ISO TC 184/SC4/WG11 N???

Date: 2004-07-27

Supersedes

ISO 10303-29

Industrial automation systems and integration — Product data representation and exchange — Part 29: Implementation methods: XML encoding of the exchange structure of product data defined by an Application Protocol

ABSTRACT: This document specifies an exchange format that allows product data to be transferred from one computer system to another using XML.

KEYWORDS: automation, automation engineering, computer applications, industrial products, data, data representation, data exchange, coding (data conversion), XML, implementation.

COMMENTS TO READER:
This document describes an XML format for Application Protocol data exchange. A more general format for EXPRESS defined data exchange is described in ISO 10303-28. The format described in this document is related to the 10303-28 format by a configuration given in 10303-28.

Project Leader:	Dr. Martin Hardwick	**Project Editor:**	Dr. Martin Hardwick
Address:	STEP Tools, Inc. 14 First Street Troy, NY 12180-3810 USA	**Address:**	As shown.
Telephone:	+1 (518) 687-2848 x306	**Telephone:**	
Telefacsimile:	+1 (518) 687-4420	**Telefacsimile:**	
Electronic mail:	hardwick@steptools.com	**Electronic mail:**	

ISO copyright office
Case Postale 56 - CH-2111 Geneve 20 - Switzerland
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

page

Tables page

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO 10303 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 10303-29 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data*.

This International Standard is organized as a series of parts, each published separately. The structure of this International Standard is described in ISO 10303-1.

Each part of this International Standard is a member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the implementation methods series.

A complete list of parts of ISO 10303 is available from the Internet:

`<http://www.nist.gov/sc4/editing/step/titles/>`

Annexes A, B, C, D, E and F form a normative part of this part of ISO 10303. Annexes G and H are for information only.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for archiving.

This part of ISO 10303 specifies a mechanism that allows product data described by an Application Protocol to be transferred from one computer system to another using XML.

Major subdivisions in this part of ISO 10303 are:

— specification of the exchange structure for data conforming to an Application Protocol;

— specification of the exchange structure for data conforming to Application Objects in an Application Protocol;

— mapping from an EXPRESS schema onto XML.

NOTE The examples of EXPRESS usage in this part of ISO 10303 do not conform to any particular style rules. Indeed, the examples sometimes use poor style to conserve space or to concentrate on the important points. The examples are not intended to reflect the content of the information models defined in other parts of this International Standard. They are crafted to show particular features of EXPRESS or of the exchange structure. Many examples are annotated in a way that is not consistent with the syntax rules of this part of ISO 10303. These annotations are introduced by symbolic arrows, either horizontal '---->', or vertical. These annotations should be ignored when considering the parse rules. Any similarity between the examples and the normative models specified in other parts of this International Standard should be ignored. Several mapping examples have been provided throughout this document. Additional *spaces* and new lines have been inserted into some of these examples to aid readability. These *spaces* and new lines need not appear in an exchange structure.

This part will be related to the more general mapping of EXPRESS to XML Schema given in ISO 10303-28 by a configuration description that will be included in ISO 10303-28.

Industrial automation systems and integration — Product data representation and exchange — Part 29: Implementation methods: XML encoding of the exchange structure of product data defined by an Application Protocol

1 Scope

This part of ISO 10303 specifies an exchange structure format using a XML encoding of product data defined by an Application Protocol. The exchange format is suitable for the transfer of data conforming to a complete Application Protocol, and for the transfer of data conforming to one or more Application Objects in an Application Protocol.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 8824-1:1998, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO 10303-1:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: The EXPRESS language reference manual.*

Uniform Resource Identifiers (URI): Generic Syntax. Internet Engineering Task Force RFC 2396 August 1998 [cited 2000-08-07]. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2396.txt>.

URN Syntax. Internet Engineering Task Force RFC 2141 May 1997 [cited 2000-09-28]. Available from World Wide Web: <http://www.ietf.org/rfc/rfc2141.txt>.

Extensible Markup Language (XML) 1.0. World Wide Web Consortium Recommendation 10 February 1998 [cited 2000-04-26]. Available from World Wide Web: <http://www.w3.org/TR/1998/REC-xml-19980210>.

Namespaces in XML. World Wide Web Consortium Recommendation 14 January 1999 [cited 2000-04-26]. Available from World Wide Web: <http://www.w3.org/TR/1998/REC-xml-19990114>.

XML Information Set World Wide Web Consortium Candidate Recommendation 24 October 2001 [cited 2003-05-12]. Available from World Wide Web: <http://www.w3.org/TR/xml-infoset>.

XML Linking Language (XLink) Version 1.0. World Wide Web Consortium Candidate Recommendation 3 July 2000 [cited 2000-08-08]. Available from World Wide Web: <http://www.w3.org/TR/2000/CR-xlink-20000703>.

XML Path Language (Xpath) Version 1.0. World Wide Web Consortium Recommendation 16 November 1999 [cited 2003-05-12]. Available from World Wide Web: <http://www.w3.org/TR/xpath/>.

XML Pointer Language (Xpointer) Version 1.0. World Wide Web Consortium Recommendation 8 January 2001 [cited 2004-01-19]. Available from World Wide Web: <http://www.w3.org/TR/WD-xptr.html/>.

