.

SCALE
Number value of scaling factor.

Business Rules

EXPRESSION Specification

ENTITY transformation
  SUBTYPE OF (geometry);
  axis1 : OPTIONAL three_space_direction;
  axis2 : OPTIONAL three_space_direction;
  axis3 : OPTIONAL three_space_direction;
  local_origin : OPTIONAL cartesian_point;
  scale : OPTIONAL REAL;
END_ENTITY;

Entity Name: CURVE

Entity Number: GEO-6

The class of all curves.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
CURVE.ID

Other Attributes
CURVE.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY curve
  SUPERTYPE OF (line IOR conic IOR bounded_curve IOR curve_on_surface IOR offset_curve)
  SUBTYPE OF (geometry);
END_ENTITY;

Entity Name: SURFACE
Entity Number: GEO-7

The general class of all surfaces.

Primary Key Attributes
SURFACE.ID

Other Attributes
SURFACE.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY surface
  SUPERTYPE OF (elementary_surface IOR swept_surface IOR bounded_surface IOR offset_surface IOR rectangular_composite_surface)
  SUBTYPE OF (geometry);
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: CARTESIAN_POINT

Entity Number: GEO-8

A point defined in a cartesian coordinate system.

Primary Key Attributes
CARTESIAN_POINT_ID

Other Attributes
CARTESIAN_POINT.TYPE (discriminator)
X.COORDINATE
Y.COORDINATE

Business Rules

EXPRESS Specification

ENTITY cartesian_point
SUPERTYPE OF (cartesian_two_coordinate OR
    cartesian_three_coordinate)
SUBTYPE OF (point);
END_ENTITY;

Entity Name: CARTESIAN_TWO.POINT

Entity Number: GEO-10

A point defined in two dimensional Euclidean space.

Primary Key Attributes
CARTESIAN_TWO.POINT_ID

Other Attributes

Business Rules

EXPRESS Specification

ENTITY cartesian_two_coordinate

12
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: CARTESIAN_THREE_POINT

Entity Number: GEO-11

A point defined in three dimensional Euclidean space.

Primary Key Attributes

CARTESIAN_THREE_COORDINATE_ID

Other Attributes

Z_COORDINATE

Business Rules

EXPRESS Specification

ENTITY cartesian_three_coordinate
    SUBTYPE OF (cartesian_point);
    x_coordinate : REAL;
    y_coordinate : REAL;
    z_coordinate : REAL;
END_ENTITY;

Entity Name: POINT_ON_CURVE

Entity Number: GEO-12

A point located in the parametric space of a curve.

Primary Key Attributes

POINT_ON_CURVE_ID

Other Attributes

CURVE_ID (FK)

POINT_PARAMETER
    The parametric value of the point.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

**EXPRESS Specification**

```express
ENTITY point_on_curve_coordinate
    SUBTYPE OF (point);
    basis_curve : curve;
    point_parameter : REAL;
END_ENTITY;
```

Entity Name: POINT_ON_SURFACE

Entity Number: GEO-13

A point located in the parametric space of a surface.

**Primary Key Attributes**

POINT_ON_SURFACE_ID

**Other Attributes**

SURFACE_ID (FK)

POINT PARAMETER_1
   The first parameter value of the point location.

POINT PARAMETER_2
   The second parameter value of the point location.

Business Rules

**EXPRESS Specification**

```express
ENTITY point_on_surface
    SUBTYPE OF (point);
    basis_surface : surface;
    point_parameter_1 : REAL;
    point_parameter_2 : REAL;
END_ENTITY;
```

Entity Name: DIRECTION

Entity Number: GEO-14

A geometric direction.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

**DIRECTION ID**

Other Attributes

**DIRECTION_TYPE** (discriminator)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td>The direction ratio with respect to the X axis.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>The direction ratio with respect to the Y axis.</td>
</tr>
</tbody>
</table>

Business Rules

**EXPRESS Specification**

```express
ENTITY direction
  SUPERTYPE OF (two_space_direction IOR three_space_direction)
  SUBTYPE OF (vector);
END_ENTITY;
```

Entity Name: **TWO_SPACE DIRECTION**

Entity Number: **GEO-15**

A direction vector in Euclidean two space.

Primary Key Attributes

**TWO_SPACE_DIRECTION_ID**

Other Attributes

None

Business Rules

**EXPRESS Specification**

```express
ENTITY two_space_direction
  SUBTYPE OF (direction);
  x : REAL;
  y : REAL;
```

15
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: THREE_SPACE_DIRECTION
Entity Number: GEO-16
A direction vector in Euclidean three space.

Primary Key Attributes
THREE_SPACE_DIRECTION_ID

Other Attributes
z
The direction ratio with respect to the Z axis.

Business Rules

EXPRESS Specification
ENTITY three_space_direction
  SUBTYPE OF (direction);
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;

Entity Name: VECTOR_WITH_MAGNITUDE
Entity Number: GEO-17
A direction vector and the magnitude of the vector.

Primary Key Attributes
VECTOR_WITH_MAGNITUDE_ID

Other Attributes
DIRECTION_ID (FK)
The orientation.

MAGNITUDE
Real value which is the magnitude of the vector.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESSION Specification

ENTITY vector_with_magnitude
  SUBTYPE OF (vector);
  orientation : direction;
  magnitude : REAL;
END_ENTITY;

Entity Name: AXIS1_PLACEMENT
Entity Number: GEO-18

An axisymmetric local coordinate system.

Primary Key Attributes
AXIS1_PLACEMENT.ID

Other Attributes
None

Business Rules

EXPRESSION Specification

ENTITY axis1_placement
  SUBTYPE OF (axis_placement);
  location : point;
  axis : OPTIONAL direction;
END_ENTITY;

Entity Name: AXIS2_PLACEMENT
Entity Number: GEO-19

A complete local, cartesian coordinate system.

Primary Key Attributes
AXIS2_PLACEMENT.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

REF.DIRECTION.ID (FK)
The approximate direction of the local X axis.

Business Rules

EXPRESS Specification

ENTITY axis2_placement
   SUBTYPE OF (axis_placement);
   location : point;
   axis : OPTIONAL direction;
   ref_direction : OPTIONAL direction;
END_ENTITY;

Entity Name:_LINE

Entity Number:_GEO-20

An unbounded straight line. The positive direction of the line is in the positive direction of its direction vector.

Primary Key Attributes

LINE.ID

Other Attributes

CARTESIAN POINT.ID (FK)
Location of the line.

DIRECTION.ID (FK)
The direction of the line.

Business Rules

EXPRESS Specification

ENTITY line
   SUBTYPE OF (curve);
   pnt : point;
   dir : direction;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: CONIC

Entity Number: GEO-21

A conical curve. Namely a circle, ellipse, hyperbola, or parabola.

Primary Key Attributes

CONIC.ID

Other Attributes

CONIC.TYPE (discriminator)

AXIS2Placement.ID (FK)

The location and orientation of the conic.

Business Rules

EXPRESS Specification

ENTITY conic

SUPERTYPE OF (circle XOR ellipse XOR hyperbola XOR parabola);

SUBTYPE OF (curve);

END_ENTITY;

Entity Name: BOUNDED CURVE

Entity Number: GEO-22

A class of curves with finite length. The length may be finite naturally (as a circle) or finite due to trimming.

Primary Key Attributes

BOUNDED CURVE.ID

Other Attributes

BOUNDED CURVE.TYPE (discriminator)
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY bounded_curve
  SUPERTYPE OF (line_segment IOR bezier_curve IOR b_spline_curve IOR trimmed_curve IOR composite_curve);
  SUBTYPE OF (curve);
END_ENTITY;

Entity Name: CURVE_ON_SURFACE

Entity Number: GEO-23

The class of curves whose members are restricted to being contained in a specified surface.

Primary Key Attributes

CURVE_ON_SURFACE.ID

Other Attributes

CURVE_ON_SURFACE.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY curve_on_surface
  SUPERTYPE OF (pcurve IOR surface_curve IOR intersection_curve
  IOR composite_curve_on_surface);
  SUBTYPE OF (curve);
END_ENTITY;

Entity Name: OFF_SET

Entity Number: GEO-24

A curve a constant distance from a basis curve.

Primary Key Attributes

OFFSET_CURVE.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

CURVE ID (FK)
The curve that is being offset.

DISTANCE
The distance of the offset from the basis curve.

SELF_INTERSECTION
Three value flag to indicate whether curve self intersects:

TRUE
Curve does self intersect.

FALSE
Curve does not self intersect.

UNDEFINED
Whether curve self intersects or not is unknown.

OFFSET_CURVE_TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY offset_curve
  SUPERTYPE OF (d2_offset_curve OR d3_offset_curve)
  SUBTYPE OF (curve);
END_ENTITY;

Entity Name: CIRCLE

Entity Number: GEO-25

A complete circle.

Primary Key Attributes

CIRCLE_ID

Other Attributes

RADIUS
Real number value which is the radius of the circle.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY circle;
  SUBTYPE OF (conic);
  radius : REAL;
  position : axis2_placement;
END_ENTITY;

Entity Name: ELLIPSE

Entity Number: GEO-26

A complete ellipse.

Primary Key Attributes

ELLIPSE.ID

Other Attributes

SEMI_AXIS_1
  Real number value which is the major radius.

SEMI_AXIS_2
  Real number value which is the minor radius.

Business Rules

EXPRESS Specification

ENTITY ellipse;
  SUBTYPE OF (conic);
  semi_axis_1 : REAL;
  semi_axis_2 : REAL;
  position : axis2_placement;
END_ENTITY;

Entity Name: HYPERBOLA

Entity Number: GEO-27

An unbounded hyperbola.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

HYPERBOLA_ID

Other Attributes

SEMI_AXIS
Real number value which is the length of the "major radius" of the hyperbola.

SEMI(IMAG)_AXIS
Real number value which is the length of the "minor radius".

Business Rules

EXPRESS Specification

ENTITY hyperbola;
  SUBTYPE OF (conic);
  semi_axis : REAL;
  semi_imag_axis : REAL;
  position : axis2_placement;
END_ENTITY;

Entity Name:   PARABOLA

Entity Number: GEO-28

An unbounded parabola.

Primary Key Attributes

PARABOLA_ID

Other Attributes

FOCAL_DIST
Real number value which is the distance of the focal point from the vertex point.

Business Rules

EXPRESS Specification

ENTITY parabola;
  SUBTYPE OF (conic);
Section 2: NOMINAL SHAPE INFORMATION MODEL

focal_dist : REAL;
position : axis2_placement;
END_ENTITY;

Entity Name: LINE_SEGMENT

Entity Number: GEO-29

A straight line segment. This entity is included for efficiency. Alternately a TRIMMED_CURVE/GEO-32 trimming a LINE/GEO-20 could be used.

Primary Key Attributes
LINE_SEGMENT_ID

Other Attributes
FIRST_POINT_ID (FK) The starting location of the line segment.
SECOND_POINT_ID (FK) The ending location of the line segment.

Business Rules

EXPRESSION Specification

ENTITY line_segment
    SUBTYPE OF (bounded_curve);
    first_point : cartesian_point;
    last_point : cartesian_point;
END_ENTITY;

Entity Name: BEZIER_CURVE

Entity Number: GEO-30

A Bezier curve. It should be noted that every Bezier curve has an equivalent representation as a B-spline curve, but not every B-spline curve can be represented as a single Bezier curve.

Primary Key Attributes
BEZIER_CURVE_ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

RATIONAL
Logical value which indicates whether the curve is rational (TRUE) or simple polynomial (FALSE).

UNIFORM
Logical value which indicates whether the knot set is uniform (TRUE) or not uniform (FALSE).

DEGREE
Integer value which is the algebraic degree of the basis functions.

FORM NUMBER
Symbol used to identify particular type of curve.

SELF_INTERSECTION
Three value flag to indicate whether curve self intersects:

TRUE
Curve does self intersect.

FALSE
Curve does not self intersect.

UNDEFINED
Whether curve self intersects or not is unknown.

Business Rules

EXPRESS Specification

ENTITY bezier_curve
  SUBTYPE OF (bounded_curve);
  rational : LOGICAL;
  degree : INTEGER;
  control_points : ARRAY [0:k] OF cartesian_point;
  weights : OPTIONAL ARRAY {0:k} OF REAL;
  form_number : OPTIONAL bspline_curve_form;
  self_intersect : true_false_or_undefined;
END_ENTITY;

Entity Name: B_SPLINE_CURVE

Entity Number: GEO-31

A b-spline curve. The b-spline may be either rational or not rational.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
B.SPLINE.CURVE_ID

Other Attributes
RATIONAL
Logical value which indicates whether the curve is rational (TRUE) or simple polynomial (FALSE).

UNIFORM
Logical value which indicates whether the knot set is uniform (TRUE) or not uniform (FALSE).

DEGREE
Integer value which is the algebraic degree of the basis functions.

FORM_NUMBER
Enumeration selection used to identify the particular type of curve.

SELF INTERSECTION
Three value flag to indicate whether curve self intersects:
TRUE
Curve does self intersect.
FALSE
Curve does not self intersect.
UNDEFINED
Whether curve self intersects or not is unknown.

Business Rules

EXPRESS Specification

ENTITY b_spline_curve
SUBTYPE OF (bounded_curve);
rational : LOGICAL;
uniform : LOGICAL;
degree : INTEGER;
control_points : ARRAY [0:?] OF cartesian_point;
weights : OPTIONAL ARRAY {0:?} OF REAL;
form_number : OPTIONAL b spline_curve_form;
self_intersect : true_false_or_undefined;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

**Entity Name:** TRIMMED.CURVE

**Entity Number:** GEO-32

A portion of a curve.

**Primary Key Attributes**

TRIMMED.CURVE.ID

**Other Attributes**

TRIMMED.CURVE.TYPE (discriminator)
CURVE.ID (FK)
  Curve being trimmed.
SENSE
  Logical value which indicates whether the curve is being traversed in the direction of increasing parametric value (TRUE) or decreasing parametric value (FALSE).

**Business Rules**

**EXPRESS Specification**

```
ENTITY trimmed_curve
  SUBTYPE OF (bounded_curve);
  basis_curve : curve;
  parameter_1 : OPTIONAL REAL;
  parameter_2 : OPTIONAL REAL;
  point_1 : OPTIONAL cartesian_point;
  point_2 : OPTIONAL cartesian_point;
  sense : LOGICAL;
END_ENTITY;
```

**Entity Name:** COMPOSITE.CURVE

**Entity Number:** GEO-33

A collection of curves joined end to end.

**Primary Key Attributes**

COMPOSITE.CURVE.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

CLOSED CURVE
Logical value which indicates whether the curve is closed (TRUE) or open (FALSE).

SELF.INTERSECTION
Three value flag to indicate whether curve self intersects:

TRUE
Curve does self intersect.
FALSE
Curve does not self intersect.
UNDEFINED
Whether curve self intersects or not is unknown.

Business Rules

EXPRESS Specification

ENTITY composite_curve
  SUBTYPE OF (bounded_curve);
  closed_curve : LOGICAL;
  segments : LIST [1:0] OF bounded_curve;
  senses : ARRAY [1:0] OF LOGICAL:
  transitions : ARRAY [1:0] OF curve_transition_code;
  param_range : OPTIONAL ARRAY [1:0] OF REAL;
END_ENTITY;

Entity Name: CONTROL_POINT_BEZIER

Entity Number: GEO-34
A control point a BEZIER_CURVE/GEO-30.

Primary Key Attributes
BEZIER_CURVE.ID (FK)
CONTROL_POINT_BEZIER.ID

Other Attributes

CARTESIAN_POINT.ID (FK)
WEIGHTS
Real number value of weight associated with this control point.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

See attribute “control.points” in EXPRESS for BEZIER_CURVE/GEO-30.

Entity Name: CONTROL POINT ORDER BEZIER

Entity Number: GEO-35

IDEFIX defaults to unordered sets. This entity makes CONTROL POINT BEZIER/GEO-34 an ordered list.

Primary Key Attributes

BEZIER_CURVE_ID (FK)
PREDECESSOR CONTROL POINT BEZIER (FK)
SUCCESSOR CONTROL POINT BEZIER (FK)

Other Attributes

None

Business Rules

EXPRESS Specification

See attribute “control.points” in EXPRESS for BEZIER_CURVE/GEO-30.

Entity Name: KNOT

Entity Number: GEO-36

A knot for a B_SPLINE_CURVE/GEO-31.

Primary Key Attributes

B_SPLINE_CURVE_ID (FK)
KNOT ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

MULTIPLICITY
   Integer value of multiplicity of this knot.

KNOT.VALUE
   Real number value of this knot which is one of a set of knots used to define the B-spline basis function.

Business Rules

EXPRESS Specification

See attributes "knots" and "knot multiplicities" in the EXPRESS for B.SPLINE.CURVE/GEO-31.

Entity Name: CONTROL.POINT

Entity Number: GEO-37

A control point for a B.SPLINE.CURVE/GEO-31.

Primary Key Attributes

B.SPLINE.CURVE.ID (FK)
CONTROL.POINT.ID

Other Attributes

CARTESIAN.POINT.ID (FK)
WEIGHTS
   Real number value of weight associated with this control point.

Business Rules

EXPRESS Specification

See attributes "control.points" and "weights" in the EXPRESS for B.SPLINE.CURVE/GEO-31.

Entity Name: CONTROL.POINT.ORDER

Entity Number: GEO-38

IDEF1X defaults to unordered sets. This entity makes CONTROL.POINT/GEO-37 an ordered list.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

B.SPLINE.CURVE.ID (FK)
PREDECESSOR.CONTROL.POINT.ID (FK)
SUCCESSOR.CONTROL.POINT.ID (FK)

Other Attributes

None

Business Rules

EXPRESS Specification

See attributes "control points" and "weights" in the EXPRESS for B.SPLINE.CURVE/GEO-31.

Entity Name: POINT.POINT.TRIMMED.CURVE

Entity Number: GEO-39

A curve trimmed with two cartesian points.

Primary Key Attributes

TRIMMED CURVE.ID (FK)

Other Attributes

POINT.1.CARTESIAN.POINT.ID (FK)
POINT.2.CARTESIAN.POINT.ID (FK)

Business Rules

EXPRESS Specification

See the EXPRESS for TRIMMED.CURVE/GEO-32.

Entity Name: POINT.PARAM.TRIMMED.CURVE

Entity Number: GEO-40

A curve trimmed with one cartesian point and one parametric point.
## Section 2: NOMINAL SHAPE INFORMATION MODEL

### Primary Key Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIMMED.CURVE.ID (FK)</td>
<td></td>
</tr>
</tbody>
</table>

### Other Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARTESIAN POINT.ID (FK)</td>
<td>First trimming point.</td>
</tr>
<tr>
<td>PARAMETER.2</td>
<td>Real number value which is the second trimming point in the parametric space of the basis curve.</td>
</tr>
<tr>
<td>CARTESIAN POINT.ID (FK)</td>
<td>Second trimming point.</td>
</tr>
</tbody>
</table>

### Business Rules

**EXPRESS Specification**

See the EXPRESS for TRIMMED.CURVE/GEO-32.

**Entity Name:** PARAM.POINT.TRIMMED.CURVE

**Entity Number:** GEO-41

A curve trimmed with one cartesian point and one parametric point.

### Primary Key Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIMMED.CURVE.ID (FK)</td>
<td></td>
</tr>
</tbody>
</table>

### Other Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAMETER.1</td>
<td>Real number value which is the first trimming point in the parametric space of the basis curve.</td>
</tr>
<tr>
<td>CARTESIAN POINT.ID (FK)</td>
<td>Second trimming point.</td>
</tr>
</tbody>
</table>

### Business Rules

**EXPRESS Specification**

See the EXPRESS for TRIMMED.CURVE/GEO-32.

**Entity Name:** PARAM.PARAM.TRIMMED.CURVE

**Entity Number:** GEO-42

A curve trimmed with two parametric values.
Primary Key Attributes

PARAMETER.1
Real number value which is the first trimming point in the parametric space of the basis curve.

PARAMETER.2
Real number value which is the second trimming point in the parametric space of the basis curve.

Other Attributes
None

Business Rules

EXPRESS Specification
See the EXPRESS for TRIMMED.CURVE/GEO-32.

Entity Name: KNOT.ORDER

Entity Number: GEO-43
IDEFIX defaults to unordered sets. This entity makes KNOT/GEO-36 an ordered list.

Primary Key Attributes
B.SPLINE.CURVE.ID (FK)
PREDECESSOR.KNOT.ID (FK)
SUCCESSOR.KNOT.ID (FK)

Other Attributes
None

Business Rules

EXPRESS Specification
See attributes “control.points” and “weights” in the EXPRESS for B.SPLINE.CURVE/GEO-31.

Entity Name: SEGMENT

Entity Number: GEO-44
A component curve of a COMPOSITE.CURVE/GEO-33.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

COMPOSITE.CURVE.ID (FK)
SEGMENT.ID

Other Attributes

BOUNDED.CURVE.ID (FK)
   A component curve.

SENSE
   Logical value which indicates whether the sense of component curve agrees or differs from the sense of the composite curve.

Business Rules

EXPRESSION Specification
See attribute "segments" in EXPRESSION for COMPOSITE.CURVE/GEO-33.

Entity Name: SEGMENT.ORDER

Entity Number: GEO-45
IDEFIX defaults to unordered sets. This entity makes SEGMENT/GEO-44 an ordered list.

Primary Key Attributes

COMPOSITE.CURVE.ID (FK)
PREDECESSOR_SEGMENT.ID (FK)
SUCCESSOR_SEGMENT.ID (FK)

Other Attributes

TRANSITION
   Enumeration selection for form of transition between predecessor and successor.

PARAM RANGE
   Real number value which is the composite curve parameter value at this join.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

**EXPRESS Specification**

See attribute “segments” in EXPRESS for COMPOSITE.CURVE/GEO-33.

Entity Name: PCURVE

Entity Number: GEO-46

A curve defined in the parametric space of a surface.

**Primary Key Attributes**

PCURVE.ID

**Other Attributes**

SURFACE.ID (FK)

The surface on which the curve lies.

CURVE.ID (FK)

A 2D curve.

**Business Rules**

**EXPRESS Specification**

ENTITY pcurve
    SUBTYPE OF (curve_on_surface);
    basis_surface : surface;
    basis_curve : curve;
END_ENTITY;

Entity Name: SURFACE.CURVE

Entity Number: GEO-47

A bounded curve on a surface.

**Primary Key Attributes**

SURFACE.CURVE.ID
Other Attributes

CURVE.ID (FK)
The curve which is the three dimensional image of the surface curve.

SURFACE.ID (FK)
The surface on which the curve lies.

MASTER
Enumeration selection for representation “preferred” by sending system.

Business Rules

EXPRESS Specification

ENTITY surface_curve
  SUBTYPE OF (curve_on_surface);
  surface_1 : surface;
  curve_1 : curve;
  pcurve_s1 : OPTIONAL pcurve;
  master : OPTIONAL enumeration_curve1_pcurve1;
END_ENTITY;

Entity Name: SURFACE.CURVE.PCURVE

Entity Number: GEO-48

Captures information that a SURFACE.CURVE/GEO-47 may optionally have an associated PCURVE/GEO-46. An associated PCURVE/GEO-46 is the two dimensional parametric image of the SURFACE.CURVE/GEO-47.

Primary Key Attributes

SURFACE CURVE.ID (FK)

Other Attributes

PCURVE.ID (FK)
The curve which is the two dimensional image of the surface curve.

Business Rules

EXPRESS Specification

See attribute “pcurve_s1” in EXPRESS for SURFACE.CURVE/GEO-47.
**Section 2: NOMINAL SHAPE INFORMATION MODEL**

**Entity Name:** INTERSECTION_CURVE

**Entity Number:** GEO-49

An intersection curve is defined by intersecting two surfaces. The multiple representation permits the receiving system to recompute the intersection curve if it has the capability.

**Primary Key Attributes**

INTERSECTION_CURVE.ID

**Other Attributes**

CURVE_ID (FK)

MASTER_REPRESENTATION

Enumeration selection for representation "preferred" by sending system.

SELF_INTERSECTION

Three value flag to indicate whether curve self intersects:

TRUE

Curve does self intersect.

FALSE

Curve does not self intersect.

UNDEFINED

Whether curve self intersects or not is unknown.

**Business Rules**

**EXPRESS Specification**

```express
ENTITY intersection_curve
  SUBTYPE OF (curve_on_surface);
  pcurve_s1 : OPTIONAL pcurve;
  surface_s1 : OPTIONAL surface;
  pcurve_s2 : OPTIONAL pcurve;
  surface_s2 : OPTIONAL surface;
  basis_curve : curve;
  master_representation : OPTIONAL intersection Enumeration;
  self_intersection : true_false_or_undefined;
END_ENTITY;
```
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: COMPOSITE CURVE ON SURFACE

Entity Number: GEO-50

An assembly of curves on a surface.

Primary Key Attributes

COMPOSITE CURVE ON SURFACE ID

Other Attributes

SURFACE ID (FK)

CLOSED CURVE

Logical value which indicates whether the curve is closed (TRUE) or open (FALSE).

SELF INTERSECTION

Three value flag to indicate whether curve self intersects:

TRUE

Curve does self intersect.

FALSE

Curve does not self intersect.

UNDEFINED

Whether curve self intersects or not is unknown.

Business Rules

EXPRESS Specification

ENTITY composite_curve_on_surface

SUBTYPE OF (curve_on_surface);

basis_surface : surface;

closed_curve : LOGICAL;

segments : LIST [1:*] OF curve_on_surface;

senses : ARRAY [1:*] OF LOGICAL;

transitions : ARRAY [1:*] OF curve_transition_code;

END_ENTITY;

Entity Name: PCURVE PCURVE INTERSECTION

Entity Number: GEO-51

One of three definition choices of the INTERSECTION CURVE GEO-49. In this choice the intersection curve is defined with two PCURVE GEO-46.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

INTERSECTION.CURVE.ID (FK)

Other Attributes

PCURVE.S1.PCURVE.ID (FK)
   The intersection curve of the two surfaces defined in the parametric space of the first surface.

PCURVE.S2.PCURVE.ID (FK)
   The intersection curve of the two surfaces defined in the parametric space of the second surface.

Business Rules

EXPRESS Specification

See EXPRESS for INTERSECTION.CURVE/GEO-49. This IDEF1X entity represents one valid set of options.

Entity Name: PCURVE_SURFACE_INTERSECTION

Entity Number: GEO-52

One of three definition choices of the INTERSECTION.CURVE/GEO-49. In this choice the intersection curve is defined with one PCURVE/GEO-46 and one SURFACE/GEO-7.

Primary Key Attributes

INTERSECTION.CURVE.ID (FK)

Other Attributes

PCURVE.ID (FK)
   The intersection curve of the two surfaces defined in the parametric space of the first surface.

SURFACE.ID (FK)
   The second surface of the intersection.

Business Rules

EXPRESS Specification

See EXPRESS for INTERSECTION.CURVE/GEO-49. This IDEF1X entity represents one valid set of options.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: SURFACE SURFACE INTERSECTION

Entity Number: GEO-53

One of three definition choices of the INTERSECTION CURVE/GEO-49. In this choice the intersection curve is defined with two SURFACE/GEO-7.

Primary Key Attributes

INTERSECTION CURVE ID (FK)

Other Attributes

SURFACE 51 SURFACE ID (FK)
The first surface of the intersection.

SURFACE 52 SURFACE ID (FK)
The second surface of the intersection.

Business Rules

EXPRESS Specification
See EXPRESS for INTERSECTION CURVE/GEO-49. This IDEF1X entity represents one valid set of options.

Entity Name: ON SURFACE SEGMENT

Entity Number: GEO-54

The use of a CURVE ON SURFACE/GEO-23 as a component curve of a COMPOSITE CURVE ON SURFACE/GEO-50.

Primary Key Attributes

COMPOSITE CURVE ON SURFACE ID (FK)

ON SURFACE SEGMENT ID

Other Attributes

CURVE ON SURFACE ID (FK)
A component curve.

SENSE
Logical value which indicates whether the sense of this component curve agrees (TRUE) with the composite curve or not (FALSE).
**Section 2: NOMINAL SHAPE INFORMATION MODEL**

Business Rules

**EXPRESS Specification**

See attribute "segments" in EXPRESS for COMPOSITE_CURVE.ON_SURFACE/GEO-50.

Entity Name: ON_SURFACE_SEGMENT_ORDER

Entity Number: GEO-55

IDEFIX defaults to unordered sets. This entity makes ON_SURFACE_SEGMENT/GEO-54 an ordered list.

**Primary Key Attributes**

COMPOSITE_CURVE.ON_SURFACE.ID (FK)

PREDECESSOR.ON_SURFACE_SEGMENT.ID (FK)

SUCCESSOR.ON_SURFACE_SEGMENT.ID (FK)

**Other Attributes**

TRANSITION

Enumeration selection for the transition code between predecessor curve and successor curve.

This applies to the 3-D representations of curves, not to pcurves.

Business Rules

**EXPRESS Specification**

See attribute "segments" in EXPRESS for COMPOSITE_CURVE.ON_SURFACE/GEO-50.

Entity Name: TWO_D_OFFSET_CURVE

Entity Number: GEO-56

This defines a simple plane-offset curve by offsetting by DISTANCE along the normal to BASIS_CURVE in the plane of BASIS_CURVE.

**Primary Key Attributes**

2D_OFFSET_CURVE.ID

**Other Attributes**

None
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY d2_offset_curve
  SUBTYPE OF (offset_curve);
  basis_curve : curve;
  distance : REAL;
  self_intersect : true_false_or_undefined;
END_ENTITY;

Entity Name: THREE_D_OFFSET_CURVE

Entity Number: GEO-57

A curve generated as an offset from another curve in three space.

Primary Key Attributes

3D_OFFSET_CURVE.ID

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY d3_offset_curve
  SUBTYPE OF (offset_curve);
  basis_curve : curve;
  distance : REAL;
  self_intersect : true_false_or_undefined;
  ref_direction : direction;
END_ENTITY;

Entity Name: ELEMENTARY_SURFACE

Entity Number: GEO-58

The class of elementary surfaces. Namely plane, cylinder, cone, sphere and torus.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
ELEMENTAL_SURFACE_ID

Other Attributes
ELEMENTAL_SURFACE_TYPE (discriminator)
AXIS2_PLACEMENT_ID (FK)
  Location and orientation of the surface.

Business Rules

EXPRESS Specification

ENTITY elementary_surface
  SUPERTYPE OF (plane XOR
  cylindrical_surface XOR
  conical_surface XOR
  spherical_surface XOR
  toroidal_surface)
  SUBTYPE OF (surface);
END_ENTITY;

Entity Name: BEZIER_SURFACE

Entity Number: GEO-59

A Bezier surface.

Primary Key Attributes
BEZIER_SURFACE_ID

Other Attributes

U.RATIONAL
  Logical value which indicates whether the surface is rational (TRUE) or simple polynomial (FALSE) in first (u) parametric direction.

V.RATIONAL
  Logical value which indicates whether the surface is rational (TRUE) or simple polynomial (FALSE) in second (v) parametric direction.
Section 2: NOMINAL SHAPE INFORMATION MODEL

U.UNIFORM
Logical value which indicates whether the knot set is uniform (TRUE) or not uniform (FALSE) in "u".

V.UNIFORM
Logical value which indicates whether the knot set is uniform (TRUE) or not uniform (FALSE) in "v".

U.DEGREE
Integer value which is the algebraic degree of the basis functions in "u".

V.DEGREE
Integer value which is the algebraic degree of the basis functions in "v".

FORM_NUMBER
Enumeration selection used to identify the particular type of surface.