XML Schema Part 1: Structures. W3C Recommendation, 2 May 2001 [cited 2002-08-01]. Available from World Wide Web: < http://www.w3.org/TR/xmlschema-1/>.

XML Schema Part 2: Datatypes. W3C Recommendation, 2 May 2001 [cited 2002-08-01]. Available from World Wide Web: < http://www.w3.org/TR/xmlschema-2/>.

3 Terms, definitions, and abbreviations

3.1 Terms defined in ISO 10303-1

This part of ISO 10303 makes use of the following terms defined in ISO 10303-1.

— application protocol;

— application object;

— application interpreted model;

— application requirements model;

— exchange structure;

— mapping path.

3.2 Terms defined in ISO 10303-11

This part of ISO 10303 makes use of the following terms defined in ISO 10303-11.

— complex entity instance;

— data type;

— entity;

— partial complex entity instance;

— simple entity instance.

3.3 Terms defined in XML Standards

Terms defined in the XML Standards and used for the purposes of this part of ISO 10303 are repeated below for convenience.

— XML element;

— XML attribute;

— XML document.

3.4 Other definitions

For the purposes of this part of ISO 10303, the following definitions apply.

3.4.1
XML encoding
the encoding of information using XML.

3.4.2
Most specific defined type
the outermost type in a sequence of nested type definitions.

3.5 Abbreviations

For the purpose of this part of ISO 10303, the following abbreviations apply.

URI	Uniform Resource Identifier
URN	Uniform Resource Name

XML	Extensible Markup Language
AIM	Application Interpreted Model
ARM	Application Requirements Model

4 Exchange structure fundamental concepts and assumptions

4.1 Introduction

The exchange structure is described by an XML syntax. All of the data in the exchange structure shall belong to the name spaces defined in the header element 7.2.5. Any data not in these name spaces shall be ignored.

NOTE Users that want to mix STEP and non-STEP data can put the non-STEP data into a different namespace.

4.2 Notational and typographical conventions

Any *quotation marks* used in this part of ISO 10303 are not part of the text that appears in the exchange structure but serve to delimit that text. This statement applies to all places in the text where *quotation marks* are used.

Within examples in this part of ISO 10303, an annotation is introduced by the sequence ----> where clarification is required.

4.3 Conformance

Three levels of conformance are specified in the six conformance classes listed in clause 11:

— syntactical conformance of the exchange structure: an exchange structure conforms to ISO 10303-29 if it is well formed XML and the requirements of this part of ISO 10303 are satisfied;

— application protocol conformance of the exchange structure: the instances represented in the exchange structure represent a complete instance of the AIM schema of the Application Protocol listed in the header element of the exchange structure if every requirement or constraint of that schema is satisfied and the requirements defined in clauses 8, 9 and 11 of this part of ISO 10303 are satisfied.

— application object conformance of the exchange structure: the instances represented in the exchange structure conform to the requirements defined by the mapping tables for the application objects of the Application Protocol listed in the header element of the exchange structure and the requirements defined in clauses 8, 10 and 11 of this part of ISO 10303 are satisfied.

An implementation that claims syntactical conformance to this part of ISO 10303 shall read or write documents or both that exhibit syntactical conformance.

An implementation that claims application protocol conformance to this part of ISO 10303 shall read or write documents or both that exhibit application protocol as well as syntactical conformance.

An implementation that claims application object conformance to this part of ISO 10303 shall read or write documents or both that exhibit application object as well as syntactical conformance.

5 Formal definitions

5.1 Exchange structure

The exchange structure shall shall consist of a header element and one or more data elements. The header provides data relating to the exchange structure itself. The structure of the header element is specified in clause 7. The data elements provide the data to be transferred. The structure of the data elements is specified in clause 8. The header element and data elements are embedded in larger XML documents.

The element "<ISO10303-29>" shall contain the Header of an exchange structure.

The element "<AIM>" shall contain a subset of the population of the AIM data of an exchange structure.

The element "<ARM>" shall contain a subset of the population of the ARM data of an exchange structure.

NOTE The header and data elements can be embedded in SOAP envelopes and other XML infrastructure.

6 Basic Tokens

In the exchange structure, a basic token is a keyword or a simple data type.

6.1 Keywords

Keywords are tokens indicating an application object, attribute, defined type, entity or mapping path in the exchange structure.

6.1.1 Entity and Defined Type Keywords

Entity and Defined Type keywords shall be defined by the application interpreted model of the Application Protocol and consist of one *upper case letter*, followed by *lower case letters*, *digits* and *low lines*.

6.1.2 Attribute Keywords

Attribute keywords shall be defined by the application interpreted model of the Application Protocol and consist of one *lower case letter*, followed by *lower case letters*, *digits* and *low lines*.

6.1.3 Application Object Keywords

Application Object keywords shall be defined by the mapping table of the Application Protocol and consist of one *upper case letter* followed by *upper case letters*, *digits* and *low lines*.

6.1.4 Application Object Attribute Keywords

Application Object Attribute keywords shall be defined by the mapping table of the Application Protocol and shall consist of one *lower case letter* followed by *lower case letters*, *digits* and *low lines*.