Business Rules

EXPRESS Specification

ENTITY bezier_surface
  SUBTYPE OF (bounded_surface):
    u_rational : LOGICAL;
    v_rational : LOGICAL;
    u_uniform : LOGICAL;
    v_uniform : LOGICAL;
    u_degree : INTEGER;
    v_degree : INTEGER;
    u_upper : INTEGER;
    v_upper : INTEGER;
    control_points : ARRAY [0..u_upper] OF
      ARRAY [0..v_upper] OF cartesian_point;
    weights : OPTIONAL ARRAY [0..u_upper] OF
      ARRAY [0..v_upper] OF REAL;
    form_number : OPTIONAL bspline_surface_form;
END_ENTITY;

Entity Name: B.SPLINE_SURFACE
Entity Number: GEO-60
A b-spline surface.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

B.SPLINE_SURFACE.ID

Other Attributes

U.RATIONAL
Logical value which indicates whether the surface is rational (TRUE) or simple polynomial (FALSE) in first (u) parametric direction.

V.RATIONAL
Logical value which indicates whether the surface is rational (TRUE) or simple polynomial (FALSE) in second (v) parametric direction.

U.UNIFORM
Logical value which indicates whether the knot set is uniform (TRUE) or not uniform (FALSE) in “u”.

V.UNIFORM
Logical value which indicates whether the knot set is uniform (TRUE) or not uniform (FALSE) in “v”.

U.DEGREE
Integer value which is the algebraic degree of the basis functions in “u”.

V.DEGREE
Integer value which is the algebraic degree of the basis functions in “v”.

FORM NUMBER
Enumeration selection used to identify the particular type of surface.

Business Rules

EXPRESS Specification

ENTITY b_spline_surface
  SUBTYPE OF (bounded_surface);
  u.rational : LOGICAL;
  v.rational : LOGICAL;
  u.uniform : LOGICAL;
  v.uniform : LOGICAL;
  u.degree : INTEGER;
  v.degree : INTEGER;
  u.upper : INTEGER;
  v.upper : INTEGER;
  control_points : ARRAY [0..u.upper] OF
Section 2: NOMINAL SHAPE INFORMATION MODEL

ENTITY swept_surface
    SUPERTYPE OF (surface_of_revolution OR surface_of_linear_extrusion)
    SUBTYPE OF (surface);
END_ENTITY;

Entity Name: SWEPT_SURFACE

Entity Number: GEO-61

A surface that is constructed by sweeping a curve by another curve.

Primary Key Attributes
SWEPT_SURFACE.ID

Other Attributes
CURVE.ID (FK)
DIRECTION.ID (FK)
SWEPT_SURFACE.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY swept_surface
    SUPERTYPE OF (surface_of_revolution OR surface_of_linear_extrusion)
    SUBTYPE OF (surface);
END_ENTITY;

Entity Name: RECTANGULAR_TRIMMED_SURFACE

Entity Number: GEO-62

A simple bounded surface in which the boundaries are constant parametric lines.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
RECTANGULAR.TRIMMED.SURFACE.ID

Other Attributes
SURFACE.ID (FK)
Surface to be trimmed.

UMIN
Real number value of first “u” parametric value.

UMAX
Real number value of second “u” parametric value.

VMIN
Real number value of first “v” parametric value.

VMAX
Real number value of second “v” parametric value.

Business Rules

EXPRESS Specification

ENTITY rectangular_trimmed_surface
  SUBTYPE OF (bounded_surface);
  basis_surface : EXTERNAL surface;
  umin : REAL;
  vmin : REAL;
  umax : REAL;
  vmax : REAL;
END_ENTITY;

Entity Name: BOUNDED_SURFACE

Entity Number: GEO-63

A class of surfaces with finite area. The area may be finite naturally (as a sphere) or finite due to trimming.

Primary Key Attributes
BOUNDED SURFACE ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes
BOUND_SURFACE_TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY bounded_surface
   SUPERTYPE OF (b_spline_surface IOR
       bezier_surface IOR
       rectangular_trimmed_surface IOR
       curve_bounded_surface)
   SUBTYPE OF (surface);
END_ENTITY;

Entity Name: OFFSET_SURFACE

Entity Number: GEO-64

This is a procedural definition of a simple offset surface at a normal distance from the originating surface.

Primary Key Attributes
OFFSET_SURFACE.ID

Other Attributes
SURFACE.ID (FK)
The surface that is to be offset.
DISTANCE
Real value which is offset distance.
SELF.INTERSECTION
Three value flag to indicate whether curve self intersects:
   TRUE
       Curve does self intersect.
   FALSE
       Curve does not self intersect.
   UNDEFINED
       Whether curve self intersects or not is unknown.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY offset_surface
  SUBTYPE OF (surface);
  basis_surface : surface;
  distance : REAL;
  self_intersect : true_false_or_undefined;
END_ENTITY;

Entity Name: RECTANGULAR.COMPOSITE_SURFACE

Entity Number: GEO-65

This is a composite surface having a simple rectangular topology of patches. Each patch (component surface) must be a bound and topologically rectangular and therefore be either a RECTANGULAR.COMPOSITE_SURFACE PATCH/GEO-79 and/or B.SPLINE.SURFACE/GEO-60.

Each patch is, if necessary, reparameterized from 0 to 1 and the resulting composite surface has a simple cumulative parameterization.

It is required that there is at least positional continuity between adjacent surfaces.

Primary Key Attributes

RECTANGULAR.COMPOSITE_SURFACE_ID

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY rectangular_composite_surface
  SUBTYPE OF (surface);
  n_u : INTEGER;
  n_v : INTEGER;
  surfaces : ARRAY [1:n_u] OF
    ARRAY [1:n_v] OF surface;
  u_senses : ARRAY [1:n_u] OF
    ARRAY [1:n_v] OF LOGICAL:
  v_senses : ARRAY [1:n_u] OF
Section 2: NOMINAL SHAPE INFORMATION MODEL

ARRAY [1:n_v] OF LOGICAL:
END_ENTITY;

Entity Name: PLANE

Entity Number: GEO-66

A unbounded planar surface.

Primary Key Attributes
PLANE.ID

Other Attributes
None

Business Rules

EXPRESS Specification
ENTITY plane
    SUBTYPE OF (elementary_surface);
    position : axis2_placement;
END_ENTITY;

Entity Name: CYLINDRICAL_SURFACE

Entity Number: GEO-67

An unbounded cylindrical surface.

Primary Key Attributes
CYLINDRICAL_SURFACE.ID

Other Attributes
RADIUS
Real value which is the radius of the cylinder.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY cylindrical_surface
  SUBTYPE OF (elementary_surface);
  radius : REAL;
  position : axis2_placement;
END_ENTITY;

Entity Name: CONICAL_SURFACE

Entity Number: GEO-68
An unbounded surface of a cone.

Primary Key Attributes

CONICAL_SURFACE.ID

Other Attributes

SEMI_ANGLE
  Real value which is the cone semi-angle in degrees.

RADIUS
  Real value which is the radius of the circular curve of intersection between the cone and a plane perpendicular to the axis of the cone passing through the location point.

Business Rules

EXPRESS Specification

ENTITY conical_surface
  SUBTYPE OF (elementary_surface);
  semi_angle : REAL;
  radius : REAL;
  position : axis2_placement;
END_ENTITY;

Entity Name: SPHERICAL_SURFACE

Entity Number: GEO-69
A complete spherical surface.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

SPHERICAL_SURFACE.ID

Other Attributes

RADIUS

Business Rules

EXPRESS Specification

ENTITY spherical_surface
   SUBTYPE OF (elementary_surface);
   radius : REAL;
   position : axis2_placement;
END_ENTITY;

Entity Name: TOROIDAL_SURFACE

Entity Number: GEO-70

The complete surface of a torus.

Primary Key Attributes

TOROIDAL_SURFACE.ID

Other Attributes

MAJOR_RADIUS
   Real value which is the major radius of the torus.

MINOR_RADIUS
   Real value which is the minor radius of the torus.

Business Rules

EXPRESS Specification

ENTITY toroidal_surface
   SUBTYPE OF (elementary_surface);
   major_radius : REAL;
   minor_radius : REAL;
Section 2: NOMINAL SHAPE INFORMATION MODEL

ENTITY surface_of_linear_extrusion
  SUBTYPE OF (swept_surface);
  axis : direction;
  extruded_curve : curve;
END_ENTITY;

Entity Name: SURFACE_OF_LINEAR_EXTRUSION

Entity Number: GEO-71

A simple swept surface or a generalized cylinder obtained by sweeping a curve in a given direction.

Primary Key Attributes
SURFACE_OF_LINEAR_EXTRUSION_ID

Other Attributes
DIRECTION_ID (FK)
The direction of extrusion.

CURVE_ID (FK)
The curve to be swept.

Business Rules

EXPRESS Specification

ENTITY surface_of_linear_extrusion
  SUBTYPE OF (swept_surface);
  axis : direction;
  extruded_curve : curve;
END_ENTITY;

Entity Name: SURFACE_OF_REVOLUTION

Entity Number: GEO-72

A surface obtained by rotating a curve a complete revolution about an axis.

Primary Key Attributes
SURFACE_OF_REVOLUTION_ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

CARTESIAN POINT ID (FK)
A point on the axis of revolution.

DIRECTION ID (FK)
The direction of the axis of revolution.

CURVE ID (FK)
The curve that is revolved about the axis.

Business Rules

EXPRESS Specification

ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
  axis_point : point;
  axis : direction;
  revolved_curve : curve;
END_ENTITY;

Entity Name: B.SPLINE.SURFACE.KNOT

Entity Number: GEO-73

A knot of a B-spline surface.

Primary Key Attributes

B.SPLINE.SURFACE.ID (FK)

Other Attributes

U.OR.V
Enumeration selection which indicates whether this is for the first (u) parametric or second (v).

MULTIPLICITY
Integer value of multiplicity of this knot.

KNOT.VALUE
Real number value of this knot which is one of a set of knots used to define the B-spline basis function.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

In the EXPRESS for B.SPLINE_SURFACE/GEO-60 see the following attributes:

- u_multiplicities
- u_knots
- v_multiplicities
- v_knots

Entity Name: BS.SURFACE.KNOT.ORDER

Entity Number: GEO-74

IDEF1X defaults to unordered sets. This entity makes BS.SURFACE.KNOT/GEO-73 a matrix.

Primary Key Attributes

B.SPLINE_SURFACE.KNOT_ID
PREDECESSOR.B.SPLINE_SURFACE.KNOT_ID (FK)
SUCCESSOR.B.SPLINE_SURFACE.KNOT_ID (FK)

Other Attributes

None

Business Rules

EXPRESS Specification

In the EXPRESS for B.SPLINE_SURFACE/GEO-60 see the following attributes:

- u_multiplicities
- u_knots
- v_multiplicities
- v_knots

Entity Name: BEZIER_SURFACE.CONTROL.POINT

Entity Number: GEO-77

A control point for a BEZIER_SURFACE/GEO-59.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

BEZIER_SURFACE.ID (FK)

BEZIER_SURFACE.CONTROL.POINT.ID

Other Attributes

CARTESIAN.POINT (FK)

WEIGHT

Real number value of weight associated with this control point.

Business Rules

EXPRESS Specification

In the EXPRESS for BEZIER_SURFACE/GEO-59 see attribute control_points.

Entity Name: BZ_SURFACE.CONTROL.POINT.ORDER

Entity Number: GEO-78

IDF1X defaults to unordered sets. This entity makes BEZIER_SURFACE.CONTROL.POINT/GEO-77 a matrix.

Primary Key Attributes

BEZIER_SURFACE.ID (FK)

PREDECESSOR.BEZIER_SURFACE.CONTROL.POINT.ID (FK)

SUCCESSOR.BEZIER_SURFACE.CONTROL.POINT.ID (FK)

U.OR.V

Enumeration selection of the parametric direction of the predecessor-successor relationship.

Other Attributes

None

Business Rules

EXPRESS Specification

In the EXPRESS for BEZIER_SURFACE/GEO-59 see attribute control_points.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: RECTANGULAR.COMPOSITE_SURFACE_PATCH

Entity Number: GEO-79


Primary Key Attributes

RECTANGULAR.COMPOSITE_SURFACE_ID (FK)
RECTANGULAR.TRIMMED_SURFACE_ID (FK)

Other Attributes

U.SENSE
Logical value which indicates whether the sense of "u" parameterization in composite agrees with (TRUE) the original surface.

V.SENSE Logical value which indicates whether the sense of "v" parameterization in composite agrees with (TRUE) the original surface.

Business Rules

EXPRESS Specification

In the EXPRESS for RECTANGULAR.COMPOSITE_SURFACE/GEO-65, see the following attributes:

- surfaces
- u.senses
- v.senses

Entity Name: RC_SURFACE_PATCH_ORDER

Entity Number: GEO-80

IDEFIX defaults to unordered sets. This entity makes RECTANGULAR.COMPOSITE_SURFACE_PATCH/GEO-79 a matrix.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

RECTANGULAR.COMPOSITE_SURFACE.ID (FK)
PREDECESSOR.RECTANGULAR.TRIMMED_SURFACE.ID (FK)
SUCCESSOR.RECTANGULAR.TRIMMED_SURFACE.ID (FK)

U.OR.V

Enumeration selection which indicates whether the predecessor-successor relation is in the "u" (U) or "v" (V) direction.

Other Attributes

None

Business Rules

EXPRESS Specification

In the EXPRESS for RECTANGULAR.COMPOSITE_SURFACE/GEO-65, see the following attributes:

- surfaces
- u_senses
- v_senses

Entity Name: BOUNDED_SURFACE_BOUNDARY

Entity Number: GEO-81

The use of a COMPOSITE_CURVE_ON_SURFACE/GEO-50 as part or all of the boundary of a CURVE_BOUNDED_SURFACE/GEO-110.

Primary Key Attributes

CURVE_BOUNDED_SURFACE.ID (FK)
COMPOSITE_CURVE_ON_SURFACE.ID (FK)

Other Attributes

None
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

In EXPRESS for CURVE_BOUNDED_SURFACE/GEO-110, see attribute "boundaries".

Entity Name: CURVE_BOUNDED_SURFACE

Entity Number: GEO-110

A surface trimmed with COMPOSITE CURVE ON_SURFACE/GEO-50 entities. If the outer boundary is omitted, the natural boundaries of the trimmed surface are assumed.

Primary Key Attributes

CURVE_BOUNDED_SURFACE.ID

Other Attributes

SURFACE.ID (FK)
The surface to be bound.

OUTERPRESENT
Logical value to indicate whether the outer boundary is explicitly present (TRUE) or is defaulted to the natural boundary (FALSE).

Business Rules

EXPRESS Specification

ENTITY curve_bounded_surface
  SUBTYPE OF (bounded_surface);
  basis_surface : surface;
  boundaries : SET [1:*] OF composite_curve_on_surface;
  outer_present : LOGICAL;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-1: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-2: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-3: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-4: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-5: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-6: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-7: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-8: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-9: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-10: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-11: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-12: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-13: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-14: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-15: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-16: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-17: SHAPE Entity/Relationship Diagram

76
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-18: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-19: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-20: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-21: SHAPE Entity/Relationship Diagram
SECTION 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-22: SHAPE Entity/Relationship Diagram
SECTION 2: NOMINAL SHAPE INFORMATION MODEL

2.4 SOLIDS

Entity Name: SOLID.MODEL

Entity Number: GEO-82

A complete representation of product nominal shape; any point can be classified as being inside, outside or on the boundary of a solid.

Primary Key Attributes

SOLID.MODEL.ID

Other Attributes

SOLID.MODEL.TYPE (discriminator)

Business Rules

EXPRESS Specification

```
ENTITY solid_model
  SUPERTYPE OF (boolean_operation XOR
csg_primitive XOR
csg_solid XOR
facetted_brep XOR
half_space XOR
manifold_solid_brep XOR
solid_instance XOR
swept_area_solid)
  SUBTYPE OF (shape_model);
END_ENTITY;
```

Entity Name: CSG.PRIMITIVE

Entity Number: GEO-84

A collector for all CSG primitives. A primitive is a simple solid used in the construction of more complex solids using regularized operations.

Primary Key Attributes

CSG.PRIMITIVE.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-22: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

3.4 SOLIDS

Entity Name: SOLID.MODEL

Entity Number: GEO-82

A complete representation of product nominal shape; any point can be classified as being inside, outside or on the boundary of a solid.

Primary Key Attributes

SOLID.MODEL.ID

Other Attributes

SOLID.MODEL.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY solid_model
  SUPERTYPE OF (boolean_operation IOR
csg.primitive IOR
csg.solid IOR
faceted_brep IOR
half_space IOR
manifold.solid.brep IOR
solid.instance IOR
swept_area.solid)
  SUBTYPE OF (shape_model);
END_ENTITY;

Entity Name: CSG.PRIMITIVE

Entity Number: GEO-84

A collector for all CSG primitives. A primitive is a simple solid used in the construction of more complex solids using regularized operations.

Primary Key Attributes

CSG.PRIMITIVE.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-23: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

CSG_PRIMITIVE_TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY csgPrimitive
  SUPERTYPE OF (sphere IOR
    primitive_with_one_axis IOR
    primitive_with_axes)
  SUBTYPE OF (solid_model);
END_ENTITY;

Entity Name: CSG.SOLID
Entity Number: GEO-85

A solid made by combining simpler solids using regularized Boolean operations. The allowed operations are intersection, union, and difference. A regularized operation is defined as the closure of the interior of the result of the Boolean operation.

The CSG.SOLID entity serves two functions. First it identifies the root Boolean operation. Second it is a means to differentiate intermediate results and significant results. The result of the Boolean operation pointed to by a CSG.SOLID is significant and should be kept; otherwise, the result is only an intermediate result.

Primary Key Attributes

CSG SOLID ID

Other Attributes

BOOLEAN OPERATION ID (FK)

Boolean expression of primitives and regularized operators describing the solid.

Business Rules

EXPRESS Specification

ENTITY csgSolid
  SUBTYPE OF (solid_model);
  tree_expression : boolean_operation;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: FACETTED_BREP

Entity Number: GEO-86

The FACETTED BREP has been introduced in order to support the large number of systems that allow boundary type solid representations with planar surfaces only. Facetted models may be represented by MANIFOLD_SOLID_BREP/GEO-88 but their representation as a FACETTED_BREP/GEO-86 will be more compact.

Primary Key Attributes

FACETTED_BREP.ID

Other Attributes

SHELL_LOGICAL_STRUCTURE.ID (FK)
  A closed shell defining the exterior boundary of the solid.

Business Rules

EXPRESS Specification

ENTITY faceted_brep
  SUBTYPE OF (solid_model);
  outer : shell_logical_structure;
  voids : SET [0:*] OF shell_logical_structure;
END_ENTITY;

Entity Name: HALF_SPACE

Entity Number: GEO-87

The half space which is the regular subset of the domain which lies on one side of an unbounded surface. The domain is an orthogonal box or all space (all space is the default). See BOX_DOMAIN/GEO-109 for domain definition. Which side of the surface is determined by the surface normals and the complement flag. If the complement flag is FALSE, then the subset is the one the normals point away from. If the complement flag is TRUE, then the subset is the one the normals point into.

For a valid HALF_SPACE/GEO-87, the surface must divide the domain into exactly two subsets. Also, within the domain the surface must be manifold and all the surface normals must point into the same subset.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
HALF_SPACE_SOLID_ID

Other Attributes
SURFACE_ID (FK)
Surface defining side of half space.

COMPLEMENT_FLAG
Logical value which is the complement flag defined above.

Business Rules

EXPRESS Specification

ENTITY half_space
  SUBTYPE OF (solid_model);
  base_surface : surface;
  enclosure : OPTIONAL box_domain;
  complement_flag : LOGICAL;
END_ENTITY;

Entity Name: MANIFOLD_SOLID_B.REP
Entity Number: GEO-88

A manifold solid is an arcwise connected closed finite volume and the surface(s) of the solid is arcwise connected orientable compact two-manifold. There is no restriction on the genus of the volume, nor on the number of voids within the volume.

Primary Key Attributes
MANIFOLD_SOLID_B.REP.ID

Other Attributes
CLOSED_SHELL_LOGICAL_STRUCTURE_ID (FK)
A closed shell defining the exterior boundary of the solid.

Business Rules

EXPRESS Specification

ENTITY manifold_solid_brep
Section 2: NOMINAL SHAPE INFORMATION MODEL

ENTITY solid_instance
  SUBTYPE OF (solid_model);
  solid_to_be_copied : solid_model;
  location : axis2_placement;
END_ENTITY;

Entity Name: SWEPT_AREA_SOLID
Entity Number: GEO-90
A collector for solids defined by a sweeping action on planar faces.

Primary Key Attributes
SWEPT_AREA_SOLID.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

SWEPT_AREA.SOLID.TYPE (discriminator)

FACE_ID (FK)
  Face to be swept.

Business Rules

EXPRESS Specification

ENTITY swept_area_solid
  SUPERTYPE OF (solid_of_revolution IOR
    solid_of_linear_extrusion)
  SUBTYPE OF (solid_model);
END_ENTITY;

Entity Name: BOOLEAN.OPERATION

Entity Number: GEO-91

A regularized operation on two solids to create a new solid. Valid operations are regularized union, regularized intersection, regularized complement, and regularized difference. For purposes of Boolean operations, a solid is a regularized set of points.

Primary Key Attributes

BOOLEAN.OPERATION.ID

Other Attributes

BOOLEAN.OPERATION.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY boolean_operation
  SUPERTYPE OF (union IOR
    intersection IOR
    difference)
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: UNION

Entity Number: GEO-92

UNION on two solids is the regularization of the set of all points that are in the FIRST.OPERAND or the SECOND.OPERAND or both.

Primary Key Attributes
UNION.ID

Other Attributes
FIRST.OPERAND.SOLID.MODEL.ID (FK)
SECOND.OPERAND.SOLID.MODEL.ID (FK)

Business Rules

EXPRESS Specification

ENTITY union
    SUBTYPE OF (boolean_operation);
    first_operand : solid_model;
    second_operand : solid_model;
END_ENTITY;

Entity Name: INTERSECTION

Entity Number: GEO-93

INTERSECTION on two solids is the regularization of the set of all points that are in both the FIRST.OPERAND and the SECOND.OPERAND.

Primary Key Attributes
INTERSECTION.ID

Other Attributes
FIRST.OPERAND.SOLID.MODEL.ID (FK)
SECOND.OPERAND.SOLID.MODEL.ID (FK)
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY intersection
  SUBTYPE OF (boolean_operation);
  first_operand : solid_model;
  second_operand : solid_model;
END_ENTITY;

Entity Name: DIFFERENCE
Entity Number: GEO-94

DIFFERENCE on two solids is the regularization of the set of all points that are in the WORK PIECE, but not in the TO BE REMOVED.

Primary Key Attributes
  DIFFERENCE_ID

Other Attributes
  WORK PIECE.SOLID_MODEL.ID (FK)
  TO BE REMOVED.SOLID_MODEL.ID (FK)

Business Rules

EXPRESS Specification

ENTITY difference
  SUBTYPE OF (boolean_operation);
  work_piece : solid_model;
  to_be_removed : solid_model;
END_ENTITY;

Entity Name: BREP SHELL LOGICAL STRUCTURE
Entity Number: GEO-96

The use of a CLOSED SHELL/TOP-18 by a MANIFOLD SOLID B REP/GEO-88 as an interior void.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
MANIFOLD.SOLID.B.REP_ID (FK)
CLOSED SHELL_ID (FK)

Other Attributes
FLAG
Logical value which indicates whether the shell normal agrees with (TRUE) or is in the opposite direction (FALSE) to the manifold solid normal.

Business Rules

EXPRESS Specification
ENTITY closed_shell_logical_structure;
closed_shell_element : closed_shell;
flag : LOGICAL;
END_ENTITY;

Entity Name: PRIMITIVE.WITH.ONE_AXIS
Entity Number: GEO-97

A collector for axis symmetrical CSG primitives.

Primary Key Attributes
PRIMITIVE.WITH.ONE_AXIS.ID

Other Attributes
PRIMITIVE.WITH.ONE_AXIS.TYPE (discriminator)
AXIS1_PLACEMENT (FK)
The location of a point on the axis and the direction of the axis.

Business Rules

EXPRESS Specification
ENTITY primitive_with_one_axis
SUPERTYPE OF (right_circular_cone ID OR right_circular_cylinder ID)
Section 2: NOMINAL SHAPE INFORMATION MODEL

torus)
    SUBTYPE OF (csg.primitive);
END_ENTITY;

Entity Name: RIGHT.CIRCULAR.CYLINDER
Entity Number: GEO-98

A right circular cylinder CSG primitive.

Primary Key Attributes
RIGHT.CIRCULAR.CYLINDER.ID

Other Attributes
RADIUS
    Real number value which is the radius of the cylinder.

HEIGHT
    Real number value which is the height of the cylinder.

Business Rules

EXPRESS Specification

ENTITY right_circular_cylinder
    SUBTYPE OF (primitive_with_one_axis);
    radius : REAL;
    position : axis1_placement;
    height : REAL;
END_ENTITY;

Entity Name: RIGHT.CIRCULAR.CONE
Entity Number: GEO-99

A right circular cone CSG primitive.

Primary Key Attributes
RIGHT.CIRCULAR.CONE.ID
Other Attributes

SEMI-ANGLE
Real number value which is one half the angle of the cone in degrees.

RADIUS
Real number value which is the radius of the cone at the axis point. That is a non-zero radius means the cone is truncated.

HEIGHT
Real number value which is the distance the base of the cone and the point at which the radius is measured.

Business Rules

EXPRESS Specification

ENTITY right_circular_cone
    SUBTYPE OF (primitive_with_one_axis);
    semi_angle : REAL;
    radius : REAL;
    position : axis1_placement;
    height : REAL;
END_ENTITY;

Entity Name: TORUS

Entity Number: GEO-100

A torus CSG primitive.

Primary Key Attributes
TORUS.ID

Other Attributes

MAJOR_RADIUS
Real number which is the radius of the directrix.

MINOR_RADIUS
Real number which is the radius of the generatrix.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY torus
    SUBTYPE OF (primitive_with_one_axis);
    major_radius : REAL;
    minor_radius : REAL;
    position : axis1_placement;
END_ENTITY;

Entity Name: SPHERE

Entity Number: GEO-101

A sphere CSG primitive.

Primary Key Attributes
SPHERE_ID

Other Attributes
RADIUS
Real value which is the radius of the sphere.

Business Rules

EXPRESS Specification

ENTITY sphere
    SUBTYPE OF (csg_primitive);
    radius : REAL;
    center : point;
END_ENTITY;

Entity Name: PRIMITIVE.WITH_AXES

Entity Number: GEO-102

A collector for non-Axisymmetric CSG primitives.
Primary Key Attributes
PRIMITIVE.WITH_AXES.ID

Other Attributes
PRIMITIVE.WITH_AXES.TYPE (discriminator)
AXIS2_PLACEMENT.ID

Business Rules

EXPRESSION Specification

ENTITY primitive_with_axes
  SUPERTYPE OF (right-angular_wedge IOR block)
  SUBTYPE OF (csg_primitive);
END_ENTITY;

Entity Name: RIGHT ANGULAR WEDGE
Entity Number: GEO-104

A right angular wedge CSG primitive. The wedge may be truncated or come to an apex.

Primary Key Attributes
RIGHT ANGULAR WEDGE_ID

Other Attributes
X
Real number value which is the size of the wedge along the X axis.

Y
Real number value which is the size of the wedge along the Y axis.

Z
Real number value which is the size of the wedge along the Z axis.

LTX
Real number value which is the distance in the positive X direction of the smaller surface of the wedge.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY right_angular_wedge
  SUBTYPE OF (primitive_with_axes);
  x : REAL;
y : REAL;
z : REAL;
ltx : REAL;
  position : axis2_Placement;
END_ENTITY;

Entity Name: BLOCK

Entity Number: GEO-105

A rectangular block CSG primitive.

Primary Key Attributes

BLOCK.ID

Other Attributes

X
Real number value which is the size of the block along the X axis.

Y
Real number value which is the size of the block along the Y axis.

Z
Real number value which is the size of the block along the Z axis.

Business Rules

EXPRESS Specification

ENTITY block
  SUBTYPE OF (primitive_with_axes);
  x : REAL;
y : REAL;
z : REAL;
  position : axis2_Placement;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

END_ENTITY;

Entity Name: SOLID_OF_LINEAR_EXTRUSION

Entity Number: GEO-106

A SOLID_OF_LINEAR_EXTRUSION is a solid defined by sweeping a planar face. The direction translation is defined by a direction vector. The planar face may have holes which will sweep into holes in the solid.

Primary Key Attributes

SOLID_OF_LINEAR_EXTRUSION.ID

Other Attributes

DIRECTION.ID (FK)

The direction in which the face is to be swept.

DEPTH

Real number which is the distance the face is to be swept.

Business Rules

EXPRESS Specification

ENTITY solid_of_linear_extrusion
   SUBTYPE OF (swept_area_solid);
   extruded_face : face;
   extruded_direction : direction;
   depth : REAL;
END_ENTITY;

Entity Name: SOLID_OF_REVOlUTION

Entity Number: GEO-107

A SOLID OF REVOLUTION is a solid formed by revolving a planar face about an axis. The axis must be in the plane of the face and the axis must not intersect the interior of the face. The planar face may have holes which will sweep into holes in the solid.

Primary Key Attributes

SOLID_OF_REVOlUTION.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

AXIS1 PLACEMENT ID (FK)
Axis about which swept will be made.

ANGLE
Real number which is angle through which the sweep will be made. The angle is in degrees.

Business Rules

EXPRESS Specification

ENTITY solid_of_revolution
  SUBTYPE OF (swept_area_solid);
    axis : axis1_placement;
    extruded_face : face;
    angle : REAL;
END_ENTITY;

Entity Name: BOX_DOMAIN

Entity Number: GEO-109

A BOX DOMAIN is an orthogonal box which may be used to limit the domain of a HALF_SPACE/GEO-87.

Primary Key Attributes
HALF_SPACE_SOLID_ID (FK)

Other Attributes
X.MIN
Real value which is X coordinate of most negative corner of box.

Y.MIN
Real value which is Y coordinate of most negative corner of box.

Z.MIN
Real value which is Z coordinate of most negative corner of box.

X.MAX
Real value which is X coordinate of most positive corner of box.

Y.MAX
Real value which is X coordinate of most positive corner of box.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Z.MAX
Real value which is X coordinate of most positive corner of box.

Business Rules

EXPRESS Specification

ENTITY box_domain
  x_min : REAL;
  y_min : REAL;
  z_min : REAL;
  x_max : REAL;
  y_max : REAL;
  z_max : REAL;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-24: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-25: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

SECTION 3: NOMINAL SHAPE INFORMATION MODEL

Figure D-26: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-27: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-28: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-29: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-30: SHAPE Entity/Relationship Diagram
SECTION 2. NOMINAL SHAPE INFORMATION MODEL

Figure D-31: SHAPE Entity/Relationship Diagram
2.5 TOPOLOGY

Entity Name: TOPOLOGY

Entity Number: TOP-1

A collector for all topological entities

Primary Key Attributes

TOPOLOGY_ID

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY topology
   SUPERTYPE OF (vertex XOR edge XOR path XOR loop XOR face XOR subface XOR shell XOR region);
   END_ENTITY;

Entity Name: VERTEX

Entity Number: TOP-2

A topological vertex.

Primary Key Attributes

VERTEX_ID

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY vertex
   SUBTYPE OF (topology);
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-32: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

vertex_point : OPTIONAL point;
END_ENTITY;

Entity Name: EDGE

Entity Number: TOP-3

A topological edge.

Primary Key Attributes

EDGES.ID

Other Attributes

EDGES.START.VERTEX.ID (FK)
Start point (vertex) of the edge.

EDGES.END (FK)
End point (vertex) of the edge.

Business Rules

EXPRESS Specification

ENTITY edge
  SUBTYPE OF (topology);
  edge_start : vertex;
  edge_end : vertex;
  edge_curve : OPTIONAL curve_logical_structure;
END_ENTITY;

Entity Name: LOOP

Entity Number: TOP-4

A collector for all types of topological loops.

Primary Key Attributes

LOOP.ID

Other Attributes

LOOP.TYPE (discriminator)
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY loop
   SUPERTYPE OF (vertex_loop IOR edge_loop IOR poly_loop)
   SUBTYPE OF (topology);
END_ENTITY;

Entity Name: FACE
Entity Number: TOP-5
A topological face.

Primary Key Attributes
FACE.ID

Other Attributes
None

Business Rules

EXPRESS Specification

ENTITY face
   SUBTYPE OF (topology);
   outer_bound : OPTIONAL loop_logical_structure;
   bounds : set [1 : 0] OF loop_logical_structure;
   face_surface : OPTIONAL surface_logical_structure;
END_ENTITY;

Entity Name: SHELL
Entity Number: TOP-6
A collector for all types of topological shells.

Primary Key Attributes
SHELL.ID
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes
SHELL.TYPE (discriminator)

Business Rules

EXPRESS Specification

ENTITY shell
    SUPERTYPE OF (vertex_shell IOR wire_shell IOR open_shell IOR closed_shell)
    SUBTYPE OF (topology);
END_ENTITY;

Entity Name: PATH
Entity Number: TOP-7

A collection of edges joined end to end.

Primary Key Attributes
PATH.ID

Other Attributes
None

Business Rules

EXPRESS Specification

ENTITY path
    SUBTYPE OF (topology);
    open_edge_list : LIST [1 : @] OF UNIQUE edge_logical_structure;
END_ENTITY;

Entity Name: SUBFACE
Entity Number: TOP-8

A SUBFACE is a portion of the domain of a FACE/TOP-5 or another SUBFACE/TOP-8.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes

SUBFACE ID

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY subface
    SUBTYPE OF (topology);
    outer_bound : loop_logical_structure;
    bounds : SET [1 : #] OF loop_logical_structure;
    trimming : select_face_or_subface;
END_ENTITY;

Entity Name: REGION

Entity Number: TOP-9

A collection of SHELL/TOP-6.

Primary Key Attributes

REGION.ID

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY region
    SUBTYPE OF (topology);
    outer_region_boundary : OPTIONAL shell_logical_structure;
    region_boundaries : SET [1 : #] OF shell_logical_structure;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: VERTEX_LOOP

Entity Number: TOP-10

A loop composed of a single vertex.

Primary Key Attributes

VERTEX_LOOP.ID

Other Attributes

VERTEX_ID (FK)

The unique vertex which composes the entire loop.

Business Rules

EXPRESSION Specification

ENTITY vertex_loop
  SUBTYPE OF (loop);
  loop_vertex : UNIQUE vertex;
END_ENTITY;

Entity Name: POLY_LOOP

Entity Number: TOP-11

A ordered co-planar collection of points forming the vertices of a loop. The loop is composed of straight line segments joining point in the collection to the succeeding point in the collection. The closing segment is from the last to the first point in the collection. The direction of the loop is the direction of the line segments.

Primary Key Attributes

POLY_LOOP.ID

Other Attributes

None
Section 2: Nominal Shape Information Model

**Business Rules**

**EXPRESS Specification**

ENTITY poly_loop
   SUBTYPE OF (loop);
   polygon : LIST [3 : #] OF UNIQUE point;
END_ENTITY;

Entity Name: EDGE_LOOP

Entity Number: TOP-12

A topological loop represented by an ordered collection of (edge + logical) pairs such that adjacent edges in the loop are adjacent in the collection. The vertices of the loop may be determined by examining the edge data. The logical indicator is used to identify whether the positive edge direction agrees with (TRUE) or is opposed to (FALSE) the positive direction of the loop.

**Primary Key Attributes**

EDGE_LOOP.ID

**Other Attributes**

None

**Business Rules**

**EXPRESS Specification**

ENTITY edge_loop
   SUBTYPE OF (loop);
   loop_edges : LIST [1 : #] OF UNIQUE edge_logical_structure;
END_ENTITY;

Entity Name: LOCATED_VERTEX

Entity Number: TOP-13

The use of a POINT/GEO-2 to locate a VERTEX/TOP-2 in a coordinate space.

**Primary Key Attributes**

VERTEX.ID (FK)
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

POINT.ID (FK)
Location of the vertex

Business Rules

EXPRESS Specification
See located.vertex attribute of EXPRESS for VERTEX/TOP-2.

Entity Name: POLY.POINT

Entity Number: TOP-14

The use of a POINT/GEO-2 in a POLY-LOOP/TOP-11.

Primary Key Attributes

POLY.LOOP.ID (FK)

POINT.ID (FK)

Other Attributes

None

Business Rules

EXPRESS Specification
See polygon attribute of EXPRESS for POLY-LOOP/TOP-11.

Entity Name: CURVE.LOGICAL.STRUCTURE

Entity Number: TOP-15

The use of a CURVE/GEO-6 as the underlying geometry of a EDGE/TOP-3.

Primary Key Attributes

EDGE.ID (FK)
Section 2: NOMINAL SHAPE INFORMATION MODEL

Other Attributes

CURVE ID (FK)
   The underlying geometry of the edge.

FLAG
   Logical value which indicates whether the curve direction agrees with (TRUE) or is in the opposite direction (FALSE) to the edge direction.

Business Rules

EXPRESS Specification

ENTITY curve_logical_structure;
   curve_element : curve;
   flag : LOGICAL;
END_ENTITY;

Entity Name: LOOP_EDGE_LOGICAL_STRUCTURE

Entity Number: TOP-16

The use of an EDGE/TOP-3 in a EDGE_LOOP/TOP-12.

Primary Key Attributes

EDGE_LOOP_ID (FK)

FLAG
   Logical value which indicates whether the edge direction agrees with (TRUE) or is in the opposite direction (FALSE) to the loop direction.

Other Attributes

None

Business Rules

EXPRESS Specification

ENTITY loop_edge_logical_structure;
   edge_element : edge;
   flag : LOGICAL;

117
Section 2: NOMINAL SHAPE INFORMATION MODEL

END_ENTITY;

Entity Name: PATH_EDGE_LOGICAL_STRUCTURE
Entity Number: TOP-17

The use of an EDGE/TOP-3 in a PATH/TOP-7.

Primary Key Attributes
PATH_ID (FK)
EDGE_ID (FK)

Other Attributes
FLAG
Logical value which indicates whether the edge direction agrees with (TRUE) or is in the opposite direction (FALSE) to the path direction.

Business Rules

EXPRESSION Specification
Use edge logical structure defined in EXPRESSION for LOOP_EDGE_LOGICAL_STRUCTURE/TOP-16.

Entity Name: CLOSED_SHELL
Entity Number: TOP-18

A collection of faces joined side to side to create a boundary which divides space into two regions, one finite and the other infinite. The topological normal of the shell is directed from the finite to the infinite region.

Primary Key Attributes
CLOSED_SHELL_ID (FK)

Other Attributes
None
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY closed_shell
  SUBTYPE OF (shell):
  cshell_boundary : SET [1 : #] OF face_logical_structure;
END_ENTITY;

Entity Name: OPEN.SHELL
Entity Number: TOP-19

A collection of faces joined side to side to create an arcwise connected manifold oriented finite topological surfaces that is mappable to a plane.

Primary Key Attributes
OPEN.SHELL_ID

Other Attributes
None

Business Rules

EXPRESS Specification

ENTITY open_shell
  SUBTYPE OF (shell):
  shell_boundary : SET [1 : #] OF face_logical_structure;
END_ENTITY;

Entity Name: VERTEX.SHELL
Entity Number: TOP-20

A shell consisting of a single VERTEX_LOOP/TOP-10.

Primary Key Attributes
VERTEX SHELL_ID
Other Attributes
VERTEX_LOOP_ID (FK)

Business Rules

EXPRESS Specification
ENTITY vertex_shell
    SUBTYPE OF (shell);
    vertex_shell_boundary : UNIQUE vertex_loop;
END_ENTITY;

Entity Name: WIRE SHELL

Entity Number: TOP-21

A shell of dimensionality 1. It is represented by a connect graph, specifically by the set of loops forming the graph.

Primary Key Attributes
WIRE SHELL_ID

Other Attributes
None

Business Rules

EXPRESS Specification
ENTITY wire_shell
    SUBTYPE OF (shell);
    wire_shell_boundary : SET [1 : #] OF edge_loop;
END_ENTITY;

Entity Name: CLOSED SHELL FACE LOGICAL STRUCTURE

Entity Number: TOP-22

The use of a FACE/TOP-5 as a component of a CLOSED SHELL/TOP-18.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
CLOSED.SHELL.ID (FK)
FACE.ID (FK)

Other Attributes
FLAG
Logical value which indicates whether the face direction agrees with (TRUE) or is in the opposite direction (FALSE) to the shell direction.

Business Rules

EXPRESS Specification

ENTITY face.logical_structure;
  face_element : face;
  flag : LOGICAL;
END_ENTITY;

Entity Name: OPEN.SHELL.FACE.LOGICAL.STRUCTURE

Business Rules

EXPRESS Specification

See entity face.logical.structure in EXPRESS for CLOSED.SHELL.FACE.LOGICAL.STRUCTURE/TOP-22.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Section 3: NOMINAL SHAPE INFORMATION MODEL

Entity Name: SURFACE_LOGICAL_STRUCTURE

Entity Number: TOP-24

The use of a SURFACE/GEO-7 as the geometry of a FACE/TOP-5.

Primary Key Attributes

FACE_ID (FK)

Other Attributes

SURFACE_ID (FK) Geometric surface underlying face.

FLAG
Logical value which indicates whether the surface normal agrees with (TRUE) or is in the opposite direction (FALSE) to the face normal.

Business Rules

EXPRESS Specification

ENTITY surface_logical_structure;
  surface_element : surface;
  flag             : LOGICAL;
END_ENTITY;

Entity Name: LOOP_LOGICAL_STRUCTURE

Entity Number: TOP-25

The use of a LOOP/TOP-4 as a portion of or all of the boundary of a FACE/TOP-5.

Primary Key Attributes

FACE_ID (FK)

LOOP_ID (FK)

Other Attributes

FLAG
Logical value which indicates whether the loop direction agrees with (TRUE) or is in the opposite direction (FALSE) to the face direction.
OUTER_BOUND
Three value flag to indicate whether this loop is an outer bound.

TRUE
Loop is outer bound.
FALSE
Loop is not outer bound.
UNDEFINED
Whether loop is outer bound or not is unknown.

Business Rules

EXPRESS Specification

ENTITY loop_logical_structure;
  loop_element : loop;
  flag : LOGICAL;
END_ENTITY;

For EXPRESS of OUTER_BOUND attribute see "outer.bound" attribute in EXPRESS for FACE/TOP-5.

Entity Name: WIRE_SHELL_LOOP_LOGICAL_STRUCTURE

Entity Number: TOP-26

The use of a LOOP/TOP-4 as a component in a WIRE_SHELL/TOP-21.

Primary Key Attributes

WIRE_SHELL_ID (FK)
EDGE_LOOP_ID (FK)

Other Attributes

FLAG
Logical value which indicates whether the loop direction agrees with (TRUE) or is in the opposite direction (FALSE) to the used direction.

Business Rules

EXPRESS Specification

See ENTITY loop_logical_structure in EXPRESS for LOOP_LOGICAL_STRUCTURE/TOP-25.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Entity Name: SUBFACE.ON.FACE
Entity Number: TOP-27
A SUBFACE/TOP-8 trimming a FACE/TOP-5.

Primary Key Attributes
SUBFACE.ID (FK)

Other Attributes
FACE.ID (FK)

Business Rules

EXPRESS Specification
See EXPRESS entity subface in EXPRESS for SUBFACE/TOP-8.

Entity Name: RECURSIVE.SUBFACE
Entity Number: TOP-28
A SUBFACE/TOP-8 trimming another SUBFACE/TOP-8.

Primary Key Attributes
SUBFACE.ID (FK)

Other Attributes
TRIMMING.SUBFACE.ID (FK)

Business Rules

EXPRESS Specification
See EXPRESS entity subface in EXPRESS for SUBFACE/TOP-8.

Entity Name: SUBFACE.LOOP.LOGICAL.STRUCTURE
Entity Number: TOP-29
The use of a LOOP/TOP-4 as a portion of or all of the boundary of a SUBFACE/TOP-8.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Primary Key Attributes
SUBFACE.ID (FK)
LOOP.ID (FK)

Other Attributes
FLAG Logical value which indicates whether the loop direction agrees with (TRUE) or is in the opposite direction (FALSE) to the subface direction.

OUTER_BOUND Three value flag to indicate whether this loop is an outer bound. TRUE - loop is outer bound. FALSE - loop is not outer bound. UNDEFINED - Whether loop is outer bound or not is unknown.

Business Rules

EXPRESS Specification
See ENTITY loop.logical.structure in EXPRESS for LOOP_LOGICAL_STRUCTURE/TOP-25.

Entity Name: REGION_SHELL_LOGICAL_STRUCTURE

Entity Number: TOP-30
The use of a SHELL/TOP-6 as a component of a REGION/TOP-9.

Primary Key Attributes
REGION.ID (FK)
SHELL.ID (FK)
FLAG
Logical value which indicates whether the shell direction agrees with (TRUE) or is in the opposite direction (FALSE) to the region's use of the shell.

Other Attributes
OUTER_REGION_BOUNDARY
Three value flag to indicate whether this shell is an outer bound.

TRUE
Shell is outer bound.
FALSE
Shell is not outer bound.
UNDEFINED
Whether shell is outer bound or not is unknown.
Section 2: NOMINAL SHAPE INFORMATION MODEL

Business Rules

EXPRESS Specification

ENTITY shell_logical_structure;
  shell_element : shell;
  flag : LOGICAL;
END_ENTITY;
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-33: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-34: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-35: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-36: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

Figure D-37: SHAPE Entity/Relationship Diagram
Section 2: NOMINAL SHAPE INFORMATION MODEL

<table>
<thead>
<tr>
<th>Number</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEO-1</td>
<td>GEOMETRY</td>
</tr>
<tr>
<td>GEO-2</td>
<td>POINT</td>
</tr>
<tr>
<td>GEO-3</td>
<td>VECTOR</td>
</tr>
<tr>
<td>GEO-4</td>
<td>AXIS PLACEMENT</td>
</tr>
<tr>
<td>GEO-5</td>
<td>TRANSFORMATION MATRIX</td>
</tr>
<tr>
<td>GEO-6</td>
<td>CURVE</td>
</tr>
<tr>
<td>GEO-7</td>
<td>SURFACE</td>
</tr>
<tr>
<td>GEO-8</td>
<td>CARTESIAN POINT</td>
</tr>
<tr>
<td>GEO-10</td>
<td>CARTESIAN TWO POINT</td>
</tr>
<tr>
<td>GEO-11</td>
<td>CARTESIAN THREE POINT</td>
</tr>
<tr>
<td>GEO-12</td>
<td>POINT ON CURVE</td>
</tr>
<tr>
<td>GEO-13</td>
<td>POINT ON SURFACE</td>
</tr>
<tr>
<td>GEO-14</td>
<td>DIRECTION</td>
</tr>
<tr>
<td>GEO-15</td>
<td>TWO SPACE DIRECTION</td>
</tr>
<tr>
<td>GEO-16</td>
<td>THREE SPACE DIRECTION</td>
</tr>
<tr>
<td>GEO-17</td>
<td>VECTOR WITH MAGNITUDE</td>
</tr>
<tr>
<td>GEO-18</td>
<td>AXIS PLACEMENT</td>
</tr>
<tr>
<td>GEO-19</td>
<td>AXIS2 PLACEMENT</td>
</tr>
<tr>
<td>GEO-20</td>
<td>LINE</td>
</tr>
<tr>
<td>GEO-21</td>
<td>CONIC</td>
</tr>
<tr>
<td>GEO-22</td>
<td>BOUNDED CURVE</td>
</tr>
<tr>
<td>GEO-23</td>
<td>CURVE ON SURFACE</td>
</tr>
<tr>
<td>GEO-24</td>
<td>OFF SET</td>
</tr>
<tr>
<td>GEO-25</td>
<td>CIRCLE</td>
</tr>
<tr>
<td>GEO-26</td>
<td>ELLIPSE</td>
</tr>
<tr>
<td>GEO-27</td>
<td>HYPERBOLA</td>
</tr>
<tr>
<td>GEO-28</td>
<td>PARABOLA</td>
</tr>
<tr>
<td>GEO-29</td>
<td>LINE SEGMENT</td>
</tr>
<tr>
<td>GEO-30</td>
<td>BEZIER CURVE</td>
</tr>
<tr>
<td>GEO-31</td>
<td>B SPLINE CURVE</td>
</tr>
<tr>
<td>GEO-32</td>
<td>TRIMMED CURVE</td>
</tr>
<tr>
<td>GEO-33</td>
<td>COMPOSITE CURVE</td>
</tr>
<tr>
<td>GEO-34</td>
<td>CONTROL POINT BEZIER</td>
</tr>
<tr>
<td>GEO-35</td>
<td>CONTROL POINT ORDER BEZIER</td>
</tr>
<tr>
<td>GEO-36</td>
<td>KNOT</td>
</tr>
<tr>
<td>GEO-37</td>
<td>CONTROL POINT</td>
</tr>
<tr>
<td>GEO-38</td>
<td>CONTROL POINT ORDER</td>
</tr>
<tr>
<td>GEO-39</td>
<td>POINT POINT TRIMMED CURVE</td>
</tr>
<tr>
<td>GEO-40</td>
<td>POINT PARAM TRIMMED CURVE</td>
</tr>
<tr>
<td>GEO-41</td>
<td>PARAM POINT TRIMMED CURVE</td>
</tr>
</tbody>
</table>

Table 1: Entity Pool - Numeric Order
Section 2: NOMINAL SHAPE INFORMATION MODEL

<table>
<thead>
<tr>
<th>Number</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEO-42</td>
<td>PARAM PARAM.TRIMMED.CURVE</td>
</tr>
<tr>
<td>GEO-43</td>
<td>KNOT.ORDER</td>
</tr>
<tr>
<td>GEO-44</td>
<td>SEGMENT</td>
</tr>
<tr>
<td>GEO-45</td>
<td>SEGMENT.ORDER</td>
</tr>
<tr>
<td>GEO-46</td>
<td>PCURVE</td>
</tr>
<tr>
<td>GEO-47</td>
<td>SURFACE.CURVE</td>
</tr>
<tr>
<td>GEO-48</td>
<td>SURFACE.CURVE.PCURVE</td>
</tr>
<tr>
<td>GEO-49</td>
<td>INTERSECTION CURVE</td>
</tr>
<tr>
<td>GEO-50</td>
<td>COMPOSITE.CURVE.ON_SURFACE</td>
</tr>
<tr>
<td>GEO-51</td>
<td>PCURVE.PCURVE.INTERSECTION</td>
</tr>
<tr>
<td>GEO-52</td>
<td>PCURVE SURFACE.INTERSECTION</td>
</tr>
<tr>
<td>GEO-53</td>
<td>SURFACE.SURFACE.INTERSECTION</td>
</tr>
<tr>
<td>GEO-54</td>
<td>ON_SURFACE_SEGMENT</td>
</tr>
<tr>
<td>GEO-55</td>
<td>ON_SURFACE_SEGMENT.ORDER</td>
</tr>
<tr>
<td>GEO-56</td>
<td>TWO.D OFFSET.CURVE</td>
</tr>
<tr>
<td>GEO-57</td>
<td>THREE.D OFFSET.CURVE</td>
</tr>
<tr>
<td>GEO-58</td>
<td>ELEMENTARY_SURFACE</td>
</tr>
<tr>
<td>GEO-59</td>
<td>BEZIER_SURFACE</td>
</tr>
<tr>
<td>GEO-60</td>
<td>B.SPLINE_SURFACE</td>
</tr>
<tr>
<td>GEO-61</td>
<td>SWEPT_SURFACE</td>
</tr>
<tr>
<td>GEO-62</td>
<td>RECTANGULAR.TRIMMED_SURFACE</td>
</tr>
<tr>
<td>GEO-63</td>
<td>BOUNDED_SURFACE</td>
</tr>
<tr>
<td>GEO-64</td>
<td>OFFSET_SURFACE</td>
</tr>
<tr>
<td>GEO-65</td>
<td>RECTANGULAR.COMPOSITE_SURFACE</td>
</tr>
<tr>
<td>GEO-66</td>
<td>PLANE</td>
</tr>
<tr>
<td>GEO-67</td>
<td>CYLINDRICAL_SURFACE</td>
</tr>
<tr>
<td>GEO-68</td>
<td>CONICAL_SURFACE</td>
</tr>
<tr>
<td>GEO-69</td>
<td>SPHERICAL_SURFACE</td>
</tr>
<tr>
<td>GEO-70</td>
<td>TOROIDAL_SURFACE</td>
</tr>
<tr>
<td>GEO-71</td>
<td>SURFACE.OF_LINEAR.EXTRUSION</td>
</tr>
<tr>
<td>GEO-72</td>
<td>SURFACE.OF_REVOLUTION</td>
</tr>
<tr>
<td>GEO-73</td>
<td>B.SPLINE_SURFACE.KNOT</td>
</tr>
<tr>
<td>GEO-74</td>
<td>BS_SURFACE.KNOT.ORDER</td>
</tr>
<tr>
<td>GEO-75</td>
<td>BEZIER.SURFACE.CONTROL_POINT</td>
</tr>
<tr>
<td>GEO-76</td>
<td>BZ_SURFACE.CONTROL_POINT.ORDER</td>
</tr>
<tr>
<td>GEO-77</td>
<td>RECTANGULAR.COMPOSITE_SURFACE_PATCH</td>
</tr>
<tr>
<td>GEO-78</td>
<td>RC_SURFACE.PATCH.ORDER</td>
</tr>
<tr>
<td>GEO-79</td>
<td>BOUNDED_SURFACE_BOUNDARY</td>
</tr>
<tr>
<td>GEO-80</td>
<td>SOLID_MODEL</td>
</tr>
<tr>
<td>GEO-81</td>
<td>CSG.PRIMITIVE</td>
</tr>
</tbody>
</table>

Table 1: Entity Pool – Numeric Order
### Section 2: NOMINAL SHAPE INFORMATION MODEL

<table>
<thead>
<tr>
<th>Number</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEO-85</td>
<td>CSG.SOLID</td>
</tr>
<tr>
<td>GEO-86</td>
<td>FACETTED.B.REP</td>
</tr>
<tr>
<td>GEO-87</td>
<td>HALF.SPACE</td>
</tr>
<tr>
<td>GEO-88</td>
<td>MANIFOLD.SOLID.B.REP</td>
</tr>
<tr>
<td>GEO-89</td>
<td>SOLID.INSTANCE</td>
</tr>
<tr>
<td>GEO-90</td>
<td>SWEPT.AREA.SOLID</td>
</tr>
<tr>
<td>GEO-91</td>
<td>BOOLEAN.OPERATION</td>
</tr>
<tr>
<td>GEO-92</td>
<td>UNION</td>
</tr>
<tr>
<td>GEO-93</td>
<td>INTERSECTION</td>
</tr>
<tr>
<td>GEO-94</td>
<td>DIFFERENCE</td>
</tr>
<tr>
<td>GEO-96</td>
<td>BREP.SHELL.LOGICAL.STRUCTURE</td>
</tr>
<tr>
<td>GEO-97</td>
<td>PRIMITIVE.WITH.ONE.AXIS</td>
</tr>
<tr>
<td>GEO-98</td>
<td>RIGHT.CIRCULAR.CYLINDER</td>
</tr>
<tr>
<td>GEO-99</td>
<td>RIGHT.CIRCULAR.CONE</td>
</tr>
<tr>
<td>GEO-100</td>
<td>TORUS</td>
</tr>
<tr>
<td>GEO-101</td>
<td>SPHERE</td>
</tr>
<tr>
<td>GEO-102</td>
<td>PRIMITIVE.WITH.AXES</td>
</tr>
<tr>
<td>GEO-104</td>
<td>RIGHT.ANGULAR.WEDGE</td>
</tr>
<tr>
<td>GEO-105</td>
<td>BLOCK</td>
</tr>
<tr>
<td>GEO-106</td>
<td>SOLID.OF.LINEAR.EXTURSION</td>
</tr>
<tr>
<td>GEO-107</td>
<td>SOLID.OF.REVOLUTION</td>
</tr>
<tr>
<td>GEO-108</td>
<td>BOX.DOMAIN</td>
</tr>
<tr>
<td>TOP-1</td>
<td>TOPOLOGY</td>
</tr>
<tr>
<td>TOP-2</td>
<td>VERTEX</td>
</tr>
<tr>
<td>TOP-3</td>
<td>EDGE</td>
</tr>
<tr>
<td>TOP-4</td>
<td>LOOP</td>
</tr>
<tr>
<td>TOP-5</td>
<td>FACE</td>
</tr>
<tr>
<td>TOP-6</td>
<td>SHELL</td>
</tr>
<tr>
<td>TOP-7</td>
<td>PATH</td>
</tr>
<tr>
<td>TOP-8</td>
<td>SUBFACE</td>
</tr>
<tr>
<td>TOP-9</td>
<td>REGION</td>
</tr>
<tr>
<td>TOP-10</td>
<td>VERTEX.LOOP</td>
</tr>
<tr>
<td>TOP-11</td>
<td>POLY.LOOP</td>
</tr>
<tr>
<td>TOP-12</td>
<td>EDGE.LOOP</td>
</tr>
<tr>
<td>TOP-13</td>
<td>LOCATED.VERTEX</td>
</tr>
<tr>
<td>TOP-14</td>
<td>POLY.POINT</td>
</tr>
<tr>
<td>TOP-15</td>
<td>CURVE.LOGICAL.STRUCTURE</td>
</tr>
<tr>
<td>TOP-16</td>
<td>LOOP.EDGE.LOGICAL.STRUCTURE</td>
</tr>
<tr>
<td>TOP-17</td>
<td>PATH.EDGE.LOGICAL.STRUCTURE</td>
</tr>
<tr>
<td>TOP-18</td>
<td>CLOSED.SHELL</td>
</tr>
</tbody>
</table>

Table 1: Entity Pool – Numeric Order
Section 2: NOMINAL SHAPE INFORMATION MODEL

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP-19</td>
<td>OPEN_SHELL</td>
</tr>
<tr>
<td>TOP-20</td>
<td>VERTEX_SHELL</td>
</tr>
<tr>
<td>TOP-21</td>
<td>WIRE_SHELL</td>
</tr>
<tr>
<td>TOP-22</td>
<td>CLOSED_SHELL_FACE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>TOP-23</td>
<td>OPEN_SHELL_FACE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>TOP-24</td>
<td>SURFACE_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>TOP-25</td>
<td>LOOP_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>TOP-26</td>
<td>WIRE_SHELL_LOOP_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>TOP-27</td>
<td>SUBFACE_ON_FACE</td>
</tr>
<tr>
<td>TOP-28</td>
<td>RECURSIVE_SUBFACE</td>
</tr>
<tr>
<td>TOP-29</td>
<td>SUBFACE_LOOP_LOGICAL_STRUCTURE</td>
</tr>
<tr>
<td>TOP-30</td>
<td>REGION_SHELL_LOOP_LOGICAL_STRUCTURE</td>
</tr>
</tbody>
</table>
Section 2: NOMINAL SHAPE INFORMATION MODEL
SECTION 3: SHAPE VARIATION TOLERANCES

3 SHAPE VARIATION TOLERANCES

3.1 Abstract

Dimensions on an engineering drawing define the theoretically exact shape of a three-dimensional object. Tolerances on those dimensions define the allowable deviation of the dimension from the nominal value as measured on the manufactured object. ANSI Y14.5M 1982 and ISO 1660,1101 specify the standard representation for dimensions and tolerances on engineering drawings. The reference model captures the semantic content of the dimensions and tolerances as specified in those standards.

3.2 Purpose & Scope

3.2.1 Purpose

The most important task in a manufacturing enterprise is providing sufficient information to production to allow for planning, manufacturing, and inspection of a product. This information is referred to as product definition data. Part of the product definition data is the representation of the desired shape of the product. Another, as equally important, part of this data is information concerning how accurately the manufactured object must correlate to the shape defined within the product definition. These limits on acceptable deviation are referred to as tolerances. To the design engineer, tolerances provide a tool for controlling critical aspects of the products shape. The process planner must make manufacturing decisions based on tolerances because of the varying precision of machine tools. Most importantly, tolerances give Quality Assurance criteria with which to accept or reject a product.

Product definition data is currently conveyed on engineering drawings. The product’s shape is represented pictorially with graphics and annotated dimensions which define the theoretically exact shape of the product. Specific tolerances are annotated to the dimensions and defaults are placed elsewhere on the drawing, typically in the title block. As advanced manufacturing systems make more use of digital product models, the shape and the tolerances must be represented completely and unambiguously (this is not to imply that engineering drawings are unambiguous, however.) As the communication of the product definition data migrates from engineering drawings to digital product models, it is important that the meaning of the dimensions and tolerances as found on engineering drawings be preserved.

The reference model contained in this document attempts to identify, define and record that meaning. It is assumed that the readers of this document have not only a working knowledge of the IDEF1X and Express modelling languages, but also a very thorough knowledge of dimensioning and tolerancing practices as described in the ANSI and ISO standards. This document is not a lesson in dimensioning and tolerancing, but captures the information content of the standard practices and concepts.

3.2.2 Scope and Viewpoint

This document provides the definition of the functional content for dimensioning and tolerancing practices as specified by ANSI Y14.5M-1986 and ISO 1101 and 1660. These specifications are considered to be functionally identical for the purposes of this effort.
SECTION 3: SHAPE VARIATION TOLERANCES

This information model is intended to completely define all tolerance information specified by ANSI Y14.5M-1982 and the corresponding ISO specifications. Any deficiencies should be commented on in writing.

The primary viewpoint of the reference model will be on the application of the tolerances specified in the above standards to three dimensional, mechanical parts.

The information model contains the functional tolerancing data set as it pertains to a single finished part model. Reference to tolerances as they affect various stages of production or preliminary forms of the part (such as casted design) are not accommodated in this model except by reference to a separate Product Model.

The tolerances described in the model reference Product Model Features as described by the Mechanical Products, Form Features and Solids Committee. These include reference to both explicit and implicit form features, topologies and geometries.

Excluded from the scope are:

- Computing tolerances
- Process tolerances
- Pictorial representations of tolerances
- Dimensioning practices (i.e. what constitutes good dimensioning practice)
- Non-mechanical part tolerances (e.g. electrical component values)
- Surface finish/surface roughness specifications
- Efficiencies regarding computer memory requirements or processing methodologies

3.3 Fundamental Concepts and Assumptions

Product models are assumed to be complete, unambiguous, three-dimensional definitions of the shape of the desired object, typically represented by a Boundary Representation (BREP) Solids model or an equivalent bounded, surfaced, Wireframe Model. The general requirement is the capability to reference shape defining elements of the surface of modelled product. The terms Area, Seam and Corner, as defined in the PDES Integration Model, are used within this model to refer to the appropriate shape element (and correspond roughly to the BREP notions of Face, Edge and Vertex.)

Geometric elements always underlie the Topologic elements Face, Edge and Vertex. (This assumption is probably moot in light of the work of the Integration Committee.)

Product Models contain exact definitions of nominal product geometry, i.e. the model is considered BASIC.

The value of the dimension is implicit in model geometry or is an explicit parameter of an implicit feature. Each Tolerance specified within a model implies an underlying dimension.