6.2 Simple data type encodings

Eight simple data type encodings are used in exchange structures: integer, real, string, entity instance identifier, restricted entity instance identifier, entity instance reference, enumeration, and binary.

6.2.1 Integer

An integer shall be encoded as prescribed in XML.

EXAMPLE

Valid integer expressions	Meaning
16	Positive 16
+12	Positive 12
-349	Negative 349
012	Positive 12
00	Zero
Invalid integer expressions	**Problem**
26 54	Contains *spaces*
32.0	Contains *full stop*
+ 12	Contains *space* between *plus sign* and digits

6.2.2 Real

A real shall be encoded as prescribed by XML.

NOTE No attempt is made to convey the concept of precision in this part of ISO 10303. Where a precise meaning is necessary, the sender and receiver of the exchange structure should agree on one. Where a precise

meaning is required as part of the description of an entity data type, this meaning should be included in the entity data type definition in the EXPRESS schema.

EXAMPLE

Valid real expressions	Meaning
+0.0E0	0.0
-0.0E-0	0.0, as above example
1.5	1.5
-32.178E+02	-3217.8
0.25E8	25 million
0.E25	0.0
2.	2.0
5.0	5.0

Invalid real expressions	Problem
1.2E3.	Decimal point not allowed in exponent
1E05	Decimal point required in mantissa
1,000.00	*Comma* not allowed
3.E	Digit(s) required in exponent
.5	At least one digit must precede the decimal point

6.2.3 String

A string shall be encoded as prescribed by XML.

EXAMPLE

Valid string expressions	Meaning
Abc	Abc
Abc dEf	Abc dEf
+0.0E0	+0.0E0

Invalid real expressions	Problem

6.2.3.1 Maximum string length

The maximum length of a string is as prescribed by XML.

6.2.4 Entity Instance identifiers

A entity instance identifier shall be encoded as a upper letter, lower letter or lowline, followed by a sequence of upper letters, lower letters, digits, periods, low lines and hyphens.

EXAMPLE

Valid identifier expressions	Meaning
id_12	Names entity instance with identifier id_12
inches	Names entity with identifier inches
English-inches	Names entity with identifier English-inches
B00.1	Names entity with identifier B00.1

Invalid identifier expressions	Problem
A+23	Contains '+' sign
74	Begins with a digit
#inches	Begins with a *number sign*

Entity instance identifers are used to reference the instance with the identifier. Both forward and backward references are permitted.

6.2.5 .Restricted Entity Instance identifiers

A restricted entity instance identifier is an entity instance identifer that begins with the string "id-" followed by one or more digits.

EXAMPLE

Valid restricted identifier expressions	Meaning
id-12	Names entity instance with identifier id-12
id-90	Names entity with identifier id-90

Invalid restricted identifier expressions	Problem
inches	Identifier does not begin with the string "id-"
id-inches	The "id-" is not followed by a sequence of digits.

Restricted entity instance identifiers are required by Conformance Classes 1 and 2 (Clause 11.)

6.2.6 Entity Instance references

A entity instance reference shall be encoded as a URL which includes a fragment identifier. If no document is given in the URL then the entity instance shall exist in the same document as the referencing

instance. The fragment identifier shall identify the entity instance in the document using its entity instnace identifier.

EXAMPLE

Valid reference expressions	Meaning
http://www.abc.com/file.xml#id_12	References entity instance with identifier id_12 stored in the file "file.xml" at the web site www.abc.com
another_file.stp#id-10	References entity instance with restricted identifier id-10 stored in the file "another_file.stp"
#English-inches	References entity with identifier English-inches in the same file as the referencing instance.

Invalid reference expressions	Problem

6.2.7 Enumeration values

An enumeration value shall be encoded as an upper case string describing the value of the enumeration. The meaning of a given enumeration value is determined by the EXPRESS schema and its associated definitions from the enumeration type declarations.

EXAMPLE

TYPE Material = ENUMERATION OF (steel); END_TYPE;

Valid enumeration expression	Meaning
STEEL	Indicates a value of STEEL

Invalid enumeration expressions	Problem
.STEEL	Beginning *full stop*
Steel	Wrong case
IRON	Not defined in the EXPRESS.

6.2.8 Binary

A binary is a sequence of bits (0 or 1). A binary shall be encoded as determined by the following procedure.

— count the number of bits in the sequence. Call the result p;

— determine a number n, $0 \leq n \leq 3$, such that k=p+n is a multiple of four;

— left fill the binary with n zero bits. Divide the sequence into groups of four bits.

— precede the sequence with the 4-bit representation of n;

— if the decimal equivalent of a 4-bit group is 9 or less, add 48 to that decimal value to create an 8-bit byte; if the decimal equivalent of the 4-bit group is greater than 9, add 55 to that decimal value to create an 8-bit byte.

NOTE This is a binary to hexadecimal conversion.

— the encoding of a binary consists of k/4+1 hexadecimal digits. The first digit is the value of n. This is followed by the hexadecimal digits representing the binary;

EXAMPLE

Binary value	Representation
'empty'	0
0	30
1	31
111011	23B
100100101010	092A

7 Header element

The header element contains information that is applicable to the entire exchange structure. This section shall be present in every exchange structure. If a document contains more than one header element then each header element shall have an id attribute that uniquely identifies the header element.

NOTE Annex H presents an example of a header section.

7.1 Header section entities

The header section of every exchange structure shall contain one instance of each of the following entities: **exchange_description**, **exchange_name**, **exchange_schema, exchange_population** and **exchange_space**.

7.2 Header section schema

This clause specifies entities and types that appear in the header element of the exchange structure. The header entities are specified in EXPRESS.

EXPRESS Specification:

*)


```
SCHEMA header_section_schema;

(*
```

This schema specifies the header entities that are specific to the process of transferring product data using the exchange structure.

7.2.1 exchange_description

The **exchange_description** specifies the version of this part of ISO 10303 used to create the exchange structure as well as its contents.

EXPRESS Specification:

```
*)
ENTITY exchange_description;
  description          : LIST [1:?] OF STRING;
  implementation_level : STRING;
END_ENTITY;
(*
```

Attribute Descriptions:

description: an informal description of the contents of this exchange structure.

implementation_level: an identification of the specification to which the encoding in this exchange structure conforms and any conformance options employed in that encoding. The form of the value is "v;cc", where v is the version number of this part of ISO 10303, as specified in annex C, and cc is the encoding of conformance class as specified in Section 11.

7.2.2 exchange_name

The **exchange_name** provides human readable information about the exchange structure. With the exception of the time_stamp attribute, the contents of the attributes of this entity are not defined by this part of ISO 10303.

EXPRESS Specification:

```
*)
ENTITY exchange_name;
  name                 : STRING;
  time_stamp           : time_stamp_text ;
  author               : LIST [ 1 : ? ] OF STRING;
  organization         : LIST [ 1 : ? ] OF STRING;
  preprocessor_version : STRING;
  originating_system   : STRING;
  authorization        : STRING;
END_ENTITY;

TYPE time_stamp_text = STRING;
```



```
END_TYPE;
(*
```

Attribute Descriptions:

name: the string of graphic characters used to name this particular instance of an exchange structure.

NOTE The name is intended to be used as human to human communication between sender and receiver.

time_stamp: the date and time specifying when the exchange structure was created. The contents of the string shall correspond to the extended format for the complete calendar date as specified in 4.2.1.1 of ISO 8601 concatenated to the extended format for the time of the day as specified either in 4.3.1.1 or in 4.3.3 of ISO 8601. The date and time shall be separated by the *capital letter T* as specified in 4.4.1 of ISO 8601. The alternate formats of 4.3.1.1 and 4.3.3 permit the optional inclusion of a time zone specifier.

author: the name and mailing address of the person responsible for creating the exchange structure.

organization: the group or organization with whom the author is associated.

preprocessor_version: the system used to create the exchange structure, including the system product name and version.

originating_system: the system from which the data in this exchange structure originated.

authorization: the name and mailing address of the person who authorized the sending of the exchange structure.

EXAMPLE

Time stamp element	Complete extended format
Calendar Date 12 April 1993	1993-04-12
Time of the Day 27 minutes 46 seconds past 15 hours	15:27:46
Time Zone 5 hours west of Greenwich	Time Zone field is optional -05:00
Above date and time encoded within the Time_Stamp field	1993-04-12T15:27:46-05:00

7.2.3 exchange_schema

The **exchange_schema** entity identifies the Application Protocol that defines the data sections using the name of the EXPRESS schema that defines the Application Interpreted Model of the protocol. The attribute **schema_identifier** shall consist of a string that shall contain the name of the schema optionally followed by the object identifier assigned to that schema.

If the name of a schema contains *small letters*, such *small letters* shall be converted to the corresponding *capital letters*. Only *capital letters* shall occur in strings of the **schema_name**.

If an object identifier is provided, it shall have the form specified in ISO/IEC 8824-1. The use of object identifiers within this International Standard is described in clause 3 of ISO 10303-1. When available, the use of the object identifier is recommended as it provides unambiguous identification of the schema.

NOTE The general form of an object identifier is a sequence of space-delimited integers. The sequence is enclosed within *braces* ("{", "}").

EXPRESS Specification:

```
*)
ENTITY exchange_schema;
  schema_identifier : schema_name;
END_ENTITY;

TYPE schema_name = STRING;
END_TYPE;
(*
```

Attribute Descriptions:

schema_identifier: the schema name of the Application Interpeted model of the Application Protocol that defines the data elements.

EXAMPLE The instance below identifies an EXPRESS schema called 'CONFIG_CONTROL_DESIGN':

```
EXCHANGE_SCHEMA (('CONFIG_CONTROL_DESIGN'));
```

The instance below uses an object identifier to indicate a specific version of an EXPRESS schema called 'AUTOMOTIVE_DESIGN':

```
EXCHANGE_SCHEMA (('AUTOMOTIVE_DESIGN { 1 0 10303 214 1 1 1 }'));
```

7.2.4 exchange_population

The **exchange_population** entity identifies a collection of documents containing the data elements of the exchange structure, and a collection of documents containing other entity instances referenced by those data elements. An entity instance reference in an exchange structure shall reference entity instances in these documents only.