Graphical display of tolerance information will conform to ANSI Y14.5M 1982, specifically the Feature Control Frame.
SECTION 3: SHAPE VARIATION TOLERANCES

The tolerances described in this information model include both Coordinate Tolerances and Geometric Tolerances. Both type of tolerances are described as they pertain to common tolerancing practices which are used to define the form, fit and function of the modelled product.

Independent or dependent geometric constructs may be needed in addition to shape defining elements to specify certain types of tolerances. Examples of these are hole centerlines, off-part datums or other derived geometries. Therefore, it is assumed that these constructs are present/available as part of the product definition model for reference in tolerances.

All tolerances dimensions are derived from shape/size elements or explicitly stated as part of implicit features.

Note on Model Content

This version of the Tolerance Model expands on two concepts which are only cursorily addressed in earlier versions: dimensions and geometric derivations. Coordinate Tolerances, as defined in previous versions of the model, contained "extra" data that was intended to facilitate the determination of the value of the intended dimension. This version of the model flips the emphasis from the tolerance to the dimension: coordinate dimensions are defined and the tolerance values are the "extra" data.

Geometric derivations are defined to accommodate typical dimension practices which reference geometry that is not part of the shape definition of the object but is derivable from shape elements. The most common examples are centerlines of holes and the spatial intersection of two part surfaces.

Entity numbers used are local only to this model and are included for bookkeeping/review purposes.

This model constitutes an integration of version 2.1 of the model and the results of the interim meeting 15th - 18th February 1988 as reported in the technical report dated 25th March 1988. Because of the new work, some entities from version 2.1 have been deleted or replaced. Replacement entities have been assigned the same entity number as the entity they replaced. New entities have been assigned new number. In particular, the dimension entities have been assigned numbers in the 600's.

This model also constitutes an integration with other mature topical models. The work of the Integration Committee is reflected in this work.

Note on Modelling Conventions

The reference model presented in this package does not strictly conform to IDEF1X modelling techniques. While the model may be read as an IDEF1X model from a structural point of view, the following conventions have been adopted:

1. Key attributes have not been defined. The Keys in the model are strictly identificational (i.e. contain no meaning) and are included to illustrate how an entity "points to" a related entity.

2. The distinction between Identifying and Non-identifying relationships has not uniformly been made. All relationships in this model may be considered non-identifying. This does not mean, however, that the existence of an entity will not depend on the existence of some other entity; this condition still may exist. Unless an attribute has been defined as "Optional",

102
the attributes of an entity must have a value when the entity is instanced. A fallout of this convention is that Dependent and Independent entities have not been identified.

3. To simplify the maintenance of the model, some liberties have been taken with the entity numbering scheme. Where entities have been included in the model for illustrative purposes only and the definition or existence of the entity is not important to the tolerance model, then entity is given an asterisk (*) rather than a number. When an entity had to be categorized due to IDEF1X syntax constraints (e.g. TOL-ENT = AREA-TOL-ENT + SEAM-TOL-ENT + FOS-TOL-ENT) the categorizations are given the same entity number as the generic parent.

3.4 Glossary

The Definitions provided here are for general terms which are not defined with the context of the model (such as entity and attribute definitions). It should be noted that some of the definitions imply scoping statements or assumptions made about the model.

**Product** An item(s) which is(are) manufactured, or used in the manufacturing of another item.

**Product Model** A digital definition/representation of a product.

**Drawing** A pictorial (image-based, human interpretable) definition/ representation of a product.

**Dimension** A measure of the shape of product that specifies a relationship between two geometric elements of the shape definition of a product or a parameter implicit within a geometric element. The numerical value of the dimension is implicit within the model geometry or explicitly called out in an implicit feature.

**Feature** Feature is some characteristic of an object or an object representation which can be uniquely identified. The same term is defined more completely within the ANSI standard.

**Object Representation** Object representation refers to the form in which the shape of a real or conceived object is defined.

**Design Model** A Design Model is the complete definition of a product, typically at the point of "release" to manufacturing, such as an engineering drawing or product model.
SECTION 3: SHAPE VARIATION TOLERANCES
SECTION 3: SHAPE VARIATION TOLERANCES

3.5 Subject Reference Model

3.5.1 Entity Pool

This is a list of entities contained in this model: (Underlined entity numbers indicate a diagram page for the entity)

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>603</td>
<td>Angle Dimension</td>
</tr>
<tr>
<td>611</td>
<td>Angle Size Characteristic</td>
</tr>
<tr>
<td>609</td>
<td>Angle Size Dimension</td>
</tr>
<tr>
<td>610</td>
<td>Angle Size Parameter</td>
</tr>
<tr>
<td>4013</td>
<td>Angle Tolerance Range</td>
</tr>
<tr>
<td>406</td>
<td>Angularity</td>
</tr>
<tr>
<td>1023</td>
<td>Angul-Proj-Tol-Zone</td>
</tr>
<tr>
<td>506</td>
<td>Angul-Tol-Ent</td>
</tr>
<tr>
<td>5061</td>
<td>Angul-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5062</td>
<td>Angul-Tol-Ent-Area</td>
</tr>
<tr>
<td>5063</td>
<td>Angul-Tol-Ent-FOS</td>
</tr>
<tr>
<td>INT-8</td>
<td>Area</td>
</tr>
<tr>
<td>311</td>
<td>Center of Symmetry</td>
</tr>
<tr>
<td>3111</td>
<td>Center of Symmetry-Size</td>
</tr>
<tr>
<td>3112</td>
<td>Center of Symmetry-Angle</td>
</tr>
<tr>
<td>508</td>
<td>Circ-Tol-Ent</td>
</tr>
<tr>
<td>5081</td>
<td>Circ-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5082</td>
<td>Circ-Tol-Ent-Area</td>
</tr>
<tr>
<td>5083</td>
<td>Circ-Tol-Ent-FOS</td>
</tr>
<tr>
<td>407</td>
<td>Circular Runout</td>
</tr>
<tr>
<td>408</td>
<td>Circularity</td>
</tr>
<tr>
<td>409</td>
<td>Concentricity</td>
</tr>
<tr>
<td>509</td>
<td>Conc-Tol-Ent</td>
</tr>
<tr>
<td>5091</td>
<td>Conc-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5092</td>
<td>Conc-Tol-Ent-Area</td>
</tr>
<tr>
<td>5093</td>
<td>Conc-Tol-Ent-FOS</td>
</tr>
<tr>
<td>302</td>
<td>Conditioned Datum</td>
</tr>
<tr>
<td>600</td>
<td>Coordinate Dimension</td>
</tr>
<tr>
<td>401</td>
<td>Coordinate Tolerance Range</td>
</tr>
<tr>
<td>INT-20</td>
<td>Corner</td>
</tr>
<tr>
<td>507</td>
<td>Crun-Tol-Ent</td>
</tr>
</tbody>
</table>

Entity Pool in Alphabetic Order
### SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5071</td>
<td>Crun-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5072</td>
<td>Crun-Tol-Ent-Area</td>
</tr>
<tr>
<td>5073</td>
<td>Crun-Tol-Ent-FOS</td>
</tr>
<tr>
<td>6022</td>
<td>Curvi-linear Dim</td>
</tr>
<tr>
<td>410</td>
<td>Cylindricity</td>
</tr>
<tr>
<td>510</td>
<td>Cyl-Tol-Ent</td>
</tr>
<tr>
<td>5102</td>
<td>Cyl-Tol-Ent-Area</td>
</tr>
<tr>
<td>5103</td>
<td>Cyl-Tol-Ent-FOS</td>
</tr>
<tr>
<td>301</td>
<td>Datum</td>
</tr>
<tr>
<td>6102</td>
<td>Derivable Angle Size Dimension</td>
</tr>
<tr>
<td>6042</td>
<td>Derivable Size Parameter</td>
</tr>
<tr>
<td>109</td>
<td>Derived Geometry</td>
</tr>
<tr>
<td>GEO14</td>
<td>Direction</td>
</tr>
<tr>
<td>4151</td>
<td>Directrix</td>
</tr>
<tr>
<td>202</td>
<td>DT Area</td>
</tr>
<tr>
<td>201</td>
<td>DT Corner</td>
</tr>
<tr>
<td>300</td>
<td>DT Feature</td>
</tr>
<tr>
<td>304</td>
<td>DT Form Feature</td>
</tr>
<tr>
<td>203</td>
<td>DT Seam</td>
</tr>
<tr>
<td>200</td>
<td>DT Shape Element</td>
</tr>
<tr>
<td>306</td>
<td>Feature of Size</td>
</tr>
<tr>
<td>411</td>
<td>Flatness</td>
</tr>
<tr>
<td>511</td>
<td>Flat-Tol-Ent</td>
</tr>
<tr>
<td>FF001</td>
<td>Form Feature</td>
</tr>
<tr>
<td>3061</td>
<td>Form Feature of Size</td>
</tr>
<tr>
<td>110</td>
<td>GD-Inputs</td>
</tr>
<tr>
<td>111</td>
<td>GD-Inputs-GD</td>
</tr>
<tr>
<td>112</td>
<td>GD-Inputs-SE</td>
</tr>
<tr>
<td>113</td>
<td>GD-Parms</td>
</tr>
<tr>
<td>108</td>
<td>Geometric Derivation</td>
</tr>
<tr>
<td>402</td>
<td>Geometric Tolerance</td>
</tr>
<tr>
<td>GEO1</td>
<td>Geometry</td>
</tr>
<tr>
<td>FF002</td>
<td>Implicit Form Feature</td>
</tr>
<tr>
<td>6101</td>
<td>Independent Angle Size Dimension</td>
</tr>
<tr>
<td>6041</td>
<td>Independent Size Parameter</td>
</tr>
<tr>
<td>6021</td>
<td>Linear Dimension</td>
</tr>
<tr>
<td>602</td>
<td>Location Dimension</td>
</tr>
</tbody>
</table>

Entity Pool in Alphabetic Order - (Continued)
### SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4012</td>
<td>Location Tolerance Range</td>
</tr>
<tr>
<td>412</td>
<td>Parallelism</td>
</tr>
<tr>
<td>1021</td>
<td>Parl-Proj-Tol-Zone</td>
</tr>
<tr>
<td>512</td>
<td>Parl-Tol-Ent</td>
</tr>
<tr>
<td>5121</td>
<td>Parl-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5122</td>
<td>Parl-Tol-Ent-Area</td>
</tr>
<tr>
<td>5123</td>
<td>Parl-Tol-Ent-FOS</td>
</tr>
<tr>
<td>413</td>
<td>Perpendicularity</td>
</tr>
<tr>
<td>1022</td>
<td>Perp-Proj-Tol-Zone</td>
</tr>
<tr>
<td>513</td>
<td>Perp-Tol-Ent</td>
</tr>
<tr>
<td>5131</td>
<td>Perp-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5132</td>
<td>Perp-Tol-Ent-Area</td>
</tr>
<tr>
<td>5133</td>
<td>Perp-Tol-Ent-FOS</td>
</tr>
<tr>
<td>515</td>
<td>Plin-Tol-Ent</td>
</tr>
<tr>
<td>5151</td>
<td>Plin-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5152</td>
<td>Plin-Tol-Ent-Area</td>
</tr>
<tr>
<td>414</td>
<td>Position</td>
</tr>
<tr>
<td>1024</td>
<td>Pos-Proj-Tol-Zone</td>
</tr>
<tr>
<td>514</td>
<td>Pos-Tol-Ent</td>
</tr>
<tr>
<td>415</td>
<td>Profile of a Line</td>
</tr>
<tr>
<td>416</td>
<td>Profile of a Surface</td>
</tr>
<tr>
<td>102</td>
<td>Projected Tolerance Zone</td>
</tr>
<tr>
<td>516</td>
<td>Psrf-Tol-Ent</td>
</tr>
<tr>
<td>608</td>
<td>Related Angle Dimension</td>
</tr>
<tr>
<td>INT-14</td>
<td>Seam</td>
</tr>
<tr>
<td>3031</td>
<td>SF-Components</td>
</tr>
<tr>
<td>INT-2</td>
<td>Shape_Element</td>
</tr>
<tr>
<td>400</td>
<td>Shape.Tolerance</td>
</tr>
<tr>
<td>605</td>
<td>Size Characteristic</td>
</tr>
<tr>
<td>601</td>
<td>Size Dimension</td>
</tr>
<tr>
<td>303</td>
<td>Size Feature</td>
</tr>
<tr>
<td>604</td>
<td>Size Parameter</td>
</tr>
<tr>
<td>4011</td>
<td>Size Tolerance Range</td>
</tr>
<tr>
<td>4171</td>
<td>Straight-Direction</td>
</tr>
<tr>
<td>417</td>
<td>Straightness</td>
</tr>
<tr>
<td>517</td>
<td>Strght-Tol-Ent</td>
</tr>
</tbody>
</table>

Entity Pool in Alphabetic Order - (Continued)
SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5171</td>
<td>Strght-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5172</td>
<td>Strght-Tol-Ent-Area</td>
</tr>
<tr>
<td>5173</td>
<td>Strght-Tol-Ent-FOS</td>
</tr>
<tr>
<td>418</td>
<td>Total Runout</td>
</tr>
<tr>
<td>518</td>
<td>Trun-Tol-Ent</td>
</tr>
<tr>
<td>5182</td>
<td>Trun-Tol-Ent-Area</td>
</tr>
<tr>
<td>5183</td>
<td>Trun-Tol-Ent-FOS</td>
</tr>
<tr>
<td>310</td>
<td>Unconditioned Datum</td>
</tr>
</tbody>
</table>

Entity Pool in Alphabetic Order - (Continued)

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Direction</td>
</tr>
<tr>
<td>102</td>
<td>Projected Tolerance-Zone</td>
</tr>
<tr>
<td>1021</td>
<td>Parl-Proj-Tol-Zone</td>
</tr>
<tr>
<td>1022</td>
<td>Perp-Proj-Tol-Zone</td>
</tr>
<tr>
<td>1023</td>
<td>Angul-Proj-Tol-Zone</td>
</tr>
<tr>
<td>1024</td>
<td>Pos-Proj-Tol-Zone</td>
</tr>
<tr>
<td>108</td>
<td>Geometric Derivation</td>
</tr>
<tr>
<td>109</td>
<td>Derived Geometry</td>
</tr>
<tr>
<td>110</td>
<td>GD-Inputs</td>
</tr>
<tr>
<td>111</td>
<td>GD-Inputs-GD</td>
</tr>
<tr>
<td>112</td>
<td>GD-Inputs-SE</td>
</tr>
<tr>
<td>113</td>
<td>GD-Parms</td>
</tr>
<tr>
<td>200</td>
<td>DT Shape_Elment</td>
</tr>
<tr>
<td>201</td>
<td>DT Corner</td>
</tr>
<tr>
<td>202</td>
<td>DT Seam</td>
</tr>
<tr>
<td>203</td>
<td>DT Area</td>
</tr>
<tr>
<td>300</td>
<td>DT Feature</td>
</tr>
<tr>
<td>301</td>
<td>Datum</td>
</tr>
<tr>
<td>302</td>
<td>Conditioned Datum</td>
</tr>
<tr>
<td>303</td>
<td>Size Feature</td>
</tr>
</tbody>
</table>

Entity Pool in Numeric Order
SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>304</td>
<td>DT Form Feature</td>
</tr>
<tr>
<td>3031</td>
<td>SF-Components</td>
</tr>
<tr>
<td>306</td>
<td>Feature of Size</td>
</tr>
<tr>
<td>3061</td>
<td>Form Feature of Size</td>
</tr>
<tr>
<td>310</td>
<td>Unconditioned Datum</td>
</tr>
<tr>
<td>311</td>
<td>Center of Symmetry</td>
</tr>
<tr>
<td>3111</td>
<td>Center of Symmetry-Size</td>
</tr>
<tr>
<td>3112</td>
<td>Center of Symmetry-Angle</td>
</tr>
<tr>
<td>400</td>
<td>Shape Tolerance</td>
</tr>
<tr>
<td>401</td>
<td>Coordinate Tolerance Range</td>
</tr>
<tr>
<td>4011</td>
<td>Size Tolerance Range</td>
</tr>
<tr>
<td>4012</td>
<td>Location Tolerance Range</td>
</tr>
<tr>
<td>4013</td>
<td>Angle Tolerance Range</td>
</tr>
<tr>
<td>402</td>
<td>Geometric Tolerance</td>
</tr>
<tr>
<td>406</td>
<td>Angularity</td>
</tr>
<tr>
<td>407</td>
<td>Circular Runout</td>
</tr>
<tr>
<td>408</td>
<td>Circularity</td>
</tr>
<tr>
<td>409</td>
<td>Concentricity</td>
</tr>
<tr>
<td>410</td>
<td>Cylindricity</td>
</tr>
<tr>
<td>411</td>
<td>Flatness</td>
</tr>
<tr>
<td>412</td>
<td>Parallelism</td>
</tr>
<tr>
<td>413</td>
<td>Perpendicularity</td>
</tr>
<tr>
<td>414</td>
<td>Position</td>
</tr>
<tr>
<td>415</td>
<td>Profile of a Line</td>
</tr>
<tr>
<td>4151</td>
<td>Directrix</td>
</tr>
<tr>
<td>416</td>
<td>Profile of a Surface</td>
</tr>
<tr>
<td>417</td>
<td>Straightness</td>
</tr>
<tr>
<td>4171</td>
<td>Straight-Direction</td>
</tr>
<tr>
<td>418</td>
<td>Total Runout</td>
</tr>
<tr>
<td>506</td>
<td>Angul-Tol-Ent</td>
</tr>
<tr>
<td>5061</td>
<td>Angul-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5062</td>
<td>Angul-Tol-Ent-Area</td>
</tr>
<tr>
<td>5063</td>
<td>Angul-Tol-Ent-FOS</td>
</tr>
<tr>
<td>507</td>
<td>Crun-Tol-Ent</td>
</tr>
<tr>
<td>5071</td>
<td>Crun-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5072</td>
<td>Crun-Tol-Ent-Area</td>
</tr>
<tr>
<td>5073</td>
<td>Crun-Tol-Ent-FOS</td>
</tr>
</tbody>
</table>

Entity Pool in Numeric Order - (Continued)
### SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>508</td>
<td>Circ-Tol-Ent</td>
</tr>
<tr>
<td>5081</td>
<td>Circ-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5082</td>
<td>Circ-Tol-Ent-Area</td>
</tr>
<tr>
<td>5083</td>
<td>Circ-Tol-Ent-FOS</td>
</tr>
<tr>
<td>509</td>
<td>Conc-Tol-Ent</td>
</tr>
<tr>
<td>5091</td>
<td>Conc-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5092</td>
<td>Conc-Tol-Ent-Area</td>
</tr>
<tr>
<td>5093</td>
<td>Conc-Tol-Ent-FOS</td>
</tr>
<tr>
<td>510</td>
<td>Cyl-Tol-Ent</td>
</tr>
<tr>
<td>5102</td>
<td>Cyl-Tol-Ent-Area</td>
</tr>
<tr>
<td>5103</td>
<td>Cyl-Tol-Ent-FOS</td>
</tr>
<tr>
<td>511</td>
<td>Flat-Tol-Ent</td>
</tr>
<tr>
<td>512</td>
<td>Parl-Tol-Ent</td>
</tr>
<tr>
<td>5121</td>
<td>Parl-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5122</td>
<td>Parl-Tol-Ent-Area</td>
</tr>
<tr>
<td>5123</td>
<td>Parl-Tol-Ent-FOS</td>
</tr>
<tr>
<td>513</td>
<td>Perp-Tol-Ent</td>
</tr>
<tr>
<td>5131</td>
<td>Perp-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5132</td>
<td>Perp-Tol-Ent-Area</td>
</tr>
<tr>
<td>5133</td>
<td>Perp-Tol-Ent-FOS</td>
</tr>
<tr>
<td>514</td>
<td>Pos-Tol-Ent</td>
</tr>
<tr>
<td>515</td>
<td>Plin-Tol-Ent</td>
</tr>
<tr>
<td>5151</td>
<td>Plin-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5152</td>
<td>Plin-Tol-Ent-Area</td>
</tr>
<tr>
<td>516</td>
<td>Psrf-Tol-Ent</td>
</tr>
<tr>
<td>517</td>
<td>Strght-Tol-Ent</td>
</tr>
<tr>
<td>5171</td>
<td>Strght-Tol-Ent-Seam</td>
</tr>
<tr>
<td>5172</td>
<td>Strght-Tol-Ent-Area</td>
</tr>
<tr>
<td>5173</td>
<td>Strght-Tol-Ent-FOS</td>
</tr>
<tr>
<td>518</td>
<td>Trun-Tol-Ent</td>
</tr>
<tr>
<td>5182</td>
<td>Trun-Tol-Ent-Area</td>
</tr>
<tr>
<td>5183</td>
<td>Trun-Tol-Ent-FOS</td>
</tr>
<tr>
<td>600</td>
<td>Coordinate Dimension</td>
</tr>
<tr>
<td>601</td>
<td>Size Dimension</td>
</tr>
<tr>
<td>602</td>
<td>Location Dimension</td>
</tr>
<tr>
<td>6021</td>
<td>Linear Dimension</td>
</tr>
<tr>
<td>6022</td>
<td>Curvi-linear Dim</td>
</tr>
</tbody>
</table>

Entity Pool in Numeric Order - (Continued)
### SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>603</td>
<td>Angle Dimension</td>
</tr>
<tr>
<td>604</td>
<td>Size Parameter</td>
</tr>
<tr>
<td>6041</td>
<td>Independent Size Parameter</td>
</tr>
<tr>
<td>6042</td>
<td>Derivable Size Parameter</td>
</tr>
<tr>
<td>605</td>
<td>Size Characteristic</td>
</tr>
<tr>
<td>608</td>
<td>Related Angle Dimension</td>
</tr>
<tr>
<td>609</td>
<td>Angle Size Dimension</td>
</tr>
<tr>
<td>610</td>
<td>Angle Size Parameter</td>
</tr>
<tr>
<td>6101</td>
<td>Independent Angle Size Dimension</td>
</tr>
<tr>
<td>6102</td>
<td>Derivable Angle Size Dimension</td>
</tr>
<tr>
<td>611</td>
<td>Angle Size Characteristic</td>
</tr>
<tr>
<td>FF001</td>
<td>Form Feature</td>
</tr>
<tr>
<td>FF002</td>
<td>Implicit Form Feature</td>
</tr>
<tr>
<td>GEO1</td>
<td>Geometry</td>
</tr>
<tr>
<td>GEO14</td>
<td>Direction</td>
</tr>
<tr>
<td>INT-2</td>
<td>Shape Element</td>
</tr>
<tr>
<td>INT-8</td>
<td>Area</td>
</tr>
<tr>
<td>INT-14</td>
<td>Seam</td>
</tr>
<tr>
<td>INT-20</td>
<td>Corner</td>
</tr>
</tbody>
</table>

Entity Pool in Numeric Order - (Continued)
3.5.2 Entity Definitions

The entity definitions which constitute this model are presented (more or less) in the IDEF subject-entity diagram and page-pair format with additional formatting as follows:

Entity Number and Name:
The entity number is local to this model (i.e. do not correspond to any other models.)

Entity Definition:
The definition of the entity is relative to its use in the model.

Express Definition of Entity:
The (more-or-less) equivalent Express definitions for the IDEF1X model have been provided in this model. This includes those entities which were defined as parts of other modelling efforts (i.e. fall within the scope of other reference models).

Definition of Attributes:
Where attributes have been defined for entities (not including all cases of migrated keys), a definition is provided for each attribute.

Definition of Constraints:
Where specific constraints on the value of an attribute or existence of a relationship are known, an explanation of the constraint is provided to support the statements in the WHERE portion of the Express definition of the entity.

Propositions (a.k.a. Business Rules):
Propositions are statements of fact about the subject entity and its relationships to other entities. They are English-language statements of the relationships defined in the model. Sometimes propositions contain knowledge about the entity that cannot be captured in the IDEF1X or Express definitions.

Subject Entity Diagram:
The IDEF1X model of the entity which is being defined.

Declarations:
A number of attribute types and constraints are specified more than several times throughout the following entity definitions. The ones which occur most frequently are explained here:
SECTION 3: SHAPE VARIATION TOLERANCES

TYPE

Tol_MLSN : ENUMERATION OF (MAXMC, LEASTMC, REGARDLESS, NONE);
Tol_IBO : ENUMERATION OF (INSIDE, BILATERAL, OUTSIDE);

CONSTRAINTS

Tol_Ent <> [];

Tol_Ent is a SET containing the entities to which a tolerance applies. This constraint specifies that that set must not be empty. This is true for all tolerance entities which use the SET except for Profile of a Surface, where the set may be empty if the tolerance is a default tolerance.

IF SYMMETRIC(object) THEN
    Material.Condition = M OR
    Material.Condition = L OR
    Material.Condition = S;          -- Default condition
ELSE
    Material.Condition = N;

This constraint states that if an "object" (an entity, or members of an input LIST, SET, or ARRAY) is Symmetric about a point, curve or surface, then an associated Material-Condition modifier must have a value M, L or S. If the object is not symmetric, then a Material-Condition modifier is not applicable and should be N.

Tolerance > 0.0;

The meaning of this constraint is self-evident.
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Direction

Entity Number: GEO-14

A sequence of three values that define a unit vector which specifies a direction in cartesian space. The Euclidean norm is exactly 1.

This entity falls within the scope of the Geometry model, but is included here for completeness. The definition provided implies the requirements that the Tolerance Model has of the entity.

EXPRESS Definition

ENTITY Direction SUBTYPE OF (Wire_Frame_Geometry);
   X : Number;
   Y : Number;
   Z : Optional Number;
WHERE
   (X**2 + Y**2 + Z**2) = 1.0;
END_ENTITY;

Propositions:

Each Direction(GEO-14)

- defines the direction of 0.1, or more Projected-Tolerance-Zone(102).
- defines 0,1 or more Directrix's (4151) which are the normals for the planes which define Profile-Line(415) tolerances.
- defines 0, 1 or more directions (4171) of application for Straightness(417) tolerances.
- establishes the right hand rule for 0,1, or more Related Angle Dimension (608).
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Projected-Tolerance-Zone

Entity Number: 102

A Projected-Tolerance-Zone is a direction and a length value which establishes the extent of the projected tolerance zone.

EXPRESS Definition

ENTITY Projected_Tolerance_Zone;
  Direction : Direction;
  Extent : Number;
WHERE
  MEMBER(Position, 1, 1) OR MEMBER(Angularity, 1, 1) OR
  MEMBER(Parallelism, 1, 1) OR MEMBER(Perpendicularity, 1, 1);
(*
  Parallel(Direction, {Curve or Surface of Symmetry of
  Toleranced Entity}); *)
  Extent > 0.0;
END_ENTITY;

Attribute Definition:

Direction
The direction from the tolerance zone defined by the tolerance entity that the additional tolerance zone is projected. This direction must be parallel to the curve or surface of symmetry of the tolerated entity (if it is a Feature of Size) or the tolerated entity itself (if not a Feature of Size).

Extent
A real value that specifies the length of the projected tolerance zone.

Constraint:

MEMBER(Position, 1, 1) OR MEMBER(Angularity, 1, 1) OR
MEMBER(Parallelism, 1, 1) OR MEMBER(Perpendicularity, 1, 1);

For the existence of a Projected-Tolerance-Zone (PTZ) to be valid, it must be related to a Position(414), Angularity(406), Parallelism(412), or Perpendicularity(413) Tolerance.

Parallel(Direction, {Curve or Surface of Toleranced Entity})

The direction of the PTZ must be parallel to the curve or surface of symmetry if the tolerated entity is a Feature of Size, or parallel to the tolerated entity itself.

115
SECTION 3  SHAPE VARIATION TOLERANCES

DEFINES DIRECTRIX/4151

DIRECTORX/4151

PROJECTED-TOLERANCE-ZONE/102

STRAIGHT-DIRECTION/4171

DEFINES DIRECTION OF

DEFINES

ESTABLISHES RHR FOR RELATED ANGLE DIMENSION/608

Figure D-1: DIRECTION IDEFIX Diagram
Propositions:

Each Project-Tolerance-Zone(102)

- is directed by exactly one Direction(GEO-14).
- defines the projected tolerance zone for 0 or 1 Perpendicularity(413) tolerance
- defines the projected tolerance zone for 0 or 1 Angularity(406) tolerance.
- defines the projected tolerance zone for 0 or 1 Position(414) tolerance.
- defines the projected tolerance zone for 0 or 1 Parallelism(412)
- must define a projected tolerance zone for a tolerance.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-2: DIRECTION IDEF1X Diagram
**SECTION 3: SHAPE VARIATION TOLERANCES**

**Entity Name:** Geometric Derivation

**Entity Number:** 108

A Geometric Derivation is a geometric entity which is derived from shape representation elements. An actual entity corresponding to this geometric derivation may or may not be included in the design model. The key distinction is that the geometric derivation is not a definitional element of the shape of an object, but is determined from shape elements or other geometric derivations. On an actual physical object the geometric derivation is virtual geometry that is calculated from reference to "hard points" on the object (faces, edges and vertices on the physical object which correspond to the areas, seams and corners in the object representation). Examples include: centerlines and centerplanes; breakout tangency planes; spacial intersection of two surfaces.

A Geometric Derivation is a feature which may be used as the target or the origin of a tolerance/dimension.

A Geometric Derivation consist primarily of the shape elements of other Geometric Derivations used to determine the derived geometry, and a type designation indicating the type of derivation. A reference to the derived geometry may optionally be included.

**EXPRESS Definition**

```
ENTITY Geometric_Derivation SUBTYPE OF (DT.Feature);
  Type : Undefined;
  Parameters : LIST OF SELECT(Shape_Element,
                            Geometric_Derivation);
  Derived_Geometry : Optional Geometry;
END_ENTITY;
```

**Attribute Definition:**

**Type**

A (to be) enumerated list of operations that result in specified types of geometry which are derived from the input parameters. Each operation would consist of an input list of parameters, the anticipated type of result, and an algorithmic description of the operation.

**Parameters**

The inputs to the derivation operation. Most typically the parameters will be shape element entities from which the geometry is derived.

**Derived_Geometry**

The evaluated form of the Geometric Derivation may be included for completeness. The result of the derivation operation, however, takes precedence over a predefined result that is included in the definition of the entity.
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: DT-Shape-Element

Entity Number: 200

Shape elements or entities, within the scope of this model, are constructs that define touchable or viewable portions of a part. Shape elements are defined in various ways depending the type of shape representation method used. A DT-Shape-Element is a subset of Shape Element which is defined for the purposes of this model.

Shape Element entity is outside the scope of the tolerance model but is included for integration reasons; it falls within the scope of the work of the PDES Integration Committee.

EXPress Definition

ENTITY DT_Shape_Element SUPERTYPE OF (Area, Seam, Corner);
END_ENTITY;

Propositions:
Each DT-Shape-Element(200)

• must be of one Shape-Element type.
• may be a Corner(INT-20).
• may be a Seam(INT-14).
• may be a Area(INT-8).
• may be a component of the Size Feature(303).
• is a DT Feature(300).

Constraint Definition
A Corner may not be a component of a Size Feature(303).
Figure D-3: GEOMETRIC DERIVATION IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Corner (formerly Vertex)

Entity Number: INT-200

A corner defines a unique, zero-dimensional location on the shape of the product. Corner is outside the scope of the Tolerance Model, but is included for completeness and integration reasons. It properly falls within the scope of the work of the Integration Committee; see that work for a complete definition of this entity. The definition here is incomplete with respect to the work of the Integration Committee, but is sufficient for the purposes of this model.

EXPRESS Definition

ENTITY Corner SUBTYPE OF (DT_Shape_Element);
END_ENTITY;
SECTION 3. SHAPE VARIATION TOLERANCES

Figure D-4: DT SHAPE ELEMENT IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Seam

Entity Number: 104

A seam is a one-dimensional characteristic of the shape of a product, typically defined by the intersection of two areas and bounded by two corners.

Seam is outside the scope of the Tolerance Model, but is included for completeness and integration reasons. It properly falls within the scope of the work of the Integration Committee; see that work for a complete definition of this entity. The definition here is incomplete with respect to the work of the Integration Committee, but is sufficient for the purposes of this model.

EXPRESSION Definition

ENTITY Seam SUBTYPE OF (DT.Shape.Element);
END_ENTITY;

Propositions:
Each Seam(INT-14)
- is a Shape-Element(200).
- can be tolerated by 0 or 1 Angularity(406) tolerance.
- can be tolerated by 0 or 1 Circularity(408) tolerance.
- can be tolerated by 0 or 1 Circular-Runout(407) tolerance.
- can be tolerated by 0 or 1 Concentricity(409) tolerance.
- can be tolerated by 0 or 1 Parallelism(412) tolerance.
- can be tolerated by 0 or 1 Perpendicularity(413) tolerance.
- can be tolerated by 0 or 1 Profile-Line(415) tolerance.
- can be tolerated by 0 or 1 Straightness(417) tolerance.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-5: DT CORNER / CORNER IDEF1X Diagram
SECTION 3. SHAPE VARIATION TOLERANCES

Entity Name: Area

Entity Number: INT-8

An area is a two-dimensional characteristic of the shape of a product, and is a bounded portion of a geometric surface.

Area is outside the scope of the Tolerance Model, but is included for completeness and integration reasons. It properly falls within the scope of the work of the Integration Committee; see that work for a complete definition of this entity. The definition here is incomplete with respect to the work of the Integration Committee, but is sufficient for the purposes of this model.

EXPRESS Definition

ENTITY Area SUBTYPE OF (DT.Shape.Element);
END_ENTITY;

Proposition:

Each Area(INT-8)

- is a Shape-Element(INT-8).
- can be tolerated by 0 or 1 Angularity(406) tolerance.
- can be tolerated by 0 or 1 Circularity(407) tolerance.
- can be tolerated by 0 or 1 Circular-Runout(408) tolerance.
- can be tolerated by 0 or 1 Concentricity(409) tolerance.
- can be tolerated by 0 or 1 Cylindricity(410) tolerance.
- can be tolerated by 0 or 1 Flatness(411) tolerance.
- can be tolerated by 0 or 1 Parallelism(412) tolerance.
- can be tolerated by 0 or 1 Perpendicularity(413) tolerance.
- can be tolerated by 0 or 1 Profile-Line(415) tolerance.
- can be tolerated by 0 or 1 Profile-Surface(416) tolerance.
- can be tolerated by 0 or 1 Straightness(417) tolerance.
- can be tolerated by 0 or 1 Total-Runout(418) tolerance.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-6: DT SEAM / SEAM IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: DT-Feature (formerly Feature)

Entity Number: 300

A Dimension/Tolerance Feature is a categorization of things which may be the target or origin of dimension/tolerances and corresponds to the term “feature” as used in the ANSI Y14.5 standard. It is some characteristic of the shape of a product that can be uniquely identified. Categories of a DT Feature include: Shape Elements (Area, Seam, Corner), Form Features, Features of Size (a subset of which are Form Features), and Geometric Derivations. Angle and Location Dimensions reference DT Features and Size Dimensions reference Features of Size. It is an artificial classification defined for the purposes of the Tolerance Model.

EXPRESSION Definition

ENTITY DT_Feature SUPERTYPE OF (DT_Shape_Element, DT_Form_Feature, Feature_of_Size, Geometric_Derivation);
END_ENTITY;

Propositions:

A DT-Feature(300)

- may be the target of a Location Dimension (602).
- may be the origin of a Location Dimension (602).
- may be the target of a Related Angle Dimension (608).
- may be the origin of a Related Angle Dimension (608).
- may be used as a Datum(301).
- must be a DT-Shape-Element(200), DT-Form-Feature(304), Feature-of-Size(306), or a Geometric-Derivation(108).
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-7: DT AREA / AREA IDEFIX Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Datum

Entity Number: 301

A theoretically exact geometric reference to which tolerated features are related.

EXPRESS Definition

ENTITY Datum;
    Reference  : DT_Feature;
    Name       : String(2);
WHERE
    (* Note constraint on Name A..Z, AA..ZZ Except I, O, Q *)
END_ENTITY;

Attribute Definitions:

Reference
    An entity which serves as the exact definition of the datum.

Name
    An alphabetic string that provides a unique designation for the datum (range: A...Z, AA...ZZ, excluding I, O, and Q)

Proposition:

Each Datum(301)

- is defined by a DT_Feature(300).
- may be a Conditioned-Datum(302).
- may be an Unconditioned-Datum(310).
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-8: DIMENSION/TOLERANCE IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Conditioned Datum

Entity Number: 302

A conditioned datum is a datum to which a material condition modifier may apply (if the datum is defined by a Feature of Size).

EXPRESS Definition

ENTITY Conditioned_Datum SUBTYPE OF (Datum);
    Material_condition : Tol_MLSM;
    WHERE
    IF (SYMMETRIC(Datum.Reference)) THEN
        Material.Condition = M OR
        Material.Condition = L OR
        Material.Condition = S;
    ELSE
        Material.Condition = N;
    ENDENTITY;

Attribute Definitions:

Material-condition
An enumerated list that indicates the material condition at which the tolerance applies:
M - maximum material condition; L - least material condition; or S - regardless of feature size; N - not applicable.

Propositions:

Each Conditioned Datum(302)

- is a Datum(301).
- is the primary datum for 0, 1, or more Angularity(406) Tolerances.
- is the secondary datum for 0, 1, or more Angularity(406) Tolerances.
- is the tertiary datum for 0, 1, or more Angularity(406) Tolerances.
- is the primary datum for 0, 1, or more Profile-Line(415) Tolerances.
- is the secondary datum for 0, 1, or more Profile-Line(415) Tolerances.
- is the tertiary datum for 0, 1, or more Profile-Line(415) Tolerances.
- is the primary datum for 0, 1, or more Parallelism(412) Tolerances.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-9: DATUM IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

• is the primary datum for 0, 1, or more Perpendicularity(413) Tolerances.
• is the secondary datum for 0, 1, or more Perpendicularity(415) Tolerances.
• is the primary datum for 0, 1, or more Position(414) Tolerances.
• is the secondary datum for 0, 1, or more Position(414) Tolerances.
• is the tertiary datum for 0, 1, or more Position(414) Tolerances.
• is the primary datum for 0, 1, or more Profile-Surface(416) Tolerances.
• is the secondary datum for 0, 1, or more Profile-Surface(416) Tolerances.
• is the tertiary datum for 0, 1, or more Profile-Surface(416) Tolerances.
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Size Feature

Entity Number: 303

A collection of Areas which has a tolerancable geometric location that is derived from physical feature geometry and is symmetrical about that geometric location (e.g. point, axis, curve or surface.) This entity exists because a set of Form Features has not been fully defined. A subset of Form Features can be classified as Features-of-Size (e.g. Hole, Tab, Slot) and the concept of Feature-of-Size is required for tolerancing. This construct will allow the application of tolerances to shape definitions which to not included Form Features.

NOTE: The number of Areas used to define a Size-Feature depends on the particular implementation of the shape representation. In some, a single surface which wraps around to form a cylinder and joins itself at a seam would be an appropriate single component of a Size-Feature. The assumption here is that it would take at least two semi-cylindrical surfaces to define a hole Size-Feature.

EXPRESS Definition

ENTITY Size_Feature SUBTYPE OF (Feature_of_Size);
    SF_Components : LIST [2 to #] of SELECT(Area, Seam);
    WHERE
        SYMMETRIC(SF_Components);
    (* All members of LIST must be of same type. *)
END_ENTITY;

Propositions:

Each Size Feature(303)

- is a type of Feature-of-Size(306).
- is composed of 2 or more Areas(INT-8) or 2 or more Seams(INT-14).
Figure D-10: CONDITIONED DATUM IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: DT-Form-feature, Implicit Form Feature, Form Feature

Entity Number: 304, FF-002, FF-001

A referencable collection of shape elements whose name implies some stereotypical configuration. Form Feature and Implicit Form Feature entity are outside the scope of the Tolerance Model, but are included for integration reasons. The definition of the entity is with respect to and from the viewpoint of the Tolerance Model and is incomplete with respect to the Form Features model. The inclusion of this entity within the Tolerance Model should be regarded as a Planning Level View. That is, the relationship between Implicit Form Feature and the tolerance entities within the Tolerance Model will correspond to relationships between tolerance entities and Specific Implicit Features (e.g. Implicit-Thru-Hole) in the Form Features Model.

EXPRESS Definition

ENTITY DT_Form_Feature SUBTYPE OF (DT_Feature);
    Ref_Feature : Form_Feature;
END_ENTITY;

Propositions:

Each DT Form Feature(304)

- is a DT Feature.
- may be a Form Feature (FF-001).
- can be used as Form Feature of Size (3061).

Each Implicit Form Feature (FF-002)

- is defined by 0, 1, or more Size Parameters (604).
- is defined by 0, 1, or more Angle Size Parameter (610).
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-11: SIZE FEATURE IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Feature-of-Size

Entity Number: 306

A Feature of Size is a grouping of Areas or Seams which are characterized by a set of opposing Areas or Seams and a point, curve (or axis), or surface of symmetry. It has two subsets: 1) all defined Size Features(303) are Features of Size(306); 2) all Form Features which have been defined as Features of Size(306) (e.g. hole, tab, slot).

EXPRESS Definition

ENTITY Feature_of_Size SUBTYPE OF (DT.Feature);
  Type : Enumeration of (Form.Feature, Size.Feature);
  FF  : Optional DT_Form_Feature;
WHERE
  SYMMETRIC(FF);
  IF (Type = Form.Feature) THEN FF <> NULL;
  IF (Type = Size.Feature) THEN FF = NULL;
END_ENTITY.

Attribute Definition:

Type
  A Feature of Size is either a Size Feature or a Form Feature which is a symmetric about a point, curve or surface.

FF
  If a Form Feature is a Feature of Size, then it must be identified as such.

Constraint Definition:

SYMMETRIC(FF)

If the Form Feature is a Feature of Size, then it must be symmetric.

IF (Type = Form-Feature) THEN FF <> NULL;
IF (Type = Size-Feature) THEN FF = NULL;

If the Feature of Size is a Form Feature, then the FF attribute must reference some Form Feature.
If it isn't, if it is a Size Feature, the the FF attribute must be NULL.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-12: DT FORM FEATURE IDEFIX Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Propositions:
Each Feature-of-Size(306)
- is a DT Feature (300).
- may be a Size-Feature(303).
- may be a DT Form Feature (304).
- can be tolerated by 0 or 1 Angularity Tolerance (406).
- can be tolerated by 0 or 1 Circular Runout Tolerance (407).
- can be tolerated by 0 or 1 Circularity Tolerance (408).
- can be tolerated by 0 or 1 Concentricity Tolerance (409).
- can be tolerated by 0 or 1 Cylindricity Tolerance (410).
- can be tolerated by 0 or 1 Parallelism Tolerance (412).
- can be tolerated by 0 or 1 Perpendicularity Tolerance (413)
- can be tolerated by 0 or 1 Position Tolerance (414)
- can be tolerated by 0 or 1 Straightness Tolerance (417).
- can be tolerated by 0 or 1 Total Runout Tolerance (418).
- has 0, 1, or more Size Characteristic (605).
- has 0, 1, or more Angle Size Characteristic (611).
- must have at least one Size (605) or Angle Size Characteristic (611).
SECTION 3 SHAPE VARIATION TOLERANCES

Figure D-13: FEATURE OF SIZE IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Unconditioned Datum

Entity Number: 310

An Unconditioned Datum is a Datum to which a Material Condition Modifier (MCM: MMC, LMC, RFS) does not apply.

EXPRESS Definition

ENTITY Unconditioned_Datum SUBTYPE OF (Datum);
END_ENTITY;

Propositions:

Each Unconditioned Datum(310)

- is a Datum(301).
- is the primary datum for 0, 1, or more Total-Runout(418) Tolerances.
- is the co-datum for 0, 1, or more Total-Runout(418) Tolerances.
- is the primary datum for 0, 1, or more Circular-Runout(407) Tolerances.
- is the co-datum for 0, 1, or more Circular-Runout(407) Tolerances.
- is the primary datum for 0, 1, or more Concentricity(409) Tolerances.
- is the co-datum for 0, 1, or more Concentricity(409) Tolerances.
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Center-of-Symmetry

Entity Number: 311

Features of Size are characterized by a center of symmetry. The center of symmetry may be a point, curve, axis or surface about which the elements of the Feature of Size are symmetric.

This is an IDEF1X intersection entity and does not manifest itself as an Express Entity. Center of Symmetry is a role (attribute name) that a geometric entity plays with respect to a Size Dimension and an Angle Size Dimension.

Propositions:
Each Center of Symmetry(311)

• is defined by exactly one Geometry (GEO-1).
• is part of 1 or more Size Characteristic (605)
• is part of 1 or more Angle Size Characteristic (611)
• must be part of one Size Characteristic OR one Angle Size Characteristic.
SECTION 3 SHAPE VARIATION TOLERANCES

Figure D-14: UNCONDITIONED DATUM IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Shape.Tolerance

Entity Number: 400

The allowable deviation of a geometric aspect of a product from its design nominal geometry.

EXPRESSION Definition:

ENTITY Shape_Tolerance SUBTYPE OF (Tolerance);
END_ENTITY;

 Propositions:
Each Shape.Tolerance(400)

• may be a Geometric.Tolerance(402).
• may be a Coordinate.Tolerance.Range(401).
• must be a Geometric Tolerance or a Coordinate Tolerance Range.
• is a Tolerance.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-15: CENTER OF SYMMETRY IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Coordinate_Tolerance_Range

Entity Number: 401

A Coordinate_Tolerance_Range is the numeric values added to and subtracted from the nominal dimensional value calculated from the shape definition of a product. Coordinate_Tolerance_Range represent the traditional plus/minus tolerances found on dimensions on drawings.

EXPRESS Definition:

ENTITY Coordinate_Tolerance_Range SUBTYPE OF (Shape_Tolerance);
    Plus_Tol : Number;
    Minus_Tol : Number;
WHERE
    Plus_Tol > 0.0;
    Minus_Tol > 0.0;
    NOT ((Plus_Tol = 0.0) AND (Minus_Tol = 0.0));
    MEMBER(Size_Dimension,1,#) OR MEMBER(Location_Dimension,1,#)
    OR MEMBER(Angle_Dimension,1,#);
END_ENTITY;

Attribute Definitions:

Plus_Tol
The absolute value of the tolerance that is added to the nominal dimension value to establish the maximum allowable deviation of the tolerated entity from the nominal.

Minus_Tol
The absolute value of the tolerance that is subtracted from the nominal dimension value to establish the minimum allowable deviation of the tolerated entity from the nominal.

Propositions:
Each Coordinate_Tolerance_Range(401)
- is a Shape_Tolerance(400).
- may tolerance 0, 1 or more Location Dimensions (602).
- may tolerance 0, 1 or more Angle Dimension (603).
- may tolerance 0, 1, or more Size Dimension (601).
- must tolerate a Location, Angle or Size Dimension.
Figure D-16: SHAPE TOLERANCE IDEF1X Diagram
SECTION 3 SHAPE VARIATION TOLERANCES

Entity Name: Geometric_Tolerance

Entity Number: 402

A Geometric_Tolerance is a class of entities. Geometric tolerances as defined by ANSI Y14.5M 1982 tolerance the deviation of form, orientation, location, and runout, of a produced feature from its design nominal.

EXPRESS Definition:

ENTITY Geometric_Tolerance SUBTYPE OF (Shape_Tolerance):
END_ENTITY;

Propositions:

Each Geometric_Tolerance(402)

• is a Shape_Tolerance(400).

• may be an Angularity(406) tolerance.
  may be a Circular_Runout(407) tolerance.
  may be a Circularity(408) tolerance.
  may be a Concentricity(409) tolerance.
  may be a Cylindricity(410) tolerance.
  may be a Flatness(411) tolerance.
  may be a Parallelism(412) tolerance.
  may be a Perpendicularity(413) tolerance.
  may be a Position(414) tolerance.
  may be a Profile_Line(415) tolerance.
  may be a Profile_Surface(416) tolerance.
  may be a Straightness(417) tolerance.
  may be a Total_Runout(418) tolerance.

• must be one of the listed tolerances (406-418).
SECTION 3 SHAPE VARIATION TOLERANCES

Figure D-17: COORDINATE RANGE IDEF1X Diagram
Entity Name: Angularity Tolerance

Entity Number: 406

Angularity is the condition of a surface or axis at a specified angle (other than 90 degrees) from a datum plane or axis. An Angularity tolerance specifies a tolerance zone defined by two parallel planes at the specified basic angle from the datum plane or axis within which the surface or axis of the considered feature must lie.

See ANSI Y14.5M 1982, page 106, section 6.6.2
See ISO 1101, pages 18-19, section 5.9

EXPRESS Definition:

```
ENTITY Angularity SUBTYPE OF (Geometric_Tolerance);
    Tol_Ent : SET OF SELECT (Seam, Area, Feature_of_Size);
    Tolerance : Number;
    Material_condition : Tol_MLSN;
    Projection : Optional Projected_Tolerance_Zone;
    Primary_datum : Conditioned_datum;
    Secondary_datum : Optional Conditioned_datum;
    Tertiary_datum : Optional Conditioned_datum;
WHERE
    Tol_Ent <> [];
    Tolerance > 0;
    IF SYMMETRIC(Toleranced_Entity) THEN
    ELSE
        Material.Condition = N;
    END_CONDITION;
END_ENTITY;
```

Attribute Definitions:

Toleranced Entity
A set of entities to which the tolerance applies.

Tolerance
The allowable deviation of the measured dimensional value from the design nominal.

Material_condition
An enumerated list that indicates the material condition at which the tolerance applies:
SECTION 3: SHAPE VARIATION TOLERANCES

M - maximum material condition; L - least material condition; or S - regardless of feature size; N - Not Applicable.

Projection
A Projected Tolerance Zone which specifies the additional height and direction of the projected tolerance zone outside the feature boundary.

Note: Must be parallel to the tolerated features. Length of vector determines the extent of the zone.

Primary datum
A conditioned datum entity from which the dimension is measured. This datum is the most important datum relative to the tolerance.

Secondary datum
A conditioned datum entity which is the second most important datum relative to the tolerance.

Tertiary datum
A conditioned datum entity which is the third most important datum relative to the tolerance.

With the primary and secondary datums, it establishes a datum reference frame that exactly locates the tolerated feature.

Propositions:
Each Angularity (406) tolerance
• is a Geometric Tolerance (402).
• has a projected tolerance zone defined by 0 or 1 Projected Tolerance Zone (102).
• has a primary datum of exactly one Conditioned Datum (302).
• may have a secondary datum of one Conditioned Datum (302).
• may have a tertiary datum of one Conditioned Datum (302).
• tolerances 1 or more Seam (INT-14), Area (INT-8), or Feature of Size (306).
Entity Name: Circular Runout Tolerance

Entity Number: 407

Circular runout tolerance defines the maximum allowable deviation of position of a point on the toleranced feature during one complete revolution of the feature about the datum axis, without relative axial displacement of the measuring position. Where applied to surfaces constructed around a datum axis, it is used to control the cumulative variations of circularity and coaxiality. Where applied to surfaces constructed at right angles to the datum axis, circular runout controls circular elements of a planar surface.

See ANSI Y14.5M 1982, page 109, section 6.7.2.1
See ISO 1101, page 22-23, section 5.12

EXPRESS Definition:

ENTITY Circular_Runout SUBTYPE OF (Geometric_Tolerance);
  Tol_Ent : SET OF SELECT(Area,Feature_of_Size);
  Tolerance : Number;
  Primary_datum : Unconditioned_Datum;
  Co-primary_datum : Optional Unconditioned_Datum;
WHERE
  Tol_Ent <> [];
  Tolerance > 0;
  (* For each member in Toleranced_Entity
     ((CIRCULAR(member) = TRUE) OR
     (DISK(member) = TRUE));  *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
  A set of entities to which the tolerance applies.

Tolerance
  The allowable deviation of the measured dimensional value from the design nominal.

Primary_datum: A datum entity from which the dimension is measured. The primary datum is the most important datum relative to the tolerance.

Co-primary_datum
  An additional datum entity which establishes an axis with the primary datum.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-19: ANGULARITY IDEFIX Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Constraint Definitions:
For each member in Toleranced Entity

\[ ((\text{CIRCULAR}(\text{member}) = \text{TRUE}) \text{ OR } (\text{DISK}(\text{member}) = \text{TRUE})) \]

CIRCULAR is a function which returns the value TRUE if cross sections of the input entity (or each member of a list of entities) in a plane normal to the axis are circular. DISK is a function which returns a value TRUE of the input entity (or each member of a list of entities) is normal to the axis of rotation, planar, and has a circular boundary.

Propositions:
Each Circular-Runout(407) tolerance
- is a Geometric.Tolerance(402).
- has a datum of exactly one Unconditioned.Datum(310).
- may have a co-datum of one Unconditioned.Datum(310).
- tolerances 1 or more Seam(INT-14), Area(INT-8) or Feature.of.Size(306).
Figure D-20: CIRCULAR RUNOUT IDEFIX Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name:  Circularity Tolerance

Entity Number:  408

Circularity is a condition of a surface of revolution such that all points of the surface intersected by a plane perpendicular to the axis or center point are equidistant from the intersection point of the plane and the axis or center point. A Circularity tolerance defines the distance between two concentric circles within which the coordinates of the tolerated feature must lie. These circles are perpendicular to and centered on the axis of the tolerated feature.

See ANSI Y14.5M 1982, page 95, section 6.4.3
See ISO 1101, page 14, section 5.3

EXPRESSION Definition:

ENTITY Circularity SUBTYPE OF (Geometric_Tolerance);
     Tol_Ent : SET OF SELECT (Seam,Area,Feature_of_Size);
     Tolerance : Number;
WHERE
     Tol_Ent <> [];
     Tolerance > 0;
     (* CIRCULAR(Tol_Ent) = TRUE; *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
A set of entities to which the tolerance applies.

Tolerance
The allowable deviation of the measured dimensional value from the design nominal.

Constraint Definitions:

CIRCULAR(Tol_Ent) = TRUE;

The asserts that the entity(ies) to which this tolerance applies must circular (see Circular Runout for definition of function).

Propositions:

Each Circularity(408) tolerance
- is a Geometric_Tolerance(402).
- tolerances 1 or more Seam(INT-14), Area(INT-8), Feature.of_Size(306);
SECTION 3. SHAPE VARIATION TOLERANCES

Entity Name: Concentricity Tolerance

Entity Number: 409

Concentricity tolerance specifies the diameter of cylinder centered on a datum point or axis within which the center or axis of the tolerated circular or cylindrical feature must lie. The specified tolerance and datum reference apply only on a Regardless-of-Feature-Size basis.

See ANSI Y14.5M 1982, page 84, section 5.11.3
See ISO 1101, page 21, section 5.11.1

EXPRESS Definition:

ENTITY Concentricity SUBTYPE OF (Geometric.Tolerance);
   Tol_Ent : SET OF SELECT (Area, Feature_of_Size);
   Tolerance : Number;
   Cylindrical_zone : Logical;
   Primary_datum : Unconditioned_Datum;
   Co-primary_datum : Optional Unconditioned Datum;
WHERE
   Tol_Ent <> [];
   Tolerance > 0;
   (*
      CIRCULAR(Tol_Ent) = TRUE; *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
   A set of entities to which the tolerance applies.

Tolerance
   The allowable deviation of the measured dimensional value from the design nominal.

Cylindrical_zone
   A boolean (true/false) flag that indicates that the tolerance value is the diameter of a cylindrical zone within which the axis or line must lie. If false, the zone is parallelepipedic or the space between two parallel lines or planes.

Primary_datum
   A datum entity from which the dimension is measured. The primary datum is the most important datum relative to the tolerance.

Co-primary_datum
   An additional datum entity which establishes an axis with the primary datum.
Figure D-21: CIRCULARITY IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Constraint Description:

\[ \text{CIRCULAR(Tol.Ent)} = \text{TRUE}; \]

Asserts that the entity(ies) to which this tolerance applies must be circular (see Circular Runout for definition of function).

Propositions:

Each Concentricity(409) tolerance

- is a Geometric.Tolerance(402).
- has a datum of one Unconditioned.Datum(310).
- has a co-datum of 0 or 1 Unconditioned.Datum(310).
- tolerances 1 or more Seam(INT-14), Area(INT-8), or Feature.of.Size(306).
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Cylindricity Tolerance

Entity Number: 410

Cylindricity tolerance defines the distance between two coaxial cylinders within which the tolerated cylindrical feature must lie. Unlike circularity, the tolerance applies simultaneously to both circular and longitudinal elements of the surface.

See ISO 1101, page 14, section 5.4

EXPRESS Definition:

ENTITY Cylindricity SUBTYPE OF (Geometric.Tolerance);
  Tol_Ent : SET OF SELECT (Area,Feature_of_Size);
  Tolerance : Number;
WHERE
  Tol_Ent <> [];
  Tolerance > 0;
  (* CYLINDRICAL(Tol.Ent) = TRUE; *)
END_ENTITY;

Attribute Definitions:

Toleranced Entity
A set of entities to which the tolerance applies.

Tolerance
The allowable deviation of the measured dimensional value from the design nominal.

Constraint Definitions:

  CYLINDRICAL(Tol_Ent) = TRUE;

The CYLINDRICAL function returns a value TRUE if the input entity (or each member from a list of entities) is cylindrical.

Propositions:

Each Cylindricity(410) tolerance
  • is a Geometric.Tolerance(402).
  • tolerances 1 or more Area(INT-8) or Feature_of_Size(306).
Figure D-22: CONCENTRICITY IDEF1X Diagram
SECTION 3. SHAPE VARIATION TOLERANCES

Entity Name: Flatness Tolerance

Entity Number: 411

Flatness tolerance defines the distance between two parallel planes between which the tolerated surface must lie. The tolerance may be applied on a unit basis as a means of preventing abrupt surface variation within a small area of the feature.

See ANSI Y14.5M 1982, page 94, section 6.4.2
See ISO 1101, page 13, section 5.2

EXPRESS Definition:

ENTITY Flatness SUBTYPE OF (Geometric_Tolerance);
  Tol_Ent : SET OF Area;
  Tolerance : Number;
  Per_unit_square : Optional Number;
WHERE
  (Per_unit_Square > 0.0);
  Tol_Ent <> [];
  Tolerance > 0;
  (* PLANAR(Tol_Ent) = TRUE; *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
A set of entities to which the tolerance applies.

Tolerance
The allowable deviation of the measured dimensional value from the design nominal.

Per_unit_square
Specifies the side of square region on the Toleranced_Entity over which the tolerance value applies. Used to prevent abrupt surface variations in a relatively small area of the feature.

Constraint Definitions:

PLANAR(Tol_Ent) = TRUE

The PLANAR function returns a value TRUE if the input entity (or each member from a list of entities) is planar.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-23: CYLINDRICITY IDEF1X Diagram
Propositions:

Each Flatness(411) tolerance

- is a Geometric.Tolerance(402).
- tolerances 1 or more Area(INT-8).
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-24: FLATNESS IDEF1X Diagram
Entity Name: Parallelism Tolerance

Entity Number: 412

Parallelism is the condition of a surface equidistant at all points from a datum plane or an axis equidistant along the length from the datum axis. Parallelism tolerance specifies the distance between two planes or lines parallel to a datum plane or axis within which the line elements of the surface or axis of the considered feature must lie, or a cylindrical tolerance zone whose axis is parallel to the datum axis, within which the axis of the considered feature must lie. The allowable feature position zone may be cylindrical (fig 45), parallelepipedic (fig 51) or planar (fig 47,54) for line features and is similar to a flatness tolerance zone for surface features (fig 57,60). These figure references are to the ISO 1101 standard.

See ANSI Y14.5M 1982, page 106, section 6.6.3
See ISO 1101, page 15-17, section 5.7

EXPRESS Definition:

ENTITY Parallelism SUBTYPE OF (Geometric_Tolerance):
  Tol_Ent : SET OF SELECT(Seam,Area,
    Feature_of_Size);
  Tolerance : Number;
  Material_condition : Tol_MLSN;
  Cylindrical_zone : Logical;
  Projection : Optional Projected_Tolerance_Zone;
  Primary_datum : Conditioned_datum;
WHERE
  Tol_Ent <> [];
  Tolerance > 0;
  (*
    IF SYMMETRIC(Tol_Ent) THEN
      Material Condition = M OR
      Material Condition = L OR
      Material Condition = S; -- Default Condition
    ELSE
      Material Condition = N;
    *)
END_ENTITY:

Attribute Definitions:

Toleranced_Entity A set of entities to which the tolerance applies.

Tolerance

The allowable deviation of the measured dimensional value from the design nominal.
SECTION 3: SHAPE VARIATION TOLERANCES

Material condition
An enumerated list that indicates the material condition at which the tolerance applies:
M - maximum material condition; L - least material condition; S - regardless of feature size;
N - Not applicable.

Cylindrical zone
A boolean (true/false) flag that indicates that the tolerance value is the diameter of a cylindrical zone within which the axis or line must lie. If false, the zone is parallelepipedic or the space between two parallel lines or planes.

Projection
A point, a direction, and an extent which specify the additional height and direction of the projected tolerance zone outside the feature boundary.
Note: Must be parallel to the tolerated features.

Primary datum
A conditioned datum entity from which the dimension is measured.

Propositions:
Each Parallelism(412) tolerance
- is a Geometric.Tolerance(402).
- has exactly one datum of Conditioned.Datum(302).
- has a project tolerance zone defined by 0 or 1 Projected Tolerance.Zone(102).
- tolerances 1 or more Seam(INT-14), Area(INT-8), or Feature.of Size(306).
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Perpendicularity Tolerance

Entity Number: 413

Perpendicularity tolerance defines the allowable linear deviation from a true right angle of line or surface features with respect to line or surface datums. Several interpretations of the exact tolerance zone are possible for line features with respect to surface datums.

See figures 65, 67, and 69 in ISO 1101, page 17.
See ANSI Y14.5M 1982, page 106, section 6.6.4
See ISO 1101, page 17-18, section 5.8

EXPRESS Definition:

ENTITY Perpendicularity Tolerance SUBTYPE OF (Geometric_Tolerance);
   Tol_Ent : SET OF SELECT(Seam, Area, Feature_of_Size);
   Tolerance : Number;
   Material_condition : Tol_MLSN;
   Cylindrical_zone : Logical;
   Projection : Optional Projected_Tolerance_Zone;
   Primary_datum : Conditioned_datum;
   Secondary_datum : Optional Conditioned_datum;
WHERE
   Tol_Ent <> [];
   Tolerance > 0;
   (*) IF SYMMETRIC(Toleranced_Entity) THEN
      Material.Condition = M OR
      Material.Condition = L OR
      Material.Condition = S; -- Default Condition
   ELSE
      Material.Condition = N;
   *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
   A set of entities to which the tolerance applies.

Tolerance
   The allowable deviation of the measured dimensional value from the design nominal.

Material_condition
   An enumerated list that indicates the material condition at which the tolerance applies:
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-25: PARALLELISM IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

M - maximum material condition; L - least material condition; S - regardless of feature size.
N - Not Applicable.

Cylindrical zone
A boolean (true/false) flag that indicates that the tolerance value is the diameter of a cylindrical zone within which the axis or line must lie. If false, the zone is parallelepipedic or the space between two parallel lines or planes.

Projection
A point, a direction and an extent which specify the additional height and direction of the projected tolerance zone outside the feature boundary.
Note: Must be parallel to the tolerated features. Length of vector determines the extent of the zone.