EXPRESS Specification:

```
*)
ENTITY exchange_population;
    included_exchange_locations : SET [ 1 : ? ] OF url;
    resource_exchange_locations : SET [ 1 : ? ] OF url;
END_ENTITY;

TYPE document_url = STRING;
END_TYPE;
(*
```

Attribute Descriptions:

included_exchange_locations: the URL's for XML documents containing data elements that are to be included in the exchange structure. If the URL has a fragment identifier then only the data element referenced by the identifier is included. If no fragment identifier is given then every data element in the document shall be included.

resource_exchange_locations: the URL's for XML documents and ISO 10303-21 files containing entity instances that may be included in the exchange structure (see clause 8). If the URL addresses an XML document and has a fragment identifier then the only the XML element referenced by the id shall be included. If the URL addresses a ISO 10303-21 file then the fragment identifier shall be a restricted entity instance identifier (see 6.2.5) and only the entity instance addressed by the identifier shall be included.

NOTE 1 If the external reference "example.xml#id-10" is used to describe an entity instance reference then "example.xml" must be one of the documents in the **exchange_population** entry.

NOTE 2 The **exchange_population** entry allows a pre-processor to find and cache all of the documents and ISO 10303-21 files containing data that may be used in an exhange structure at the beginning of a read process.

NOTE 3) Resource documents can be used to defined libraries of entity instances for reuse across multiple application protocols. Application Protocol developers may wish to consider establishing a library for all of the constants used in a protocol so that exchange structures can reference these constants using a URL defined by the Application Protocol standard.

NOTE 3) Resource files can be used to share geometry data between ISO 10303-203 processors that use the Part 21 format and new processors that use the Part 29 format.

7.2.5 exchange_space

The **exchange_space** entity identifies the names spaces of the data in the exchange structure. If no name spaces are given then the exchange structure shall be limited to XML elements in the exchange population that belong to the default name space. If one or more name spaces are given then the exchange structure shall be limited to XML elements in the exchange population that belong to the given name spaces.

EXPRESS Specification:

```
*)
ENTITY exchange_space;
  urn_identifiers : LIST [0:?] OF urn_name;
END_ENTITY;

TYPE urn_name = STRING;
END_TYPE;
(*
```

Attribute Descriptions:

urn_name: the XML urn definition of a name space.

EXAMPLE The instance below identifies a name space for STEP data:

```
<exchange_space
    <urn_identifiers>
        <Urn_name>urn:iso10303:stp/www.tc184-sc4/STEP</Urn_name>
    <urn_identifiers>
<exchange_space>
```

8 Exchange Structure Population

8.1 Included Documents

Two types of data elements may be included in the exchange structure:

- An AIM element contains entity instances defined by the AIM EXPRESS model of the Application Protocol.

- An ARM element contains mapping path instances describing how application objects defined by the Application Protocol are represented by entity instances in the AIM elements of the exchange structure.

A data element may be referenced by multiple URL's in the **exchange_population** entry but it shall be included in the exchange structure once only.

If a URL in the **exchange_population** entry includes a fragment identifier then only the data element identified by that identifier shall be included in the exchange structure.

NOTE 1 The Header element can be stored in the same document as its Data elements or in different documents. Some applications may choose to modularize the data by storing related ARM and AIM elements in the same exchange. Other applications may choose to include all their data in one AIM element only.

NOTE 2 If two URL's in the **exchange_population** entry reference the same document and one URL includes a fragment identifier and the other URL does not include a fragment identifier then every data element in the document shall be included in the exchange structure.

NOTE 3 An ARM element does not define any entity instances. An ARM element defines data that allows an exchange processor to verify that an Application Object instance has been represented correctly within the exchange structure without checking the whole exchange structure.

8.1.1 AIM Elements

An AIM element shall be used to encode entity instances belonging to the exchange structure. Each entity instance shall be mapped as specified in Section 9. Each entity instance shall be represented at most once in the exchange structure. The entity instances need not be ordered in the exchange structure. An entity instance may be referenced before it is defined.

EXAMPLE

```
<AIM>
    .
    .
    .
<Pt id="id-14">
    <x>1.0</x>
    <y>2.0</y>
    <z>3.0</z>
</Pt>
    .
    .
    .
    .
</AIM>
```

8.1.2 ARM Elements

An ARM element shall be used to store mapping paths defining how instances of application objects are encoded in the exchange structure. Each application object instance shall be mapped as specified in 10.

EXAMPLE

```
<ARM>
    .
    .
    .
<POINT>
    <xcoordinate>
        <Pt href="example.xml#id-14"/>
        <x/>
    </xcoordinate>
    <ycoordinate>
        <Pt href="example.xml#id-14"/>
        <y/>
    </ycoordinate>
</POINT>
```



```
        .
        .
        .
</ARM>;
```

NOTE 1 Application Objects do not have instance names.

NOTE 1 The representation of Application Objects allows processes to verify that they are encoded correctly in an exchange structure or a fragment of an exchange structure. Therefore, libraries of correctly encoded Application Objects can be stored in databases for reuse across multiple applications.