Primary datum
A conditioned Datum entity from which the dimension is measured. This primary datum is the most important datum relative to the tolerance.

Secondary datum
A conditioned Datum entity which is the second most important datum relative to the tolerance.

Propositions:
Each Perpendicularity(413) tolerance
- is a Geometric.Tolerance(402).
- has Primary datum of exactly one Conditioned Datum(302).
- may have a secondary datum of one Conditioned Datum(302).
- has a projected tolerance zone defined by 0 or 1 Projected.Tolerance.Zone(102).
- tolerances 1 or more Seam(INT-14), Area(INT-8), or Feature.of.Size(306).
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Position Tolerance

Entity Number: 414

Position tolerance defines the allowable deviation of position of the center point axis or center plane of a feature of size. A center point tolerance defines the diameter of a spherical or circular zone, an axis tolerance defines the measure of a cylindrical, parallelepipedic or planar zone and a center plane tolerance defines a zone specified by the distance between two bounding parallel planes. Each zone is considered to contain the true position of the tolerated feature.

See ANSI Y14.5M 1982, page 53-89, section 5.2
See ISO 1101, page 19-20, section 5.10

EXPRESS Definition:

ENTITY Position SUBTYPE OF (Geometric Tolerance);
  Tol_Ent : SET OF Feature_of_Size;
  Tolerance : Number;
  Material_condition : Tol_MLSN;
  Cylindrical_zone : Logical;
  Projection : Optional Projected_Tolerance_Zone;
  Primary_datum : Conditioned_datum;
  Secondary_datum : Optional Conditioned_datum;
  Tertiary_datum : Optional Conditioned_datum;
WHERE
  Tol_Ent <> [];
  Tolerance > 0;
  (*
    IF SYMMETRIC(Toleranced_Entity) THEN
      Material.Condition = M OR
      Material.Condition = L OR
      Material.Condition = S;                     -- Default Condition
    ELSE
      Material.Condition = N;
  *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
  A set of entities to which the tolerance applies.

Tolerance
  The allowable deviation of the measured dimensional value from the design nominal.

Material.Condition
  An enumerated list that indicates the material condition at which the tolerance applies:
Figure D-26: PERPENDICULARITY IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

M - maximum material condition; L - least material condition; S - regardless of feature size; N - not applicable.

Cylindrical zone
A boolean (true/false) flag that indicates that the tolerance value is the diameter of a cylindrical zone within which the axis or line must lie. If false, the zone is parallelepipedic or the space between two parallel lines or planes.

Projection
A point, a direction and an extent which specify the additional height and direction of the Projected Tolerance Zone outside the feature boundary.
Note: Must be parallel to the toleranced features.

Primary datum
A conditioned Datum entity from which the dimension is measured. This primary datum is the most important datum relative to the tolerance.

Secondary datum
A conditioned Datum entity which is the second most important datum relative to the tolerance.

Tertiary datum
A conditioned datum entity which is the third most important datum relative to the tolerance. With the primary and secondary datums, it establishes a datum reference frame that exactly located the toleranced feature.

Propositions:
Each Position(414) tolerance
- is a Geometric Tolerance(402)
- has a primary datum of exactly one Conditioned Datum(302).
- may have a secondary datum of one Conditioned Datum(302).
- may have a tertiary datum of one Conditioned Datum(302).
- has a projected tolerance zone defined by 0 or 1 Projected Tolerance Zone(102).
- tolerances 1 or more Feature of Size(306).
SECTION 3. SHAPE VARIATION TOLERANCES

Entity Name: Profile of a Line Tolerance

Entity Number: 415

Profile of a line tolerance specifies the diameter of a circle which when its center or one tangent moves along the design nominal curve feature, sweeps the region in which the feature must lie. The tolerance is two dimensional and applies normal (perpendicular) to the true profile at all points. Where a sharp corner is included, the tolerance zone extends to the intersection of the boundary lines.

See ANSI Y14.5M 1982, page 97-104, section 6.5.1
See ISO 1101, page 14, section 5.5

EXPRESS Definition:

ENTITY Profile.Line SUBTYPE OF (Geometric.Tolerance);
  Tol_Ent : SET OF SELECT (Seam, Area);
  Tolerance : Number;
  Directrix : Optional Direction;
  Application : Tol_IBO;
  Primary.datum : Optional Conditioned.datum;
  Secondary.datum : Optional Conditioned.datum;
  Tertiary.datum : Optional Conditioned.datum;
WHERE
  Tol_Ent <> [];
  Tolerance > 0;
  (*
   NOT ((Tol_Ent = Edge) AND NOT PLANAR(Tol_Ent));
   IF (Tol_Ent = Edge) THEN Directrix = NUL; *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
  A set of entities to which the tolerance applies.

Tolerance
  The allowable deviation of the measured dimensional value from the design nominal.

Application
  An enumerated list that specifies the tolerance application to be I - inside, B - bilateral, O - Outside.

Primary.datum
  A conditioned.Datum entity from which the dimension is measured. This primary datum is the most important datum relative to the tolerance.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-27: POSITION IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Secondary datum
A conditioned Datum entity which is the second most important datum relative to the tolerance.

Tertiary datum
A conditioned datum entity which is the third most important datum relative to the tolerance. With the primary and secondary datums, it establishes a datum reference frame that exactly located the tolerated feature.

Directrix
Cross sections of the tolerated face taken in planes normal to the directrix establish the line profile that is tolerated. The directrix is required only if a datum is not included in the line tolerance entity.

Constraint Definitions:

(Tol_Ent = Edge) AND PLANAR(Tol_Ent);
IF (Tol_Ent = Edge) THEN Directrix = NULL;

This clause asserts that if the entity to which the tolerance applies is an Seam, the underlying curve of the Seam must be defined within a plane. (see Flatness for definition of PLANAR). It further asserts that the Directrix is meaningless and should be null if the Tol.Ent is an Seam (A value for the Directrix may be assigned if there are entities in the list which are Areas).

Propositions:
Each Profile of a Line(415) tolerance
- is a Geometric.Tolerance(402).
- may have a primary datum of one Conditioned.Datum(302).
- may have a secondary datum of one Conditioned.Datum(302).
- may have a tertiary datum of one Conditioned.Datum(302).
- may have profile determined in planes normal to one Direction(GEO-14).
- tolerances 1 or more Seam(INT-14) or Area(INT-8).
Entity Name: Profile of a Surface Tolerance

Entity Number: 416

A profile of a surface tolerance specifies the distance between two "ball-offset" surfaces located equally on either side, or totally on one side of the design nominal feature. The tolerated feature must lie between these surfaces. The tolerance is three-dimensional and applies normal (perpendicular) to the true profile at all points. Where a sharp corner is included, the tolerance zone extends to the intersection of the boundary.

See ANSI Y14.5M 1982, page 97-104, section 6.5.1
See ISO 1101, page 14, section 5.5

EXPRESS Definition:

```
ENTITY Profile_Surface SUBTYPE OF (Geometric_Tolerance);
  Tol_Ent : SET OF Area;
  Tolerance : Number;
  Default : Logical;
  Application : Tol_IBO;
  Primary_datum : Optional Conditioned_datum;
  Secondary_datum : Optional Conditioned_datum;
  Tertiary_datum : Optional Conditioned_datum;
  (* WHERE
   Tolerance >= 0;
   EMPTY(Toleranced_Entity) = Default;
   UNIQUE(Default = TRUE);
  *)
END_ENTITY;
```

Attribute Definitions:

Toleranced_Entity
  A set of entities to which the tolerance applies.

Tolerance
  The allowable deviation of the measured dimensional value from the design nominal.

Default
  A Boolean flag used to denote the instance of the Profile_Surface.Tolerance that is used as the default tolerance within the Product Model. Only one instance of this entity may have this attribute set to TRUE, and this one may not reference any toleranced entity (hence, a cardinality of 0 is allowed for this entity).
Figure D-28: PROFILE OF A LINE IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Application
An enumerated list that specifies the tolerance application to be I - inside, B - bilateral
O - Outside.

Primary datum
A conditioned Datum entity from which the dimension is measured. This primary datum is the most important datum relative to the tolerance.

Secondary datum
A conditioned Datum entity which is the second most important datum relative to the tolerance.

Tertiary datum
A conditioned datum entity which is the third most important datum relative to the tolerance. With the primary and secondary datums, it establishes a datum reference frame that exactly located the tolerated feature.

Constraint Definitions:

\[
\text{EMPTY(Tol.Ent)} = \text{Default}; \\
\text{UNIQUE(Default = TRUE)};
\]

This constraint asserts that if the Tol.Ent set is EMPTY, then the Default flag must be TRUE. In addition, the instance of the entity where this condition is true must be UNIQUE (i.e. there is only one entity which has a default flag set to TRUE.)

Propositions:
Each Profile of a Surface(416) tolerance
- is a Geometric.Tolerance(402).
- may have a primary datum of one Conditioned.Datum(302).
- may have a secondary datum of one Conditioned.Datum(302).
- may have a tertiary datum of one Conditioned.Datum(302).
- tolerances 1 or more Area (INT-8).
Entity Name: Straightness Tolerance

Entity Number: 417

Straightness tolerance defines the allowable deviation of a line or of line elements of a surface feature. The tolerance zones for line features are cylindrical, parallelepipedic and parallel planes. Straightness tolerance for surface features specify the distance between two parallel lines within which a linear element of the surface in a specified direction, must lie. The linear elements is a cross section of the surface in a plane parallel to the direction vector and normal to the surface.

See figures 27,29 and 31 in the ISO 1101 standard, page 17.
See ANSI Y14.5M 1982, page 91-94, section 6.4.1
See ISO 1101, page 13, section 5.1

EXPRESS Definition:

ENTITY Straightness SUBTYPE OF (Geometric_Tolerance);
  Tol_Ent : SET OF SELECT(Seam, Area, Feature_of_Size);
  Tolerance : Number;
  Direction : Optional Direction;
  Material_condition : Tol_MLSN;
  Cylindrical_zone : Logical;
  Per_unit_length : Number;
WHERE
  (per_unit_length > 0.0);
  Tol_Ent <> [];
  Tolerance > 0;
  (* IF SYMMETRIC(Tol_Ent) THEN
    Material_Condition = M OR
    Material_Condition = L OR
    Material_Condition = S; -- Default Condition
  ELSE
    Material_Condition = N;
    IF (Tol_Ent <> Edge) THEN LINEAR_SECTIONS(Direction, Tol_Ent);
    IF (Tol_Ent = Edge) THEN Direction = NULL: *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity
  A set of entities to which the tolerance applies.

Tolerance
  The allowable deviation of the measured dimensional value from the design nominal.
SECTION 3. SHAPE VARIATION TOLERANCES

Figure D-29: PROFILE OF A SURFACE IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Direction
A direction that specifies the straightness tolerance for linear surface elements Required when the tolerated entity is a surface. It must be parallel to the surface.

Material condition
An enumerated list that indicates the material condition at which the tolerance applies:
M - maximum material condition; L - least material condition; S - regardless of feature size;
N - Not applicable.

Cylindrical zone
A boolean (true/false) flag that indicates that the tolerance value is the diameter of a cylindrical zone within which the axis or line must lie. If false, the zone is parallelepipedic or the space between two parallel lines or planes.

Per unit length
Specifies the linear distance within which the tolerance value applies. Used to prevent abrupt changes in the direction of the tolerated entity.

Constraint Definitions:

IF (Tol_Ent <> Edge) THEN LINEAR_SECTIONS(Direction, Tol_Ent);
IF (Tol_Ent = Edge) THEN Direction = NULL;

LINEAR SECTIONS is a function which returns a value TRUE if all cross sections of the tolerated surface (or members of a list of entities) taken in planes tangent to the direction are straight lines. If the Tol_Ent contains a Seam, then the Direction does not apply. If the Toleranced_Entity list contains only Seams, then Direction must be Null.

Propositions:
Each Straightness(417) tolerance
- is a Geometric.Tolerance(402).
- may have a direction of application defined by one Direction(GEO-14).
- tolerances 1 or more Seam(INT-14), Area(INT-8), Feature.of.Size(306).
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Total Runout Tolerance

Entity Number: 418

A total runout tolerance specifies the maximum allowable deviation of position for all points on the tolerated feature during one complete revolution of the feature about the datum axis, with relative axial displacement of the measuring position. Where applied to surfaces constructed around a datum axis, it is used to control the cumulative variations of circularity, straightness, angularity, taper, profile of surface, and coaxiality. Where applied to surfaces constructed at right angles to the datum axis, total runout controls perpendicularity and flatness.

See ANSI Y14.5M 1982, page 109, section 6.7.2.2
See ISO 1101, page 22-23, section 5.12

Note that ANSI and ISO do not explicitly define total runout for non-cylindrical surfaces. It is assumed that this is due to mechanical measurement limitations.

EXPRESS Definition

ENTITY Total_Runout SUBTYPE OF (Geometric_Tolerance);
  Tol_Ent : SET OF SELECT(Area, Feature_of_Size);
  Tolerance : Number;
  Primary_datum : Unconditioned_Datum;
  Co-primary_datum : Optional Unconditioned_Datum;
WHERE
  Tol_Ent <> [];
  Tolerance > 0;
  (*) FOR EACH member IN Tol_Ent
    ((CIRCULAR(member) OR DISK(member));  *)
END_ENTITY;

Attribute Definitions:

Toleranced_Entity:
A set of entities to which the tolerance applies.

Tolerance
The allowable deviation of the measured dimensional value from the design nominal.

Primary_datum
A datum entity from which the dimension is measured. The primary datum is the most important datum relative to the tolerance.

Co-primary_datum
An additional datum entity which establishes an axis with the primary datum.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-30: STRAIGHTNESS IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Constraint Definitions:

FOR EACH member IN Tol_Ent ((CIRCULAR(member) OR DISK(member))

Asserts that each member in the Tol.Ent list must be either Cylindrical or a Disk. (See Circular_Runout for definition).

Propositions:

Each Total_Runout(418) tolerance

- is a Geometric_Tolerance(402).
- has datum of exactly one Unconditioned_Datum(310).
- may have a co-datum of one Unconditioned Datum(310).
- tolerance 1 or more Area(INT-8), Feature of Size(306).
Figure D-31: TOTAL RUNOUT IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

DIMENSIONS OVERVIEW

The analysis of dimensions brought out some very subtle aspects of the meaning of dimensions placed on engineering drawings. Ostensibly, the dimensions define the theoretically exact shape of the modelled object. However, further analysis reveals meaning imbedded within the plane of the paper in which the dimension is drawn, within the witness lines and even in the shape of the dimension leaders. The aspect of the dimension with the least meaning is the dimensional value. All of these characteristics are added together with a tolerance value(s) to communicate the allowable shape deviations of the physical object produced from the design model to the design analysts, machinists and inspectors who use them.

For example, one distinction that was implicitly made in earlier versions of the tolerance model and is more explicitly dealt with in these extensions is the difference between the thing which is tolerated versus the thing controlled by the tolerance (for the Location and Angle tolerances). The plane of the paper of a drawing represents a series of section cuts (parallel to the plane of the paper) through the object representation and each dimension is placed within one of these section cuts. Witness lines or leader arrowheads indicate the lines or planar curves within a section cut that are the origin and target for the dimension. The tolerated thing is the line or a point on the planar curve while the surface being controlled by the dimension/tolerance combination is the surface that intersects the plane of the section cut to form the line or curve. Another example is the location of a hole from a plane: the thing to which the tolerance applies is the centerline of the hole, but the thing being controlled by the tolerance is the surface of the hole.

**NOTE:** Unlike previous entity definitions, the Propositions of the dimension entities contain information that is not captured in either the IDEF or the Express entity definitions. These propositions are part of the entity definition.
Entity Name: Coordinate Dimension

Entity Number: 600

A coordinate dimension corresponds to dimensions typically found on engineering drawings. This is a classification of Location, Size and Angle dimensions.

**EXPRESS Definition**

```express
ENTITY Coordinate_Dimension SUPERTYPE OF (Size_Dimension,
     Angle_Dimension, Location_Dimension):
END_ENTITY;
```
SECTION 3: SHAPE VARIATION TOLERANCES

Entity Name: Size Dimension

Entity Number: 601

A Size Tolerance is a numeric range that constrains the value of a Size Dimension. A Size Dimension is a measure of an individual feature of the shape of an object. There are three characteristics of a Size Dimension:

1. the dimensional value (implicitly defined within the representation geometry or explicitly called out in a feature definition);
2. the shape representation elements which define the feature of interest; and
3. the center of symmetry of the elements of the feature.

The value of a Size Dimension is expressed as a magnitude and is independent of the location of the feature. The feature of interest must be a Feature of Size (see ANSI Y14.5) which are characterized by a point, curve or surface of symmetry. The value of the Size Dimension is determined by measuring in a straight line from a point on the feature through the center of symmetry (normal to the curve or surface of symmetry) to another point on the feature opposite the first. In most cases, the determination of the dimensional value is trivial, as in the diameter of a hole or the width of a slot.

EXPress Definition

ENTITY Size_Dimension SUBTYPE OF (Coordinate_Dimension);
   Tolerance : Coordinate_Tolerance_Range;
END_ENTITY;

ENTITY Size_Parameter SUBTYPE OF (Size_Dimension);
   Parameter_Value : Optional Number;
   WHERE
      MEMBER(Form_Feature,1,1);
END_ENTITY;

ENTITY Size_Characteristic SUBTYPE OF (Size_Dimension);
   Dimensioned_Entity : Feature_of_Size;
   Center_of_Symmetry : Geometry;
END_ENTITY;

Attribute Definitions:

Size_Dimension

Size_Dimension is an entity that specifies a tolerance range for a dimensional aspect of a product feature.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-32: COORDINATE DIMENSION IDEF1X Diagram
Tolerance
The numeric range that defines the allowable deviation of the size dimension.

Size Parameter
Implicit Form Features typically call out size parameters in the definition of the Feature, such as the diameter of a Hole. A Size Parameter is a type of Size Dimension that specifies that value of the parameter and a tolerance on the parameter (through its supertype Size Dimension). For the existence of this entity to be valid it must be referenced by some Implicit Form Feature.

Parameter Value
The numeric value of the size parameter for a characteristic of a Implicit Form Feature. In some cases, this value may be derivable from other parameters of the Implicit Form Features, therefore this value is left optional; thus Size Parameter is categorized as independent or derivable. This implies the restriction that Derivable Size Parameters (i.e. specifies no explicit value) must be DERIVED in an Implicit Form Feature.

Size Characteristic
Explicit shape elements may have dimensional characteristic to which a Size Dimension may be applied. Since a Size Dimension only applies to Features of Size, the shape element must be defined as a Feature of Size and associated with its Center of Symmetry. The Tolerance Range then applies across the dimensioned surfaces through the Center of Symmetry.

Dimensioned Entity
The explicit feature of the product shape to which the Size Dimension Tolerance applies. It must be a Feature of Size.

Center of Symmetry
Features of Size are characterized by a Center of Symmetry and the the dimension and tolerance apply across the center of symmetry.

Propositions:
- A Size Tolerance Range (403) applies to a Size Dimension (601)
- A Size Dimension (601) may be an explicit parameter (604) (attribute) or may be a calculated characteristic (605) of a Feature of Size.
- An Implicit Form Feature may have 0, 1, or more Size Parameters.
- A Size Parameter may be either independent or derivable.
- A Feature of Size may have 0, 1 or more Size Characteristics.
- A Size Characteristic of a FOS has one center of symmetry (COS).
- Some Form Features are Features of Size.
- A Size Feature is a Feature of Size.
SECTION 3. SHAPE VARIATION TOLERANCES

Entity Name: Location Dimension

Entity Number: 602

A Location Tolerance is a numeric range that constrains the value of a Location Dimension. A Location Dimension is the measurement of the location of one feature with respect to another feature. It is either directed (i.e. there is a distinction between the origin and the target of the dimension) or a bi-directional. The value of the Location Dimension is the distance between two parallel shape representation elements or things derived from shape representation elements (any combination of planar areas, linear seam or corners, or two elements separated by a constant distance). Each feature within a Location Dimension consists of two characteristics: the thing to which the tolerance applies (or is the origin of the dimension) and the actual shape representation element (surface/area, curve/seam or point/corner) on the object which is being controlled (or is the controlling surface of) the dimension and tolerance. These components may be the same thing, i.e. there does not have to be two distinct entities (in fact, the thing being controlled may be implied through the thing being tolerated.)

The value of the dimension is typically measured in along a linear path, although arc length and “true” dimensions require that a path be specified to determine the actual value to which the tolerance applies.

This is not to imply that location tolerances may apply only to linear, planar and offset curves and surfaces. A complex (shape representation elements that are not planar, linear or cylindrical) Area or Seam (the thing which is being controlled by the tolerance) may be algorithmically resolved (through a Geometric Derivation) to a plane or line (the thing which is tolerated), thereby satisfying the stipulations above.

EXPRESS Definition

ENTITY Location_Dimension SUBTYPE OF (Coordinate_Dimension);
  Tolerance : Coordinate_Tolerance_Range;
  Target : DT.Feature;
  Origin  : DT.Feature;
  Directed : Logical;
  Linear   : Logical;
  Path     : Optional Geometric_Derivation;
WHERE
  IF Linear THEN Path = NULL;
  IF NOT Linear THEN PATH <> NULL;
END_ENTITY;

Attribute Definitions:

Tolerance
The numeric range that defines the allowable deviation of the location dimension.
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-33: SIZE DIMENSION IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

Target
In a directed dimension, the Target is the feature of the product shape that represents the “to” entity of the dimension.

Origin
In a directed dimension, the Origin is the feature of the product shape that represents the “from” entity of the dimension.

Directed
A true or false value that indicates whether the Location Dimension should be viewed as a “from”/“to” dimension or whether no such distinction should be made. “True” means that the dimension should be interpreted “from” the origin “to” the target. “False” means that there is no precedence between the entities involved in the dimension.

Linear
In most cases, a Location Dimension will be a linear measurement between the two specified entities. “Linear” will be “true” in these instances. When the measurement follows some non-linear path, such as an “arc length” or “true” dimension, then a measurement path must be specified and “linear” will be false.

Path
In non-linear dimensional measurements, a Path for the measurement must be specified.

Constraint Definitions:
If the dimension is linear, then no path is specified. If non-linear, then a path must be specified.

The target and origin entities called out by a Location Dimension must be planar, linear or a point, OR they must be offset from one another by a uniform distance. (see propositions)

Propositions:
- A Location Tolerance applies to a Location Dimension.
- A Location Dimension may be a linear measurement of the distance between two parallel shape elements. (“Parallel” is a single valued function; points are parallel to everything.)
- A Location Dimension may be a curvi-linear measurement between shape elements.
- The Path of a curvi-linear measurement must be specified.
- A Path is a Geometric Derivation which specified a curve.
- The target and origin of a Location Dimension must:
  1. be planar, linear or a point;
  2. be a Geometric Derivation which resolves to a plane, line or point;
  3. be parallel to one another (offset, curvi-linear surfaces);
  4. the origin may be reference geometry.
Entity Name: Angle Dimension

Entity Number: 603

An Angle Tolerance is the numeric range that constrains the value of an Angle Dimension. An Angle Dimension is the measure of the orientation of one feature with respect to another feature. An Angle Dimension is either directed in the same sense as a Location Dimension, or bi-directional in the same sense as a Size Dimension (in that it applies to a single feature). The value of the dimension is the angle between two straight shape representation elements or the tiling derived from a shape representation element (planar areas, linear seams or elements which have a constant angular relationship to one another).

Each feature within a directed Angle Dimension consists of two characteristics, as in the Location Dimension:

1. the thing to which the tolerance applies (or is used as the origin for the dimension); and
2. the actual element which is being controlled (or is the controlling element for the dimension and tolerance).

These characteristics may be embodied within the same shape element, or may be separate elements in the case of complex features.

The feature within a bi-directional angle (angle size) dimension is a Feature of Size and must be composed of linear elements equally disposed about the center of symmetry.

A directed Angle Dimension must consist of two additional pieces of information in order to determine the dimensional value:

1. an orientation vector (parallel to area features and normal to seam features) to establish a right-hand-rule for
2. a flag that indicates the CW or CCW sense of measurement.

A bi-directional angle (angle size) dimension requires one additional piece of information to determine the toleranced angle value: a flag indicating that the angle of interest is the one less than 180 degrees or greater than 180 degrees.

EXPRESS Definition

ENTITY Angle_Dimension SUBTYPE OF (Coordinate_Dimension);
    Tolerance : Coordinate_Tolerance_Range;
END_ENTITY;

ENTITY Related_Angle_Dimension SUBTYPE OF (Angle_Dimension);
    Target : DT_Feature;
    Origin : DT_Feature;
    Sense : Logical;
    Orientation : Direction;
WHERE
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-34: LOCATION DIMENSION IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES

ENTITY Angle_Size_Parameter SUBTYPE OF (Angle_Dimension);
  Angle_Value : Optional Number;
WHERE
  MEMBER(Form_Feature,1,#);
END_ENTITY;

ENTITY Angle_Size_Characteristic SUBTYPE OF (Angle_Dimension);
  Dimensioned_Entity : Feature_of_Size;
  Center_of_Symmetry : Geometry;
  Characteristic_Angle : Logical;
END_ENTITY;

Attribute Definitions:

Angle Dimension
  Angle_Dimension is an entity that specifies a tolerance range for an angular dimensional aspect of a product feature.

Tolerance
  The numeric range that defines the allowable deviation of the Angle_Dimension.

Related_Angle_Dimension
  For explicit features of a product shape, a Related Angle Dimension specified the "from" and "to" entities of an Angle_Dimension.

Target
  As in a directed Location_Dimension, the Target is the feature of the product shape which is the "to" feature in a directed angle dimension.

Origin
  As in a directed Location_Dimension, the Origin is the feature of the product shape that represents the "from" entity of the dimension.

Sense
  Sense is a Logical value that indicates whether the measurement of the dimension is to be taken in a clockwise (TRUE) or counterclockwise (FALSE) direction for the Related_Angle_Dimension. This attribute must be used in conjunction with the orientation vector, otherwise it is ambiguous.

Orientation
  The Orientation Direction (vector) defines the Right-Hand-Rule to be used in determining the direction of measurement for the Angle Dimension. This is to be used in conjunction with the Sense flag.

200
SECTION 3: SHAPE VARIATION TOLERANCES

Angle.Size.Parameter
Implicit Form Features may angle size parameters in the definition of the Feature, such as the angle of a V-groove. An Angle.Size.Parameter is a type of Angle Dimension that specifies that value of the parameter and a tolerance on the parameter (through its supertype Angle Dimension). For the existence of this entity to be valid it must be referenced by some Form Feature.

Angle.Value
The numeric value of the angle parameter for a characteristic of a Form Feature. As is the case of the size parameter value, this value may be derivable from other parameters of the Implicit Form Features, therefore this value is left optional; thus Angle Size Parameter is categorized as independent or derivable. This implies the restriction that Derivable Angle Size Parameters (i.e. specifies no explicit value) must be DERIVED in an Implicit Form Feature.

Angle.Size.Characteristic
Explicit shape elements may have dimensional characteristic to which a Angle.Size.Dimension may be applied. Since an Angle.Size.Dimension only applies to Features.of.Size, the shape element must be defined as a Feature.of.Size and associated with its Center.of.Symmetry. The Tolerance Range then applies across the Center.of.Symmetry.

Dimensioned.Entity
The explicit feature of the product shape to which the Angle Size Dimension Tolerance applies. It must be a Feature.of.Size.

Center.of.Symmetry
Features.of.Size are characterized by a Center of Symmetry and the the dimension and tolerance apply across the center of symmetry.

Characteristic.Angle
Characteristic.Angle is a flag indicating whether it is the angle with a measure of less than 180 degree (TRUE) which is the intended dimension or the angle greater than 180 degrees (FALSE). Is the angle is equal to 180 degrees, then the distinction is moot, unneeded, indeterminate, don’t-worry-about-it-it’ll-probably-never-happen.

Propositions:
- An Angle Tolerance Range (4013) applies to an Angle Dimension (603).
- An Angle Dimension (603) may be a rotational measurement between two intersecting shape elements.
- An Angle Dimension (603) may be an intrinsic measure of a shape element and therefore has a center of symmetry.
- The value of the Angle Dimension is determined from the origin to the target using the right-hand-rule plus a direction (view vector) for explicit measures.
The origin and the target of an Angle Dimension must:

1. be a shape element which is planar or linear;
2. be a Geometric Derivation which resolves to a plane or a line;
SECTION 3: SHAPE VARIATION TOLERANCES

Figure D-35: ANGLE DIMENSION IDEF1X Diagram
SECTION 3: SHAPE VARIATION TOLERANCES
3.6 Issue Log

The purpose of this issues log is to track and report on perceived problems within the Tolerance Model. Each issues consists of the following information:

**NUMBER:**
A sequentially defined number identifying the issue. The number consists of a prefix indicating the year in which the issue was raised and a suffix which is simply an index.

**TITLE:**
A name for the issue.

**INITIATION DATE:**
The date the issue was raised

**INITIATOR:**
The person/project/meeting that raised the issue.

**STATUS:**
Indicates the current activity on the issue. The following states have been defined:

- **Unresolved**: No work is being done to resolve the issue.
- **In work**: A resolution to the issue is actively being sought and an individual or group.
- **Preliminary**: A solution has been proposed and is currently under review. Includes proposal date and proposing individual/committee.
- **Resolved**: A proposal has been accepted which adequately solves or satisfies the issue. A Resolved status includes the date of resolution.

**RELATED ISSUES:**
A list of numbers of the issues which have bearing on this issue.

**DESCRIPTION:**
The body of text describing the issue. This includes not only an explanation, but also identifies the pertinent entities/attributes/relationships and any individuals involved.
OPTIONS & EVIDENCE:
This section lists identified solutions and the arguments for and against each one.

<table>
<thead>
<tr>
<th>Option #</th>
<th>Description of solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pros</td>
<td>Arguments in favor of solution</td>
</tr>
<tr>
<td>Cons</td>
<td>Arguments against solution</td>
</tr>
</tbody>
</table>

OPTION PROPOSED:
An option selected from the above list which is the recommended resolution to the issue.

EXPLANATION:
Rationale for selecting the proposed solution.

DECISION:
The consensus that the proposed solution should be approved or disapproved. Also included the identity of the decision making body.

DECISION DATE:
Date of decision.

ACTION:
The decision making body will identify the individual or committee responsible for implementing an approved solution.
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.1 ANGLE TOLERANCE

INITIATION DATE: January 1986
INITIATOR: Mfg Tech Committee, San Diego Meeting
STATUS: Resolved
RELATED ISSUES: Superseded by 87.2, 88.26
DESCRIPTION: The method used to provide the necessary information to calculate the intended angular measure for an Angle tolerance were points which lie on the intersection of the two entities involved in the tolerance at which tangent vectors to the entities were determined. The angle between these two vectors indicated the angle to which the tolerance applies. There is some question as to whether the information provided is sufficient.

ISSUE OPTIONS & EVIDENCE:

Option 1: Define a set of coordinate triplets (three distinct points) that are associated with the Angle Tolerance, the Toleranced Entity and the Tolerance Origin. One point of each triplet will lie on the intersection of the Toleranced Entity and the Tolerance Origin; one point will lie on the Toleranced Entity; the last point will lie on the Tolerance Origin. Each triplet will specify the point at which the tolerance applies and allow the calculation of the value intended dimension (the intersection point is the vertex of the angle).

Pros: This satisfies the perceived information requirements to define an Angle Tolerance.
Cons: The concept and model of the proposed solution is somewhat complicated.

Option 2: Remove all coordinates and simply have the Angle Tolerance reference the Toleranced Entity and the Tolerance Origin, leaving the interpretation up to the individual systems.

Pros: Simple and more direct. Most Angle Tolerances will be simple and unambiguous, making interpretation of the Angle Tolerance a relatively simple matter.
Cons: There is room for error, misinterpretation, and inexactness when tolerancing the angle between two complex surfaces (such as a Ruled Surface and Plane). This approach does not have the completeness required to communicate a complex Angle Tolerance.

OPTION PROPOSED: Option #1. The model implementing this solution has been incorporated into the reference model.

DECISION: The Tolerance Application Committee decided that the proposed solution more completely represents the intention of Angle Tolerances. The coordinate triplets are optional associations that may be included in the definition of an Angle Tolerance for more specificity, but do not have to be included for simple cases.

DECISION DATE: 19th December 1986
ACTION: Tolerance Application Committee has incorporated the approved solution into the Tolerance Reference Model
NOTE: Resolution superseded by subsequent model development.
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.2 SIZE TOLERANCED ANGLE
INITIATION DATE: August 1986
INITIATOR: Mechanical Products Committee, Peoria Project.
STATUS: Resolved
RELATED ISSUES: N/A
DESCRIPTION: The definition for a Size Tolerance states that the tolerance value applies to a dimension of the tolerated feature which is independent of the feature's location. This definition allows the size of an angle to be tolerated as well. This added interpretation seems to logically make sense, but the model does not explicitly allow for it.

ISSUE OPTIONS & EVIDENCE:

Option 1: Expand the definition of the Size Tolerance to include the tolerancing of the size of the angle between two entities which are symmetrically disposed about a center plane, line, axis or point. This may require the addition of a flag to the Size Tolerance Entity to denote that the tolerance applies to the angle.

Option 2: Create an “Angle Dimension” which is subtyped as a Related Angle Dimension and an Angle Size Dimension. See the definition of Angle Dimension (604)

OPTION PROPOSED: Option 2
DECISION: Option 2 has been incorporated as a category of Angle Dimension.
DECISION DATE: 13th July 1988, Denver
ACTION: Tolerance Committee

208
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.3 APPLICATION OF TOLERANCES TO TOPOLOGY VS GEOMETRY
INITIATION DATE: Many different times (for the record, 19th Dec 86)
INITIATOR: Again, many (for the record, Tolerance Appl. Com.)
STATUS: Resolved
RELATED ISSUES: N/A
DESCRIPTION: Are tolerances more appropriately applied to topology or geometry entities? Topology all by itself defines logical connectivity, whereas geometry defines the mathematical shape of an object. It would seem that since tolerances control the shape of the produced object and it is the mathematical definition of that shape which is subject to variation, the tolerances would more appropriately be applied to geometry entities. Geometry alone, however, does not adequately describe the shape of a part because the boundaries of the geometric elements (if, indeed, bounded) do not necessarily correspond to real boundaries of the faces on a physical object, or may correspond to several.

ISSUE OPTIONS & EVIDENCE:

Option 1: Apply Tolerances to geometry entities rather than topology entities.
Pros: This option is intuitively pleasing because the tolerance is compared to the measured deviation of the mathematical (geometric) definition of a surface (or other feature) on a produced (physical) object from the theoretical, mathematical (geometric) definition of the corresponding surface (or other feature) in the (solids) model of the object. It is the geometric definitions of the surfaces which are compared.
Cons: The problem with this option is that geometry functions in many different roles, only one of which is defining the shape of an object. Even the geometry which defines the shape sometimes is not specific enough to be used in conjunction with a tolerance. Several different faces may share the same surface definition and it would be incorrect to imply that both faces must always share the same tolerance as well. Also, a face may define a portion of an unbounded surface, and a tolerance applied to an unbounded surface makes no sense because unbounded surfaces cannot be measured.

Option 2: Leave approach as-is a enhance the explanation of the intention of the Faces, Edges and Vertices described in the model.
Pros: The concept of Face, Edge and Vertex adequately serve the needs of the Tolerance Model. To impose the constraints implied by these constructs on geometric elements would be difficult, awkward and confusing. The explanation within the Model Assumptions states that Topologic elements always have underlying geometric definitions. Also stated within the Model Assumptions is that the constructs Face, Edge and Vertex as defined with a BREP Solid Model are not required, but that their functional equivalent is required. This allows the Tolerance Model to reference Faces/surfaces which correspond to actual, physical, touchable portions of a produced object.
SECTION 3: SHAPE VARIATION TOLERANCES

Cons: Implies a requirement for BREP Solid Model. Is intuitively unappealing because topology doesn't always have shape (e.g. schematics.)

Option 3: Rename entities in accordance with work of Integration Committee.

Pros: The concept of "Shape Elements" as defined by the Integration committee corresponds exactly with the use of "Face, Edge and Vertex" within the Tolerance Model. By removing the BREP-ish nature of the models "integration entities", this issue becomes moot.

Cons: None

OPTION PROPOSED: Option 3. The Tolerance Models use of topological terms mislead and confused the intent. The concept of Shape Elements embody the correct viewpoint of the model and make this issue moot.

DECISION: The Tolerance Application Committee has approved the solution.

DECISION DATE: 29th March 1988

ACTION: Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.4 TOLERANCES TO/FROM DERIVED GEOMETRY

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee, St. Louis
STATUS: Resolved
RELATED ISSUES: Superceded by 88.1
DESCRIPTION: The point was made that some tolerances apply to geometry entities which do not have a corresponding topology entity; gauge points and breakout tangency planes were cited.

ISSUE OPTIONS & EVIDENCE:

Option 1: Add an entity called Derived geometry as a candidate Toleranced Entities and Tolerance Origins for coordinate tolerances. The stipulation is that the Derived geometry must be calculable from product topology (e.g., spatial intersection curves, centerlines). Each type of Derived geometry must be defined.
Pros: Satisfies perceived need for Tolerance to/from off-part geometry.
Cons: Open ended and ill-defined. Similar to the dilemma of form feature enumeration.

OPTION PROPOSED: Option 1.
DECISION: The Tolerance Application Committee agreed that the need for such a construct exists. However, the time to fully explore this topic is not currently available. Therefore, an entity called Derived Geometry will be added and treated like Faces, Edges, Vertices, Size Features and Features of Size. An explanation of the intent and purpose of this entity will be included, with the assertion that it will be analyzed in the future, but no further definition of it will be made. See example for the purpose and use of this entity.
DECISION DATE: 19th December 1986
ACTION: Tolerance Application Committee
NOTE: The proposal made in the resolution to this issue has been expanded upon in the work of the committee dated 25th March 1988. See 88.1 for more details.
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.5  TOLERANCE ENTITY DISPLAY
INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee, St. Louis
STATUS: Preliminary
RELATED ISSUES: N/A
DESCRIPTION: A practical requirement of all entities is the ability to see and manipulate the entities. How are Tolerance Entities to be displayed? This raises two questions:
1) will an answer to this question be of value; and
2) is this the responsibility of the Tolerance Application Committee?
The answer to 1) is obviously yes; in fact it is almost a requirement. The answer to 2) is no, it is not the responsibility of the TAC.

ISSUE OPTIONS & EVIDENCE:

Option 1: Recommend to some authoritative body that they take action on this subject and await a solution.
Pros: This problem will be faced by many application committees and should be addressed level common to all of them.
Cons: It may never get done with this approach.

Option 2: Add three pieces of information to the generic Tolerance Entity (which thereby applies to all tolerance entities):
1) a coordinate (point) location on the toleranced entity; 2) a coordinate (point) location of the tolerance symbol; and
3) an orientation (vector or plane) in which the symbol is visible.
Pros: This provides sufficient information for the display of Tolerance Entities.
Cons: This solution may not be appropriate for all application committees.

OPTION PROPOSED: Option 2.
DECISION: After a long waiting period during which no more appropriate committee addressed this problem, the Tolerance Committee decided to define the information needed to display a tolerance entity. A proposal will be prepared outlining the approach along the line suggested in Option 2.
DECISION DATE: 29th March 1988
ACTION: The Tolerance Committee.
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.6 TOLERANCES TO/FROM PART-IN-PROCESS SHAPE

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee, St. Louis
STATUS: Resolved
RELATED ISSUES: N/A
DESCRIPTION: Can Tolerances span intermediate part shapes? Surfaces are often machined based on dimensions from machined tooling surfaces which will not be present in the final part.

ISSUE OPTIONS & EVIDENCE:

Option 1: Take no action
Pros: Doesn't really seem to be a tolerance, but more of a BASIC dimension. In any case, it is outside the scope of this model.
Cons: N/A

OPTION PROPOSED: Option 1. At his time there appears to be no need to pursue this issue further.
DECISION: Adopt Option 1.
DECISION DATE: 19th December 1986
ACTION: None.
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.7  BASIC DIMENSIONS

INITIATION DATE: Various (f.t.r. 19th December 1986)
INITIATOR: Various (f.t.r. 19th December 1986)
STATUS: Resolved
RELATED ISSUES: 87.2
DESCRIPTION: Should BASIC dimensions as described in the standards be addressed within this model?

ISSUE OPTIONS & EVIDENCE:

Option 1: Take no action.
Pros: BASIC dimensions are currently cursorily addressed within the model.
Cons: BASIC dimensions are part of the standard and should be accommodated within the reference model.

Option 2: Remove any reference to BASIC dimensions from model.
Pros: Because of the assumption that the geometry in the product model defines the theoretically exact shape, the model is in effect BASIC and, therefore, BASIC dimensions do not need to be explicitly addressed.
Cons: BASIC dimensions are part of the standard and should be accommodated within the reference model.

Option 3: Enhance the capability of the reference model to handle BASIC dimensions.
Pros: Will satisfactorily address the issue by coming up with a thought out answer.
Cons: Will take more time than is available, and the issue is not crucial to the usefulness of the reference model.

OPTION PROPOSED: Option 2. The Tolerance Application Committee felt that the argument in favor of Option 2 was the strongest. Option 1 would be adequate, but it would open the door to confusion. Option 3 was deemed unsatisfactory in interest of time and importance.

DECISION: All references to BASIC dimensions in the reference model will be removed. A statement explaining the role of BASIC dimensions within the approach embodied by this model will be added to the assumptions.

DECISION DATE: 19th December 1986
ACTION: Tolerance Application Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.8  DEFAULT TOLERANCES
INITIATION DATE: Various (f.t.r. 19th December 1986)
INITIATOR: Various (f.t.r. 19th December 1986)
STATUS: Resolved
RELATED ISSUES: N/A
DESCRIPTION: Default tolerances of some kind are present on virtually all drawings. How is the need for this capability addressed within the model? It was agreed within the Tolerance Application Committee that the appropriate default tolerance for a three dimensional geometric model would be a Profile of a Surface tolerance applied to all otherwise untoleranced surfaces of the model.

ISSUE OPTIONS & EVIDENCE:

Option 1: Include statement in Assumptions that Profile of a Surface tolerance would apply to all otherwise untoleranced surfaces of the model. The value of the tolerance would be located somewhere else within the product model.

Pros: This is the current approach taken in the reference model and would require no modifications to it.
Cons: The value of the default tolerance is not readily available. It should not be included in another reference model, but should be located within the Tolerance model. This option does not address the possibility of other kinds of default tolerances.

Option 2: Modify the Profile of a Surface entity to include a flag indicating whether it is default or not. A stipulation would be included in the definition of the entity which states that only one Profile of a Surface entity with the default flag set TRUE may be present within a given product model.

Pros: The default tolerance is explicitly addressed by the reference model and it is clear what the default tolerance is.
Cons: The possibility that two or more entities may exist which are identified as default is not prohibited other than by the stipulation in the definition. Does not address the possibility of other kinds of default tolerances.

Option 3: Further investigation into default tolerances.

Pros: Will provide a more complete solution.
Cons: Time constraints prohibit a thorough investigation.

OPTION PROPOSED: Option 2. The Tolerance Application committee felt that the argument for Option 2 and against Option 1 were strong enough to select Option 2. Option 3 was not considered at this time due to time constraints. It is recognized that the argument against Option 1 and 2 about other kinds of default tolerances is valid. At some point in the future, when Option 3 becomes feasible, it is likely that the approach described by Option 2 will be carried over to other kinds of tolerances.

DECISION: The reference model will be changed to reflect the approach described in Option 2.

215
SECTION 3: SHAPE VARIATION TOLERANCES

DECISION DATE: 19th December 1986
ACTION: Tolerance Application Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86 9

RELATIONSHIP BETWEEN FORM FEATURE, FEATURE.OF.SIZE, AND SIZE FEATURE

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee
STATUS: Preliminary
RELATED ISSUES: N/A
DESCRIPTION: A Form Feature is a collection of geometric/topologic elements with a name that connotes a special shape or meaning (e.g. hole, slot, groove). Feature.of.Size is the name of a classification of things which exhibit symmetry about a surface, curve or point and is defined for the Tolerance Application referenced model (i.e. it will not be used in other reference models.) A Size Feature is the name of an entity defined within the Tolerance Application reference model to serve as a surrogate in the absence of defined Form Features. A Size Feature is a Feature.of.Size and the idea of Feature.of.Size is required for the Tolerance Application. The Tolerance Application Committee agrees on the fundamental relationship between these concepts. The relationship is described by the following Venn diagram (examples in each set are included):

A Size Feature is a Feature.of.Size but is not a Form Feature. Some Form Features are Features.of.Size. A hole is both a Form Feature and Feature.of.Size. IDEF-IX cannot model this situation, so the issue is "what is the best way to model this relationship?"

ISSUE OPTIONS & EVIDENCE:

Option 1: Redefine terms.
Option 2: Violate IDEF-IX and allow a categorization to have two generic parents.
Option 3: Create an association (intersection entity) between categories of Form Feature and Feature.of.Size to achieve this dual-parent relationship.
Option 4: Since Feature.of.Size is only used in the Tolerance Application reference model, create an optional relationship between a Form Feature and a category of Feature of Size such that whenever a Form Feature is used as a Feature.of.Size, it is identified as such.

The Pros and Cons of the above options consists primarily of pictures and are available from the chairman.

OPTION PROPOSED: The current, working solution is Option 4. Options 1 and 2 were deemed inadequate by the Tolerance Application Committee. Option 3 was awkward. Option 4 satisfied the current needs of the Tolerance Application reference model.
SECTION 3: SHAPE VARIATION TOLERANCES

DECISION: The changes described in Option 4 have been incorporated into the reference model. However, the Tolerance Application Committee does not recommend this as a firm solution because it does describe the same situation as in the Venn diagram. A final decision will await comments and review.

DECISION DATE: N/A

ACTION: Tolerance Application Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.10  
APPLICATION OF MATERIAL CONDITION MODIFIER

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee
STATUS: Resolved
RELATED ISSUES: N/A
DESCRIPTION: The Material Condition Modifier (MCM - M/MMC: Maximum Material Condition; L/LMC: Least Material Condition; S/RFS: Regardless of Feature Size) only is applicable when the feature of interest (i.e. the tolerated entity, tolerance origin, or datum) is a Feature of Size. An analysis of ANSI Y14.5M 1982 turned up no counter examples. In fact, section 2.8 specifically states that the MCM applies to feature subject to variation in size.

ISSUE OPTIONS & EVIDENCE:

Option 1: Categorize Datum into Conditioned and Unconditioned; the conditioned Datum contain the MCM indicator and is defined by a Feature of Size. The MCM indicator with the Tolerance Entities is changed to include an "N" for not applicable and a stipulation is added to the definition explaining when it is and is not applicable.

Pros: Satisfies the problem presented by the issue.
Cons: None to speak of.

OPTION PROPOSED: Choice of 1. Other options were not examined because the Tolerance Application Committee felt that this solution satisfactorily addressed the issue.

DECISION: Change the reference model to reflect the changes described in Option 1.

DECISION DATE: 19th December 1986
ACTION: Tolerance Application Committee
ISSUE: TOL-86.11 LOCATION TOLERANCE COORDINATES

INITIATION DATE: 19\textsuperscript{th} December 1986

INITIATOR: Tolerance Application Committee

STATUS: Resolved

RELATED ISSUES: 86.1, 87.2, 88.26

DESCRIPTION: There is an ambiguity in the meaning and intent of the coordinate associated with the Toleranced Entity and the Tolerance Origin of the Location Tolerance. Does a dimension exist in 3-space or is it a strictly planar idea? What is the relationship between the coordinates on either entity? Do they exist to be more specific on the intention of the tolerance or merely to facilitate the calculation of the desired dimension?

With the idea of "path," the answer to the last question is both. The two coordinates and path are included in the definition of location tolerance to provide information which is implicit in a location tolerated dimension on a drawing but not readily identifiable. The plane of the paper, leaders, witness lines and the view of the product all provide information beyond "this is so far from that plus-or-minus this."

ISSUE OPTIONS & EVIDENCE:

Option 1: To reduce possible ambiguity in meaning and make simple cases more straightforward, the Location Tolerance should be allowed to tolerance only a single entity and the coordinates should be made optional.

Pros: Restricting the Location Tolerance to a single tolerated entity reduces the possibility of misinterpretation of the meaning of the coordinates. Optional vs. required coordinates makes simple cases (e.g. distance between two parallel planes) more straightforward.

Cons: Doesn’t fully address the problem.

Option 2: Expand on the idea of a “location dimension” which explicitly contains information that allows a simple determination of the value to which the tolerance applies. Details of this option are contained in the description of the location dimension in the Tolerance Model Version 3.0.

Pros:

Cons:

OPTION PROPOSED: Option 2.

DECISION: The adoption of Option 2 actually makes the issue moot. Formal acceptance of the resolution closes this issue.

DECISION DATE: 13\textsuperscript{th} July 1988

ACTION: Tolerance Application Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.12  CONSTRAINTS BETWEEN TOLERANCES

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Should constraints between tolerances be modelled? Some tolerances imply or take into account other tolerances (a cylindricity tolerance also controls the circularity of the tolerated entity.)

ISSUE OPTIONS & EVIDENCE:

Not applicable.

OPTION PROPOSED: Not Applicable.
DECISION: The Tolerance Application Committee has left this issue open. Such situations do exist and are identified in the standard, but the worth of including them in the reference model was questionable. Addressing them would eliminate the possibility of over-toleranced or conflicting tolerated features, but it was felt that this was a question of good D & T practice and outside the scope of the reference model. A formal decision will be made when comments and reviews have been made.

DECISION DATE: Not Applicable
ACTION: Not Applicable
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.13 DATUM TARGETS AND TAPER TOLERANCES
INITIATION DATE: 19th December 1986, January 1988
INITIATOR: Tolerance Application Committee, Bob Parks
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Datum target and Taper Tolerances as specified in the ANSI standard (sections 2.13, 2.14, pgs. 25-27) are not addressed by the reference model.

ISSUE OPTIONS & EVIDENCE:

Not applicable.

OPTION PROPOSED: Not Applicable
DECISION: The Tolerance Application Committee recognizes this as a deficiency. A solution will be developed in the future.
DECISION DATE: Not Applicable
ACTION: Not Applicable
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.14          FORMAT OF DELIVERED REFERENCE MODEL

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee
STATUS: Preliminary
RELATED ISSUES: N/A
DESCRIPTION: IDEF-1X has been chosen as the language for the reference models developed for PDES. Does the Logical Layer Committee require syntactically correct IDEF-1X model or can conventions be adopted within delivered models? The current reference model is not in a fully developed IDEF-1X format and contains syntactic errors. Specifically, the model does not identify Key attributes of entities (rather uses "surrogate" keys: XXXX-id), does not distinguish between identifying and non-identifying relationships, and does not distinguish between independent and dependent entities (among others.)

ISSUE OPTIONS & EVIDENCE:

Option 1: Revise model in accordance with IDEF-1X rules.
Pros: ?
Cons: Will take a considerable amount of time to convert the model.

Option 2: Leave reference model as-is and explain the author conventions adopted within the reference model in the assumptions section.
Pros: The deviations from IDEF-1X are consistent and not major with respect to the activities of the Logical Layer Committee. The reference model may still be read as an IDEF-1X model.
Cons: Uses of the reference model which require a complete, syntactically correct reference model will be unable to use the Tolerance Application reference model.

OPTION PROPOSED: Option 2. Due to time considerations, the reference model should delivered in as-is condition. There has been no identified reason for delivering a model which completely adheres to IDEF-1X.

DECISION: The Tolerance Application Committee has decided to adopt the solution proposed in Option 2. Conversion of the model to pure IDEF-1X will be considered for future versions of the model.

DECISION DATE: 19th December 1986
ACTION: Tolerance Application Committee

223
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.15  SORICITING REVIEW OF MODEL
INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The size of the Tolerance Application reference model is intimidating (although the content is not overwhelming) and will inhibit quality review. What is the best way to solicit a good review of the model?

ISSUE OPTIONS & EVIDENCE:

Not applicable.

OPTION PROPOSED: Not Applicable
DECISION: Not Applicable
DECISION DATE: Not Applicable
ACTION: Not Applicable
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-86.16  TYPES OF DATUM ENTITIES

INITIATION DATE: 19th December 1986
INITIATOR: Tolerance Application Committee
STATUS: Unresolved
RELATED ISSUES: 88.17
DESCRIPTION: The types or nature of entities (e.g. "planar" face) which may be tolerated by a given tolerance entity are constrained. Should the type of entities which serve as datums for given tolerances be similarly constrained?

ISSUE OPTIONS & EVIDENCE:

Not applicable.

OPTION PROPOSED: Not Applicable
DECISION: Not Applicable
DECISION DATE: Not Applicable
ACTION: Not Applicable
SECTION 3 SHAPE VARIATION TOLERANCES

ISSUE: TOL-87.1 UTILITY OF TOLERANCING EDGES AND VERTICES

INITIATION DATE: 21st August 1987
INITIATOR: Bob Johnson
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Since "faces" of a part are the features which are actually produced and inspected, is there utility in applying tolerances to edges and vertices? What do tolerances applied to edges of vertices really mean; doesn't such a tolerance imply a tolerance to or control of the adjacent faces? Isn't the ideal situation for CIM to work only to the surfaces of the part?

ISSUE OPTIONS & EVIDENCE:

Not applicable.

OPTION PROPOSED: Not Applicable
DECISION: Not Applicable
DECISION DATE: Not Applicable
ACTION: Not Applicable
SECTION 3: SHAPE VARIATION TOLERANCES

**ISSUE:** TOL-87.2  
**DIMENSIONS**

**INITIATION DATE:** 21st August 1987  
**INITIATOR:** Bob Johnson  
**STATUS:** Resolved  
**RELATED ISSUES:** 86.1, 88.26, 88.5  
**DESCRIPTION:** The tolerances defined in this reference model imply underlying dimensions of the geometric model, particularly the Coordinate Tolerances. While dimensionality is inherent in the model geometry and dimensions can be derived, the fact that a dimension has been called out by a designer implies more meaning than simply the dimensional value. It implies a critical relationship between features on the part which must then be checked by QA. The implicitness of the dimension in the tolerances buries the significance of specifying a dimension. Therefore, the aspect of "dimensions" as incorporated in the Tolerance reference model should be explained in greater detail.

**ISSUE OPTIONS & EVIDENCE:**

Option 1: Expand on the idea of Location, Angle and Size dimensions. Version 3.0 of the Tolerance Model contains a proposed expansion on the idea of dimension.

**OPTION PROPOSED:** Option 1.

**DECISION:** The proposal appears to satisfy the concerns expressed in this issue, although it is recognized that deeper implications of this issue may be missed. Any omissions will be documented as new issues.

**DECISION DATE:** 13th July 1988

**ACTION:** Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

**ISSUE:** TOL-87.3   ALGORITHMIC DEFINITION OF FEATURES

**INITIATION DATE:** 21st August 1987

**INITIATOR:** Bob Johnson

**STATUS:** Unresolved

**RELATED ISSUES:** N/A

**DESCRIPTION:** The features of a part to which tolerances apply and which are used for datums need to be algorithmically defined in order to rigorously define the meaning of a tolerance. A simple statement that a tolerance applies to a cylindrical feature does not convey enough information; a procedure should be defined to evaluate just what is a cylindrical feature. Datum reference frames also need to be constructed according to a pre-defined procedure. A procedural approach would ensure that the feature, datum or tolerance is interpreted exactly as intended.

**ISSUE OPTIONS & EVIDENCE:**

Not applicable.

**OPTION PROPOSED:** Not Applicable

**DECISION:** Not Applicable

**DECISION DATE:** Not Applicable

**ACTION:** Not Applicable
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.1 DERIVED GEOMETRY

INITIATION DATE: 13th January 1988

INITIATOR: Mark Dunn, James Blaha

STATUS: Preliminary

RELATED ISSUES: 86.4

DESCRIPTION: Issue 86.4 talked about the use of off-part geometry which was derivable from part geometry as the target or the origin of a tolerance. At the time a "stub" was included to introduce the idea into the Tolerance Model. The problem has arisen again and must be resolved: how to include derived geometry in the model.

ISSUE OPTIONS & EVIDENCE:

Option 1: Develop a construct which algorithmically defines non-part geometry based upon part geometry.

Pros:
Cons:

Option 2: Develop a construct that associates the components of the shape model with a derivation type designation and (optionally) the derived result. The derivation type designation would reference an externally defined algorithm which will produce the result from the input components.

OPTION PROPOSED: Option 2

DECISION: Not Applicable

DECISION DATE:

ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.2  TOLERANCING SUBFACES

INITIATION DATE: 22nd Oct 86, Jan 88
INITIATOR: James Blaha, Bob Parks
STATUS: Unresolved
RELATED ISSUES: 86.13, 88.6
DESCRIPTION: Tolerances may apply only to a given subregion of a face rather than the whole face. Also, these subregions may be used as datums. Datum Targets (86.13) pose a similar problem. (Reference ANSI Y14.5M - 1982, pg. 45, section 4.4.8 for datums).
Drafting must be able to provide for the representation of such datum specifications, it must be able to obtain the extent of the surface being specified as a datum. (Figure 104, pg. 45).

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

OPTION PROPOSED:
DECISION: Not Applicable
DECISION DATE:
ACTION:
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.3 DATUM LABELS
INITIATION DATE: 15th December 1986
INITIATOR: ?
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Is the label attribute in datums simply a man-readable piece of text? If so, does it properly belong in this model? If label is a meaningless attribute, then does datum really have any meaning? That is not as strange as it sounds, because "datum" is not a thing itself, but rather a role played by a thing.

ISSUE OPTIONS & EVIDENCE:

Option 1: Delete the attribute Label from the Datum Entity.
   Pros:
   Cons:

Option 2: Delete both the attribute label and the entity Datum.
   Pros:
   Cons:

Option 3: Leave as-is

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.4  
UNCONDITIONED DATUM AND NOT APPLICABLE MLSN

INITIATION DATE: January 1988
INITIATOR: W.C. Burkett
STATUS: unresolved
RELATED ISSUES: N/A
DESCRIPTION: The use of a material condition modifier (MLS) in conditioned datums is applicable only when the underlying shape element is a feature of size. Otherwise, it is not applicable (N). This also includes the question of the default RES for an unspecified material condition call-out.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.5 AMBIGUOUS SIZE TOLERANCES
INITIATION DATE: 13th January 1988
INITIATOR: Mark Dunn
STATUS: Resolved
RELATED ISSUES: 87.2, 88.26, 88.13
DESCRIPTION: If a toleranced feature has more than one dimension subject to variation is size, how can multiple size tolerances be applied and interpreted unambiguously? This problem is particularly apparent in an implicit feature like a pocket, which has both a width and a length dimension. The size tolerance for each would be applied to the whole pocket, thereby causing the ambiguity.

ISSUE OPTIONS & EVIDENCE:

Option 1: Expand on the idea of a Size dimension that will distinguish between different size dimensions on a single feature and allow each to be tolerated separately. Size Dimension will be categorized as either a Size Parameter, which is referenced by Implicit Form Feature, and Size Characteristics (from which the value is calculated). See details in Tolerance Model Version 3.0.

Pros:
Cons:

OPTION PROPOSED: Option 1
EXPLANATION:
DECISION: The proposal for handling Size Dimensions as outlined in version 3.0 of the Tolerance Model solves the problem as outlined in the issue and meets with the approval of the Form Feature Committee.

DECISION DATE: 13th July 1988
ACTION: Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.6        GAGE GEOMETRY (POINTS, LINES, CIRCLES)

INITIATION DATE: 15th December 1986
INITIATOR: ?
STATUS: Unresolved
RELATED ISSUES: 86.4, 86.13, 88.1, 88.2
DESCRIPTION: Gage points are often used to dimension/tolerance slanted planes on prismatic parts, conical surfaces on turned parts, etc. The essence of a gage point is the use of a basic dimension to specify the "to" (occasionally the "from") location, which is then constrained by the tolerance.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.7    NUMBER OF ANGULARITY AND PERPENDICULARITY DATUMS

INITIATION DATE: 22nd December 1986, 22nd July 1987
INITIATOR: Mark Dunn, Ed Klages
STATUS: Unresolved
RELATED ISSUES: 88.9, 88.24, 88.25, 88.30, 88.46
DESCRIPTION: Angularity Tolerance and Perpendicularity Tolerance currently allow for up to two datums to be specified. The Y14.5 standards state that each of the tolerances are with respect to "a" datum.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.8  CONCENTRICITY CYLINDRICAL ZONE

INITIATION DATE: 9th January 1987
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: 88.10, 88.12
DESCRIPTION: The cylindrical zone attribute is superfluous. According to Y14.5 and Foster, the tolerance zone is necessarily cylindrical.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:

EXPLANATION:

DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3 SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.9 CIRCULAR RUNOUT, TOTAL RUNOUT DATUMS

INITIATION DATE: 9th January 1987, 22nd July 1987
INITIATOR: Mark Dunn, Ed Klages
STATUS: Unresolved
RELATED ISSUES: 88.7, 88.24, 88.25
DESCRIPTION: Per Y14.5 and Foster, these tolerances may have a secondary datum. The model only provides for one datum.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.10  CONCENTRICITY PROJECTED TOLERANCE ZONE
INITIATION DATE: 9th January 1988
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: 88.16, 88.25, 88.42
DESCRIPTION: Y14.5 says that projected tolerances zones may be used with location and orientation tolerances. The model is consistent with this one exception – there is no Projected attribute for the Concentricity tolerance. Should there be? Note that 14.5 does not explicitly say that any location or orientation tolerance type is susceptible to projected zones, nor did a brief search uncover an example of a concentricity tolerance with a projected zone.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N, A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.11  PROFILE OF A LINE, SURFACE TOLERANCED ENTITY
INITIATION DATE: 9th January 1988
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: 88.8, 88.10
DESCRIPTION: One application of these tolerances may apply to multiple faces or edges as a set. Hence the Toleranced Entity attribute needs to be a list of lists, rather than a simple list. More precisely, the Profile of a Line Toleranced Entity needs to be a list of coplanar edges or a list of faces and the Profile of a Surface Toleranced Entity needs to be a list of faces.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.12      TOLERANCE ZONE TYPES
INITIATION DATE: 9th January 1987
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The model makes no provision for spherical Tolerance zones. The Cylindrical zone attribute needs to be replaced by an attribute having one of three values: parallelepipedic, cylindrical, or spherical.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.13 TOLERANCES ON IMPLICIT FORM FEATURES

INITIATION DATE: 13th January 1988, 9th January 1987
INITIATOR: Mark Dunn
STATUS: Resolved
RELATED ISSUES: 88.5
DESCRIPTION: While implicit form features are shape defining components of shape model, they do not contain explicit pieces of geometry or other components to which a tolerance may be applied. If an implicit feature contains a single size parameter (such as the diameter of a hole), then the tolerance may be applied to the whole feature, with the implication that the size tolerance refers to the single size dimension. If the implicit feature contains more than one size dimension, an ambiguity arises.

ISSUE OPTIONS & EVIDENCE:

Option 1: Separate the notion of a size dimension from the definition of implicit features and allow size tolerances to be applied to these explicit size dimensions or to explicit features of size. See details of Tolerance Model Version 3.0.

Pros:
Cons:

OPTION PROPOSED: Option 1
EXPLANATION: This issue is essentially equivalent to 88.5. See the explanation in that issue for a full description of Size Dimension and Size Parameters.
DECISION: The categorization of Size Dimension into Size Parameter and Size Characteristic satisfies the requirements of this issue.
DECISION DATE: 13th July 1988
ACTION: Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.14 TOLERANCE QUALIFIERS
INITIATION DATE: 9th January 1987, January 1988
INITIATOR: Mark Dunn, Bob Parks
STATUS: Unresolved
RELATED ISSUES: 88.5
DESCRIPTION: There are a number of "qualifiers" that may be associated with tolerance data and which the tolerances model does not support. Examples are: ALL OVER, ALL AROUND, EACH ELEMENT, EACH RADIAL ELEMENT, FREE STATE, AVERAGE DIAMETER, FIM (FULL INDICATOR MOVEMENT). These are all apparently substantive data. In many cases, their message may be deducible from context, but this is not always possible or easy.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros: 
Cons: 
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.14A
CONSTRANTS AND MODEL COMPLEXITY

INITIATION DATE: 9th January 1987
INITIATOR: Mark Dunn
STATUS: Preliminary
RELATED ISSUES: N/A
DESCRIPTION: If representation were the only motivating consideration, the tolerance information model could be very terse. (About 10 or 15 entities would suffice.) The much larger number of entities in the model is a consequence of using the information model to constrain the data. With this in mind, and considering that only a fraction of “good practice” can be enforced in the information model, the following issue arises: should extensive constraining via the information model be attempted? Or would it be better to develop a simple, terse unconstrained model that focuses on representation, awaiting (and agitating for) appropriate constraining tools?

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION: a This issue has been resolved to a certain degree already. Extensions to the Express language enabled the model to be simplified by moving the constraints from the model (hence, deleting entities) to Express statements. This change is reflected between versions 2.0 and 2.1 of the model.

(2nd October 1987)

DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.15 TOLERANCES AND POLYHEDRIC MODELS
INITIATION DATE: 1st April 1987
INITIATOR: Ulrich Gengenbach
STATUS: Resolved
RELATED ISSUES: N/A
DESCRIPTION: The Tolerance Model has a very Boundary Representation feeling to it. How (or should) Tolerances be applied to Polyhedric representation models?

ISSUE OPTIONS & EVIDENCE:

Option 1: The application of tolerances to polyhedral models depends upon the intended use of the model. If the model is used primarily for efficient display of an object model and is not used for manufacturing applications such as NC and QA, then there is no need to apply tolerances to the model. If this is the case, then the model should be considered application specific, like an FEM mesh.

Pros: 
Cons:

Option 2: If there is still a need to apply tolerances to this type of model, then there are two problems which must be resolved. The first concerns the multitude of facets used to approximate a non-planar surface. Although a polyhedral model is basically amenable to the requirements of the Tolerance Model, tolerances are applied to single surfaces, and the many independent faces used in a polyhedral approximation prevent this. If the facets could be grouped to form a single thing, then a tolerance may be applied to it. The second problem is somewhat philosophical. When a tolerance is applied to a non-planar surface, it is an approximation of the desired surface which is tolerated. This poses a contradiction because a tolerance is usually applied to the theoretically exact surface definition. The toleranced surface on an actual produced part cannot be inspected from the polyhedral model because it does not contain the exact surface definition.

Pros: 
Cons:

OPTION PROPOSED: Option 1
EXPLANATION: Option 1 allows both models to remain unaffected by this issue, while placing the role of the Tolerance Model and Polyhedral Model into perspective. Choosing Option 2 would require some extensions to the polyhedral model.

DECISION: Option 1
DECISION DATE: 18th May 1987
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.16  PROFILE TOLERANCE APPLICATION
INITIATION DATE: 22nd July 1987, January 1988
INITIATOR: Ed Klages, Bob Parks
STATUS: Unresolved
RELATED ISSUES: 88.11
DESCRIPTION: In addition to specifying whether the tolerance application is to be inside, outside or bilateral, some provision must be made for an unequally disposed bilateral tolerance. See ANSI Y14.5 - 1982 6.5.1 (b).

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.17  DATUM CONSTRAINTS
INITIATION DATE: 14th October 1987
INITIATOR: Bob Parks
STATUS: Unresolved
RELATED ISSUES: 86.16
DESCRIPTION: Primary, Secondary and Tertiary datums must be exclusive of each other (primary and secondary cannot be the same, primary and tertiary must be different, etc.). Also, where multiple datums are specified for a particular geometric tolerance they should be sufficient to establish a datum reference frame of three mutually perpendicular planes.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.18 TOLERANCE EXISTANCE FOR DRAFTING
INITIATION DATE: 14th October 1987
INITIATOR: Bob Parks
STATUS: Preliminary
RELATED ISSUES: N/A
DESCRIPTION: If a product model is not complete (i.e. lacking a particular tolerance with a datum specification), how will drafting be allowed to deal with the product data (augmentation)?

ISSUE OPTIONS & EVIDENCE:

Option 1: The expansion of the "dimension" concept in the Tolerance Model Version 3.0 may resolve this concern.

Pros:
Cons:

Option 2:

OPTION PROPOSED: Option 1
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.19  TOLERANCE NAMES
INITIATION DATE: 23rd December 1987
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The names of the tolerance entities are not sufficiently descriptive in that the word “tolerance” is absent. This is particularly worrisome for entities LOCATION, ANGLE, SIZE, and POSITION, whose names suggest nominal shape concepts.

ISSUE OPTIONS & EVIDENCE:

Option 1: Add the word “tolerance” to the names of the tolerance entities so that, for example, entity 404 would be ANGLE TOLERANCE.

Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
### SECTION 3: SHAPE VARIATION TOLERANCES

<table>
<thead>
<tr>
<th>ISSUE</th>
<th>LOCATION TOLERANCE DIMENSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIATION DATE</td>
<td>23rd December 1987</td>
</tr>
<tr>
<td>INITIATOR</td>
<td>Mark Dunn</td>
</tr>
<tr>
<td>STATUS</td>
<td>Preliminary</td>
</tr>
<tr>
<td>RELATED ISSUES</td>
<td>86.11, 87.2, 88.26</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The information in and associated with the LOCATION/403 entity is not adequate to describe the dimension being tolerated. (The SIZE/405 and probably ANGLE/404 suffer from the same problem.) The original and basic idea of the model is that location tolerances apply to the dimensions (nominal distances) between pairs of &quot;physical&quot; topological/geometric entities. However, this is sufficient only for the simple cases where there is a constant distance between the origin and target (toleranced entity), e.g. parallel planes, concentric cylinders. There are many cases that are not so simple either because the origin or the target is non-physical or because the Brep-ish model contains no pair of parallel topological/geometric entities to serve as the origin and target. The following data were included in the LOCATION/403 entity to provide the specificity needed for non-simple cases: Origin.Loc, a point nominally on the origin of the toleranced dimension; Tol Ent loc, a point nominally on the target of the toleranced dimension; Path, the nominal direction of measurement of the toleranced dimension. It is felt that even with this information, the model is unsatisfactory. On any particular piece, the points will be inside the material or in the air and the path (direction) will deviate from the nominal. The origin and target of the dimension of interest, as well as the direction of measurement, can only be determined from physical measurement and a knowledge of the intended dimension. Thus, the specification of the intended measurement must be explicit in or derivable from the modeled information. <strong>NOTE:</strong> This problem description has been superceded and rendered obsolete by the work of the Tolerance Committee as reported in the report dated 25th March 1988 and Version 3.0 of the Tolerance Model. The concept of dimension has been expanded upon, hopefully rectifying the problem spelled out above. The details of the Technical Report have been incorporated into a new version of the Tolerance Model, version 3.0.</td>
</tr>
</tbody>
</table>

**ISSUE OPTIONS & EVIDENCE:**

Option 1: Expand on the "dimension" concept for Location, Size and Angle tolerances. See details in the Tolerance Model Version 3.0.

**Pros:**

**Cons:**

Option 2:
OPTION PROPOSED: Option 1

EXPLANATION: This issue is no longer valid because the new approach for determining the dimensional value for a coordinate dimension requires that the tolerance origin and tolerance target be parallel to one another.

DECISION: This issue is considered resolved by the committee.

DECISION DATE: 13th July 1988

ACTION: Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.21 TABULATED TOLERANCES
INITIATION DATE: January 1988
INITIATOR: Bob Parks
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Tabulated Tolerances for a family of parts cannot be accommodated in the model.

ISSUE OPTIONS & EVIDENCE:

Option 1: Make the tolerance attribute of the Geometric Tolerance entity a selection of type number or tolerance-id, where the tolerance-id corresponds to a table of values.

Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

**ISSUE:** TOL-88.22

**INITIATION DATE:** January 1988

**INITIATOR:** Bob Parks

**STATUS:** Unresolved

**RELATED ISSUES:** 86.6

**DESCRIPTION:** Since temporary datums may be established for machining operations that create permanent datums, should this distinction be accommodated in the model? (reference ANSI Y14.5M 1982, section 4.2.1, page 36.)

**ISSUE OPTIONS & EVIDENCE:**

Option 1:
- Pros:
- Cons:

Option 2:

**OPTION PROPOSED:**

**EXPLANATION:**

**DECISION:** N/A

**DECISION DATE:**

**ACTION:**
## SECTION 3: SHAPE VARIATION TOLERANCES

**ISSUE: TOL-88.23  UNIDIRECTIONAL AND BI-DIRECTIONAL DIMENSIONS**

**INITIATION DATE:** 15th February 1988  
**INITIATOR:** Mark Dunn  
**STATUS:** Preliminary  
**RELATED ISSUES:** 88.29

**DESCRIPTION:** It is necessary to distinguish between bi-directional (undirected, “between”) location dimensions and unidirectional (directed, “from/to”).

### ISSUE OPTIONS & EVIDENCE:

Option 1: Add a flag to the Location Dimension to distinguish between directed and bi-directional dimensions. See Coordinate Dimensions in Version 3.0 of the Tolerance Model.

**Pros:**

- **Cons:**

Option 2:

**OPTION PROPOSED:** Option 1

**EXPLANATION:** The addition of a simple flag, as reflected in version 3.0 of the Tolerance Model resolves this problem. The same issue, however, applies to the Related Angle Dimension (but is recorded as a separate issue).

**DECISION:** This issue is resolved with the noted change.

**DECISION DATE:** 13th July 1988

**ACTION:** Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.24  CO-DATUMS
INITIATION DATE: 15th February 1988
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: 88.7, 88.9, 88.25, 88.46
DESCRIPTION: The model only provides for co-datums for circular runout, concentricity, and total runout. It is possible that co-datums may be specified elsewhere.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:

DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.25  PROFILE OF A LINE - NUMBER OF DATUMS
INITIATION DATE: 15th February 1988
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: 88.7, 88.9, 88.24, 88.11, 88.16
DESCRIPTION: Is a tertiary datum possible for a profile of a line tolerance? It seems that a third datum may be illogical in a 2-D tolerance.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.26  DEFINITION OF "DIMENSION"
INITIATION DATE: 15th February 1988
INITIATOR: Mark Dunn
STATUS: Preliminary
RELATED ISSUES: 86.1, 87.2, 86.11, 88.20
DESCRIPTION: There are two concepts in the term "dimension": a measurement on a product and the value of that measurement. The latter can be determined (nominally) from a model if the measurement is known. The tolerance model must provide for describing the measurement.

ISSUE OPTIONS & EVIDENCE:

Option 1: The components which make up a dimension should be separated and a clear explanation of what a "dimension" is should become part of the model. A possible approach for doing this was reported in a technical report dated 25th March 1988 and in version 3.0 of the Tolerance Model (8th July 1988).

Pros:
Cons:

Option 2:

OPTION PROPOSED: Option 1
EXPLANATION: The expansion of the idea of "dimension" within the Tolerance Model will eliminate much of the ambiguity in the relationship of "dimensions" to the tolerances.
DECISION: The definition of dimension within version 3.0 of the Tolerance Model resolves this issue.
DECISION DATE: 13th July 1988
ACTION: Tolerance Committee
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.27 MINIMUM AND MAXIMUM DIMENSION VALUE

INITIATION DATE: 15th February 1988
INITIATOR: Kim Perlotto
STATUS: Unresolved
RELATED ISSUES: 88.36
DESCRIPTION: Sometimes a minimum value is given, with no maximum and vice-versa. The model can stretched to cover maximum, but not minimums.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.28 DIAMETER VS. RADIUS SIZE TOLERANCE
INITIATION DATE: 13th July 1988
INITIATOR: Mark Dunn
STATUS: Preliminary
RELATED ISSUES: 88.5
DESCRIPTION: A Size Tolerance may be applied to a diametrical or a radial dimension, but there is no way to indicate this distinction. It is important to make this distinction because the tolerance zone for a radial tolerance is twice the size of a tolerance zone for a diameter tolerance of equal value.

ISSUE OPTIONS & EVIDENCE:

Option 1: The addition of a simple flag within the Size Dimension will indicate whether the tolerance applies to the radial value or diameter value.

Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION: N/A
DECISION DATE:
ACTION:
ISSUE: TOL-88.29 UNI-DIRECTIONAL vs BI-DIRECTIONAL ANGLE DIMENSION

INITIATION DATE: 13th July 1988
INITIATOR: Mark Dunn
STATUS: Preliminary
RELATED ISSUES: 88.23
DESCRIPTION: As described in 88.23, a coordinate dimension may uni-directional (i.e. measured in a specified direction) or may be bi-directional (i.e. direction of measurement unimportant.) The issue described in 88.23 as applied to Location dimensions also applies to Related Angle Dimensions and may be similarly resolved.

ISSUE OPTIONS & EVIDENCE:

Option 1: Include a binary flag within the Related Angle Dimension that indicated whether the measurement is directed or undirected.

Pros: Consistent with Location Dimension.
Cons:

Option 2:

OPTION PROPOSED: Option 1
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.30  NUMBER OF DATUMS FOR ANGULARITY, PARALLELISM, PERPENDICULARITY

INITIATION DATE: 13th July 1988
INITIATOR: John Yanney (GD-FW), JNC-Hashimoto
STATUS: Unresolved
RELATED ISSUES: 88.7, 88.46
DESCRIPTION: Angularity, Parallelism, and Perpendicularity tolerances do not allow for the specification of three datums. Three datums may be specified to form a Datum Reference Frame for each tolerance.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:

260
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.31  RELATIONSHIP BETWEEN COORDINATE DIMENSION AND COORDINATE TOLERANCE

INITIATION DATE: 13th July 1988
INITIATOR: Mark Dunn
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The relationship between the coordinate tolerance entities and the dimension entities are backwards. The dimensional entity should be the parent, the tolerance entity the dependent child (via a "Z" relationship). The dimension is the information of first instance without which the tolerance is meaningless; the tolerance is subordinate.

ISSUE OPTIONS & EVIDENCE:

Option 1: Flip the relationship causing the Dimension to be the Parent and Tolerance the child in the relationship.

Pros:
Cons:

Option 2: Leave as is. with Tolerance the Parent and Dimension the Child.

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-88 32 D & T SHAPE ELEMENT AND FORM FEATURE CATEGORIZATION

INITIATION DATE: 13th July 1988
INITIATOR: Mark Dunn
STATUS: Preliminary
RELATED ISSUES: N/A
DESCRIPTION: The work of the Integration Committee has resulted in some specified relationships between shape elements, their representations and Form Features. The Tolerance Model, however, takes some liberties in the inclusion of these concepts. In order to keep the work in sync, the model should be changed to correspond to the Integration Work. This is basically an IDEF1X modelling problem.

ISSUE OPTIONS & EVIDENCE:

Option 1: Change “Shape Element” within the Tolerance Model to “DT Shape Element”, categorize it according the needs of the Tolerance Model and establish relationships between the categories and the corresponding Shape Element from the Integration Model. Also change Form Feature to DT Form Feature and relate it to Form.

Pros:
Cons:

Option 2:

OPTION PROPOSED: Option 1. See Version 3.1 of the Tolerance Model for details (Entity 300, DT.Feature).

EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.33 SYMMETRY TOLERANCE
INITIATION DATE: 13th July, 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Both the ISO and ANSI standards specify a geometric tolerance called Symmetry. This tolerance is not included in the Tolerance Model, but should be.

ISSUE OPTIONS & EVIDENCE:

Option 1: Include a geometric Symmetry Tolerance
Pros: Will correspond with the current specification.
Cons: Was under consideration for deletion from Standard (at least the ANSI version of the standard). The current status of this question is unknown.

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.34  
ARRANGEMENT OF ENTITY DEFINITIONS

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The Tolerance Entities are currently arranged alphabetically within the model. They should be rearranged categorically according to the standards (Tolerances of Form, Orientation, Runout, Profile.)

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.35  START POINT FOR PROJECTED TOLERANCE ZONE

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Projected Tolerance Zone is currently defined in terms of direction and extent (length). A start point for the zone should also be included as part of the definition.

ISSUE OPTIONS & EVIDENCE:

Option 1: Include a start point with the PTZ definition
   Pros: Would more clearly define the projected zone.
   Cons: Because a tolerance can be applied to more than one target at a time (e.g. a Position Tolerance can be applied to a set of Holes.) the start point for a Projected Tolerance Zone for a single tolerance may not be unique (i.e. there may be many start points, one for each target (e.g. hole), each with the same direction and extent.)

Option 2: Include a set of start points within the PTZ definition.

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3 SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.36 COORDINATE TOLERANCE RANGE APPLICATIONS

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: 88.27
DESCRIPTION: The Coordinate Tolerance Range (plus/minus) for the value of a measurement currently restricted to a positive or zero value for the Plus. Tolerance, a positive or zero value for the Minus. Tolerance, and Plus. Tol + Minus. Tol not equal to zero. There are a number of additional cases, examples of which are in the ISO and ANSI Standards, that are not accommodated by this approach. These include:
1) both the upper bound and lower bound of the range have the same sign (e.g. +0.010, +0.005; -0.002, -0.06);
2) limit dimensioning (i.e. specification of maximum/minimum values or a single maximum or minimum value for the dimension);
3) the use of ISO symbols for tolerances (e.g. 0 12H7/h6).

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.37  MATERIAL CONDITION MODIFIER FOR PROFILE OF LINE/SURFACE

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The Profile of a Line and Profile of a Surface Tolerance use Conditioned Datums, yet do not also have a Material Condition Modifier (MCM) attribute. Both tolerances should have a MCM as an attribute.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3. SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.38 LOCATION DIMENSION VALUE

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: A nominal dimension value is necessary to define the Location Dimension. A value is included for both the Size and Angle Dimensions, and should be included in Location.

ISSUE OPTIONS & EVIDENCE:

Option 1: Add an attribute for the Nominal Location Dimension Value
Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.39 TARGET OF POSITION, CONCENTRICITY, CIRCULAR RUNOUT

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Position, Concentricity and Circular Runout tolerances can apply equally to Features of Size, Area and Seams. Currently, each tolerance can apply to only a subset of these things.

ISSUE OPTIONS & EVIDENCE:

Option 1: Change the definition of each entity to allow it reference each type of target: Feature of Size, Area and Seam.

Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.40 PER UNIT LENGTH FOR PARALLELISM, PERPENDICULARITY, ANGULARITY

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: A Per Unit Length specification is valid not only with the context of a Straightness Tolerance, but also within Parallelism, Perpendicularity and Angularity Tolerances. An attribute should be added to each entity to account for this.

ISSUE OPTIONS & EVIDENCE:

Option 1: Add an attribute “Per Unit Length” to the Parallelism, Perpendicularity, and Angularity tolerances.

Pros:
Cons:

Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.41 MATERIAL CONDITION MODIFIER FOR CONCENTRICITY, CIRCULARITY, CYLINDRICITY, CIRCULAR AND TOTAL RUNOUT TOLERANCES.

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: The Material Condition Modifier can be specified for Concentricity, Circularity, Cylindricity, Circular and Total Runout Tolerances. These entities should be modified to reflect this fact.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.42 PROJECT TOLERANCE ZONE FOR CONCENTRICITY

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: 88.10

DESCRIPTION: The purpose of the Projected Tolerance Zone is to increase the size of the tolerance zone defined by a tolerance beyond the limits of the feature being tolerated. Given its applicability to Position, Angularity, Parallelism, and Perpendicularity, it would also seem applicable to Concentricity.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.43

APPLICATION
DIRECTION FOR PARALLELISM, PERPENDICULARITY, POSITION, CIRCULAR RUNOUT, AND TOTAL RUNOUT

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Straightness Tolerance can be applied to a surface in a specified direction (e.g. on a cylinder, surface elements parallel to the axis can be tolerated with a straightness tolerance.) In a similar fashion, a direction may be used to specify the application of Parallelism, Perpendicularity, Position, Circular Runout and Total Runout.

ISSUE OPTIONS & EVIDENCE:

Option 1: Include a direction specification in the identified entities.
Pros: 
Cons: 

Option 2: 

OPTION PROPOSED: 
EXPLANATION: 
DECISION: 
DECISION DATE: 
ACTION: 

273
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.44  DEFAULT TOLERANCES FOR OTHER TOLERANCES.

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: 86.8
DESCRIPTION: A default tolerance is currently only allowed for Profile of a Surface Tolerance. All tolerance entities should be allowed to be specified as a default.

ISSUE OPTIONS & EVIDENCE:
Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.45  APPLICATION OF TOTAL AND CIRCULAR RUNOUT TOLERANCE

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: N/A
DESCRIPTION: Application of the tolerance is currently specified in Profile Line Tolerance and Profile Surface Tolerance. Isn’t it necessary that Application should also be specified in Circular Runout and Total Runout Tolerance?

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
ANNEX D
(Draft Proposal)

SECTION 3: SHAPE VARIATION TOLERANCES

ISSUE: TOL-88.46 SPECIFICATION OF DATUMS FOR PERPENDICULARITY, PARALLELISM AND ANGULAR TOLERANCE.

INITIATION DATE: 13th July 1988
INITIATOR: JNC - Dr. Hashimoto
STATUS: Unresolved
RELATED ISSUES: 88.7, 88.24, 88.30
DESCRIPTION: With respect to Datums, the Angularity, Parallelism and Perpendicularity tolerances should be modified as follows:
Descriptions in Version 3.0

- In Parallelism tolerance (412), Primary Datum can be specified.

- In Perpendicularity Tolerance (413), Primary and Secondary Datums can be specified.

- In Angularity (406), Primary, Secondary and Tertiary Datums can be specified.

Amended version:

- In Parallelism Tolerance and in Perpendicularity Tolerance, Primary Datum, Co-primary Datum, Secondary Datum, and Co-secondary datum can be specified.

- In Angularity Tolerance, Primary Datum and Co-primary datum can be specified.

ISSUE OPTIONS & EVIDENCE:

Option 1:
Pros:
Cons:
Option 2:

OPTION PROPOSED:
EXPLANATION:
DECISION:
DECISION DATE:
ACTION:
3.7 Change History

Model changes reflected in this section outline the differences between the PDES Tolerance Model Version (unnumbered) dated 26th September 1986 and Version 1.0 dated 9th January 1987.

1. Tolerances are often to/from off-part geometry which is derived from product geometry (e.g. centerlines). An Entity called DERIVED GEOMETRY was added to the model.

2. The Angle Tolerance was considered confusing and imprecise. Changes were made remedy this problem.

3. References to BASIC dimensions have been removed from the model. The geometric, product model itself is considered BASIC.

4. Default tolerances will be handled by adding a flag to the Profile of a Surface tolerance to indicate a default condition.

5. The relationship between Size Feature, Feature.of.Size, and Form Feature has been clarified.

6. Material Condition Modifiers (MMC, LMC, RFS) apply only to "features" subject to variation in size (i.e. features.of.size). Changes to reflect this have been made.

7. There was some confusion over the intent and meaning of the coordinates associated with the Location Tolerance. Some minor changes have been made to reduce the probability of problems, but further work is necessary.

Model changes reflected in the following section outline the differences between the PDES Tolerance Model Version 1.0 dated 9th January 1987 and Version 2.0 dated 27th March 1987. These changes were prompted by the release of the PDES Initial Testing Draft and bring this model into correspondence with the Testing Draft. There have been no changes in technical content, but rather a conversion to bring this model in line with the released Testing Draft. All entities and definitions within the model are entities which appear in the Testing Draft.

1. The explanation on Model Content in the Assumptions section has been modified to reflect the release of the Testing Draft.

2. The Entity Pool section has been modified so that the alphabetic and numeric listing of the entities each fall entirely on a single page.

3. Geometry(100) was renamed Wire.Frame.Geometry(100) and the diagram changed to reflect the Testing Draft.

4. Unit.Vector(101) was renamed Direction(101) to bring it in line with the Testing Draft.

5. Point.Vector(102) was renamed Projected.Tolerance.Zone and changed to reflect the contents of the Geometry model. The concept of Vector and Point Vector did not exist in the Geometry Model, so rather than create them, an entity was defined (Projected.Tolerance.Zone) that contained the same functionality and had a clearly defined usage and intent.

277
SECTION 3: SHAPE VARIATION TOLERANCES

6. Vector(103) was deleted as unnecessary (see item 5).

7. Subtypes of Curve(106) were deleted as superfluous.

8. Coordinate(107) was renamed Cartesian.Point.

9. Topology(200) was renamed Shape.Brep(200).

10. Entities 204-214 were named Face or Edge with a geometric adjective modifying the meaning were deleted. The purpose of these entities were to constraint the types of entities to which a certain tolerance may apply. These constraints have been incorporated into an Express WHERE clause in the entity’s definition. This made these entities unnecessary.

11. References to Form Features have been removed since they are not pertinent to the definition of the Tolerance Model.

12. Tolerance(400) was renamed Shape.Tolerance(400).

The changes to the model outlined in the following section represent the differences between Version 2.0, 27th March 1987, and Version 2.1, 2nd October 1987. These changes were prompted by an extension to the EXPRESS language (in particular the SELECT construct) which allowed the model to be simplified. The extensions to EXPRESS allowed contraints previously represented as unique entities to be included in the EXPRESS definition of other, more relevant entities. The overall result is that all but one of the 500-series entities have been removed from the model. The technical content of the two versions should be equivalent.

1. The notation "< -" and "- >" used in the propositions for each entity has been removed. The intent had been to indicate the direction of the migration of key attributes according to IDEF-IX. The differences between the EXPRESS definition on an entity and the the IDEF-IX model have blurred this distinction to the point where it is meaningless. The SELECT capability of EXPRESS and the categorized associative entity required by IDEF-IX (the 500-series entities) appear to behave different with regard to key migration.

2. All 500-series entities have been removed from the model except for entity 521. The constraints represented by those entities are now captured using the EXPRESS SELECT option. The entities still exist, but are now included as support entities on the subject entity diagram rather than uniquely identified.

3. Entity 521 Angle.Intersection has been renamed and reconstructed as Point.Triplet. Instead of the complicated structure which existed previously, the precise angle definitions are now defined by the Point.Triplets with constraints that restrict the relationship between the points, the tolerated entity and the tolerance origin. The Angle Tolerance now references a list of Point.Triplets and a single tolerated entity and tolerance origin.

4. The introduction of the SELECT construct and elimination of the 500 series associative entities has simplified the subtyping of the tolerance entities. Rather than being a subtype of XXX.Tol.Ent, Face is now just a subtype of Shape.BREP.
SECTION 3: SHAPE VARIATION TOLERANCES

The changes outlined in the following section reflect the differences between version 2.1 of the Tolerance Model dated 2nd October 1987 and version 3.0 dated 8th July 1988. These changes were prompted by a meeting with the ad hoc Form Features committee in January 1988 and were developed during an interim meeting of the Tolerance Committee 15th – 18th February 1988. The results of this interim meeting are detailed in a technical report dated 25th March 1988.

1. The idea of “dimensions” was developed within the model. The information necessary to determine the dimension was unsatisfactorily buried with the Coordinate Tolerance entities. In the version of the model, the emphasis is flip-flopped. The Tolerance range takes on a lesser role and Coordinate Dimensions are defined. The Coordinate Tolerances Location, Size and Angle have been replaced with Location Dimension, Size Dimension and Angle Dimension. The entity Coordinate Tolerance has been replaced with Coordinate Tolerance Range. The information necessary to define the dimension is made explicitly, so implicit mechanisms such as Point Triplet and Location-Tolerance-Location point are no longer needed.

2. The role of Geometric Derivation has been expanded upon. Originally just a “stub” in this model, the work on Form Features has prompted an expansion of this idea. It is really an extension to the model, rather than a change to existing work.

3. The notion of Dimension/Tolerance Feature has been introduced to sort out the entities to which tolerances are applied “to” and “from”. This is very similar to a structure that was part of an earlier version of the Tolerance Model. It organizes Shape Elements, Features of Size, Size Features, and Form Features.

4. Projected Tolerance Zone was modified by eliminating the position for the zone. If the same tolerance applies to several entities (such as a set of holes), then a single position for the tolerance zone would be meaningless.

5. The Tolerance Value for the Geometric Tolerances was moved from the Geometric Tolerance entity into each of the subtypes because of the constraint on that value in the Profile of a Surface tolerance entity.

6. The entities Shape.BREP, Face, Edge and Vertex were renamed Shape.Element, Area, Seam and Corner in accordance with the work of the Integration Committee.

The following changes reflect the differences between version 3.0, dated 8th July 1988 and version 3.1, dated 8th August 1988. These changes were prompted by the meetings of the PDES Integration Committee and consist primarily of editorially changes.

1. The IDEF model contained in this document was “formalized” from its earlier, free-form version. This means that there are a whole bunch of new entity numbers (although there are the same number of boxes). The EXPRESS definitions did not change, nor did 99changed to reflect the fact that they were defined in other models.

2. There were only two minor technical changes introduced with this version. In order to be more “integrated” with the form features model, the attribute Parameter.Value within Size.Parameter and Angle Size.Parameter were made optional. This was to allow derivable
attributes of Implicit Form Features (like the diagonal of a rectangular pocket) to be tol-
eranced without explicitly calling out a value.

3. The other technical change were to allow Concentricity and Circular Runout Tolerances to
be applied to Seams.

4. There was also one semi-significant technical change. This involved the categorization of DT
Feature into Form Feature and Shape Elements. Since the work of the Integration Committee
explicitly defined how these concepts were related, the liberties taken including them into the
Tolerance Model resulted in a model which did not correspond to the Integration Model. For
instance, in version 3.0 of the Tolerance Model Shape Element was categorized into Area,
Seam and Corner, whereas in the Integration Model the same entity was more extensively
categorized; this apparent discrepancy was seen as unacceptable. So within the Tolerance
Model, Shape Element became DT Shape Element, and Form Feature became DT Form
Feature, essentially introducing an addition level between the entities in the Tolerance Model
and the entities in the Integration Model.

The following changes detail the differences between Version 3.1 of the Tolerance Model, dated
8th August 1988, and Version 3.2, dated 26th August 1988. These changes were prompted, once
again, by the work of the Integration Committee and were made to accommodate the publication
of Volume 3.0 of the PDES document. With a single technical exeception, there is absolutely N
difference between Version 3.1 of the model and Version 3.2. With the single caveat explained here,
3.1 may be considered the latest version of the model.

1. A single technical change was made to accommodate the needs of the Form Features committee.
A discussion at the Denver meeting questioned whether a tolerance applied to a radius was
the same as that applied to a diameter. There was an obvious difference of a factor of 2.
To take with distinction into account, a boolean valued attribute called Half dimension was
added to Size Dimension and to Angle Dimension. TRUE would mean that tolerance applies
to the radius, FALSE that it applies to the full diameter.

2. The PDES/STEP document section number was changed from 3.1.1.6 to 3.1.1.2. This is
reflected on every page of this document.