8.2 Resource elements

The resource documents and files define a library of entity instances that are to be included in the exchange structure only if they are referenced by another entity instance or application object instance already in that structure.

A resource document shall be encoded as well formed XML.

A resource file shall be encoded as ISO 10303-21.

If the resource is an XML document then the document shall be searched for an XML element with an Id attribute whose value is the entity instance identifier stored in the referencing entity. The entity instance shall be encoded using the rules described in Section 9. If the resource is identified by a URL with a fragment identifier then only the element identified by that fragment shall be searched. Elements in the resource document that are not identified by this procedure may use any encoding allowed by well formed XML.

If the resource is an ISO 10303-21 file then the referencing entity must contain a restricted entity instance identifier and the ISO 10303-21 file shall be searched for the entity instance with that identifier. The entity instance shall be encoded using the rules described in ISO 10303-21 and must conform to the EXPRESS schema identified by the header element. If the resource is identified by a URL with a fragement identifier then only the entity instance identified by that fragment shall be searched. Entity instances in the resource file that are not identified by this procedure may be defined by any EXPRESS schema.

8.3 Exchange Structure Population

8.3.1 Entity Instance Population

The entity instance population of an exchange structure shall include all of the entity instances in all of the AIM elements in the exchange structure, and all of the entity instances in the resource elements referenced by the AIM elements or ARM elements either directly or transitively.

8.3.2 Application Object Population

The application object instance population of an exchange structure shall include all of the application objects in all of the ARM elements in the exchange structure.

9 Mapping of Entity Instances to the exchange structure

This clause describes how entity instances described by the Application Interpreted Model are mapped to the exchange structure.

The Application Interpreted Model of an Application Protocol is described by an EXPRESS schema. This clause describes how entity instances are mapped by describing how each feature of the EXPRESS language is mapped to the exchange structure

The EXPRESS language includes TYPE and ENTITY declarations, CONSTANT declarations, constraint specifications and algorithm descriptions. Only instances of data types, as defined by EXPRESS data types and TYPE and ENTITY declarations, are mapped to the exchange structure. Other elements of the language are not mapped to the exchange structure.

NOTE ISO 10303-28 describes a mapping of the EXPRESS language to XMLSchema. The relationship between this schema and the mapping described in this clause is described by a configuration of ISO 10303-28.

9.1 Mapping of EXPRESS data types

This clause specifies the mapping from the EXPRESS elements that are data types to the exchange structure.

9.1.1 Mapping of EXPRESS simple data types

9.1.1.1 Integer

Values of the EXPRESS data type INTEGER shall be mapped to the exchange structure as an integer data type. 6.2.1 describes the composition of an integer data type..

9.1.1.2 String

Values of the EXPRESS data type STRING shall be mapped to the exchange structure as a string data type. 6.2.3 describes the composition of a string data type.

9.1.1.3 Boolean

Values of the EXPRESS data type BOOLEAN shall be mapped to the exchange structure as an enumeration data type. 6.2.7 describes the composition of a enumeration data type. The EXPRESS data type BOOLEAN shall be treated as a predefined enumerated data type with a value encoded as the strings "true" or "false". These values shall correspond to true and false respectively.

9.1.1.4 Logical

Values of the EXPRESS data type LOGICAL shall be mapped to the exchange structure as an enumeration data type. 6.2.7 describes the composition of a enumeration data type. The EXPRESS data type LOGICAL shall be treated as a predefined enumerated data type with a value encoded as the strings "true", "false" or "unknown". These values shall correspond to true, false, and unknown respectively.

Table 5 - Quick reference mapping table

EXPRESS element	mapped onto:
ARRAY	aggregate
BAG	aggregate
BOOLEAN	boolean
BINARY	binary
CONSTANT	NO INSTANTIATION
DERIVED ATTRIBUTE	NO INSTANTIATION
ENTITY	entity instance
ENTITY AS ATTRIBUTE	entity value
ENUMERATION	enumeration
FUNCTION	NO INSTANTIATION
INTEGER	integer
INVERSE	NO INSTANTIATION
LIST	aggregate
LOGICAL	enumeration
NUMBER	real
PROCEDURE	NO INSTANTIATION
REAL	real
REMARKS	NO INSTANTIATION
RULE	NO INSTANTIATION
SCHEMA	NO INSTANTIATION
SELECT	See 11.1.8
SET	aggregate
STRING	string
TYPE	See 11.1.6
UNIQUE rule	NO INSTANTIATION
WHERE RULES	NO INSTANTIATION

9.1.1.5 Real

Values of the EXPRESS data type REAL shall be mapped to the exchange structure as a real data type. 6.2.2 describes the composition of a real data type.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY widget;
  i1: INTEGER;   ----------> A
  i2: INTEGER;   ----------> B
  s1: STRING(3); ----------> C
  s2: STRING;    ----------> D
  l : LOGICAL;   ----------> E
  b : BOOLEAN;   ----------> F
  r1: REAL(4);   ----------> G
  r2: REAL;      ----------> H
END_ENTITY;
```

Sample instance in the data section:

```
<Widget id="id-30">
    <i1>99</i1>
    <i2>99999</i2>
```



```
    <s1>ABC</s1>
    <s2>ABC EFG</s2>
    <l>unknown</l>
    <b>false</b>
    <r1>9.</r1>
    <r2>1.2345</r2>
</Widget>
```

A: i1 has a value of 99 in this entity instance.

B: i2 has a value of 99999 in this entity instance.

C: s1 has a value of 'ABC' in this entity instance. This value falls within the range (3 characters) specified for this attribute.

D: s2 has a value of 'ABC EFG' in this entity instance.

E: l has a value of unknown in this entity instance.

F: b has a value of false in this entity instance.

G: r1 has the value of 9. in this entity instance. The precision specification does not affect the encoding.

H: r2 has a value of 1.2345 in this entity instance.

NOTE The example in this section describes the XML elements for each attribute in the order defined in the EXPRESS example. However, no attribute order is defined in the EXPRESS language or this part of ISO 10303.

9.1.1.6 Binary

Values of the EXPRESS data type BINARY shall be mapped to the exchange structure as a binary data type. 6.2.8 describes the composition of a binary data type.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY picture;
  bn :  BINARY;
END_ENTITY;
```

Sample entity instance in data section:

```
<Picture id="example">
    <bn>1556FB0</bn>
</Picture>
```

bn has an encoding of "1556FB0" in this instance, corresponding with the bit sequence 101 0101 0110 1111 1011 0000.

9.1.1.7 Number

Values of the EXPRESS data type NUMBER shall be mapped to the exchange structure as a real data type. 6.2.2 describes the composition of a real data type.

9.1.2 Aggregates

Values of type LIST, BAG, SET and ARRAY shall be mapped as aggregates. The member of the aggregate shall be mapped as follows:

- If the member of the aggregate is an entity instance then is shall be mapped as described in 9.2

- If the member of the aggregate is simple data type then it shall be mapped as an XML element with the name of the keyword of the type described in 6.1 and the XML content described in 9.1.1

- If the member is a simple defined type, enumerated type or select type then it shall be mapped as an XML element with the name of the keyword of the type described in 6.1 and the XML content described in 9.1.3, 9.1.4, 9.1.5 respectively.

- If the member of the aggregate is another aggregate and no defined type has been defined in the EXPRESS to represent this aggregate then the member shall be encoded as an element called <Aggregate>.

9.1.2.1 Unset Values in aggregates

If an aggregate contains an unset item then the XML element in the list that represents the unset item shall have an XML attribute called unset whose value shall be set to "true".

NOTE EXPRESS only allows array aggregates to contain unset items. However, an exchange structure that contains unset items in other kinds of aggregates can be in syntactic conformance with this part of ISO 10303

9.1.2.2 List

Values of the EXPRESS data type LIST shall be mapped to the exchange structure as an aggregate. If the LIST is empty, the list shall be encoded as an XML element with no content. Within the list, each instance of the element type shall be encoded as specified in clause 9 for that EXPRESS data type.

NOTE If, in a particular entity instance, no value is provided for an OPTIONAL attribute whose data type is a LIST, then the attribute is encoded either by setting the unset attribute of the XML element representing the list to "true", or by not including any XML element for the OPTIONAL attribute in the exchange structure.

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:  LIST [1 : ?] OF LIST [0 : ?] OF REAL; ----> D
  attribute5:  OPTIONAL LIST OF STRING; ------------------> E
END_ENTITY;
```

Sample entity instance in data section:

```
<Widget id="id-10">
    <attribute1/>
    <attribute2>
        <Integer>1</Integer>
        <Integer>2</Integer>
        <Integer>4</Integer>
    </attribute2>
    <attribute4>
        <aggregate>
            <Real>1.0</Real>
            <Real>2.56</Real>
        </aggregate>
        <aggregate/>
        <aggregate>
            <Real>6.4</Real>
        </aggregate>
    </attribute4>
    <attribute5 unset="true"/>
<Widget>
```

A: attribute1 is an empty list (list with zero elements).

B: attribute2 contains three elements in this instance.

C: attribute3 does not have a value in this instance.

D: attribute4 contains three nested lists in this instance. The first nested list contains two elements. The second nested list contains zero elements. The third nested list contains one element.

E: attribute5 does not have a value in this instance.

NOTE The XML example in this section are describing the XML elements for each attribute in the order defined in the EXPRESS example. However, no attribute order is defined in the EXPRESS language or this part of ISO 10303.

9.1.2.3 Array

Values of the EXPRESS data type ARRAY shall be mapped to the exchange structure as aggregates. If an EXPRESS attribute is a multidimensional array the attribute shall be encoded as a list of lists, nested as deeply as there are dimensions. In constructing such lists, the inner-most list, the list containing only instances of the element type, shall correspond to the right-most ARRAY specifier in the EXPRESS statement defining the entity. The ordering of the elements within the encoding shall be that all the elements of the inner-most list are encoded for each element of the next outer list. This order means that the right-most index in each list shall vary first. Within the list, each instance of the element type shall be encoded as specified in clause 9 for that EXPRESS data type.

NOTE If, in a particular array item no value is provided then the absence of the value can be shown using the unset attribute of the XML element representing the item.

EXAMPLE 1 Entity definition in EXPRESS:

```
X : ARRAY[1:5] OF ARRAY[100:102] OF INTEGER
```

This is encoded in the following order:

```
( (X [1,100], X [1,101], X [1,102] ),
  (X [2,100], X [2,101], X [2,102] ),
  (X [3,100], X [3,101], X [3,102] ),
  (X [4,100], X [4,101], X [4,102] ),
  (X [5,100], X [5,101], X [5,102] )  )
```

EXAMPLE 2 Entity definition in EXPRESS:

```
ENTITY widget;
  attribute1:  ARRAY [-1 : 3] OF INTEGER;     ----------------------> A
  attribute2:  ARRAY [1 : 5] OF OPTIONAL INTEGER;     --------------> B
  attribute3:  ARRAY [1 : 3] OF OPTIONAL ARRAY [1 : 3] OF INTEGER;     ------> C
END_ENTITY;
```

Sample entity instance in data section:

```
<Widget id="id-30">
     <attribute1>
          <Integer>1</Integer>
          <Integer>2</Integer>
          <Integer>3</Integer>
          <Integer>4</Integer>
          <Integer>5</Integer>
     </attribute1>
     <attribute2>
          <Integer>1</Integer>
          <Integer>2</Integer>
          <Integer unset="true"/>
          <Integer/>
          <Integer>5</Integer>
     </attribute2>
     <attribute3>
          <Aggregate>
               <Integer>1</Integer>
               <Integer>2</Integer>
               <Integer>3</Integer>
          </Aggregate>
          <Aggregate unset="true"/>
          <Aggregate>
               <Integer>4</Integer>
               <Integer>5</Integer>
               <Integer>6</Integer>
          </Aggregate>
     </attribute3>
</Widget>
```

A: attribute1 contains the following values:


```
attribute1 [-1] = 1
attribute1 [0]  = 2
attribute1 [1]  = 3
attribute1 [2]  = 4
attribute1 [3]  = 5
```

B: attribute2 contains the following values:

```
attribute2 [1] = 1
attribute2 [2] = 2
attribute2 [3] = not provided
attribute2 [4] = not provided
attribute2 [5] = 5
```

The significance of a missing value is defined within the EXPRESS schema.

C: attribute3 contains the following values:

```
attribute3 [1,1] = 1
attribute3 [1,2] = 2
attribute3 [1,3] = 3
attribute3 [2] = not provided
attribute3 [3,1] = 4
attribute3 [3,2] = 5
attribute3 [3,3] = 6
```

EXAMPLE 3Entity definition in EXPRESS:

```
ENTITY widget;
  attribute1:  ARRAY [1 : 3] OF OPTIONAL ARRAY [1 : 3] OF OPTIONAL STRING;     ------> C
END_ENTITY;
```

Sample entity instance in data section:

```
<Widget id="id-30">
    <attribute1>
        <Aggregate>                         <!-- A -->
            <String>John</String>
            <String unset="true"/>
            <String/>
        </Aggregate>
        <Aggregate unset="true"/>           <!-- B -->
        <Aggregate/>                        <!-- C -->
    </attribute1>
</Widget>
```

A: In the first row the second item has its "unset" XML attribute set to "true" so its value is unset, and the third item is set to the empty string.

B: The second row is unset which is legal because the nested array is OPTIONAL.

C: The third row is set to an empty content. Syntactically this is correct. However, the row is incorrect with respect to the EXPRESS because the array bounds are [1:3] so the third row should either be unset or it should contain another instance of the underlying aggregate with three items (not zero) defined.

9.1.2.4 Set

Values of the EXPRESS data type SET shall be mapped to the exchange structure as aggregates. Within the aggregate, each instance of the element type shall be encoded as specified in clause 9 for that EXPRESS data type. If the SET is empty, the list shall be encoded as an XML element with no content.

NOTE If, in a particular entity instance, no value is provided for an OPTIONAL attribute whose data type is a SET, the attribute is encoded either by setting the unset attribute of the XML element representing the list to "true", or by not including any XML element for the OPTIONAL attribute in the exchange structure.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY widget;
  a_number: SET OF INTEGER;
END_ENTITY;
```

Sample entity instance in data section:

```
<Widget id="A">
    <a_number>
        <Integer>0</Integer><Integer>1</Integer><Integer>2</Integer>
    </a_number>
</Widget>
<Widget id="B">
    <a_number>
        <Integer>0</Integer><Integer/><Integer>2</Integer>
    </a_number>
</Widget>
<Widget id="C">
    <a_number>
        <Integer>0</Integer><Integer>0<Integer/><Integer>2</Integer>
    </a_number>
</Widget>
```

A: The attribute a_number was defined by the set numbers 0, 1, 2 in this instance.

B: Syntactically the instance is correct. However, the instance is incorrect with respect to the definition of a SET in EXPRESS because a SET is not allowed to have missing members.

C: Syntactically the instance is correct. However, the instance is incorrect with respect to the definition of a SET in EXPRESS because a SET is not allowed to have duplicate members.

9.1.2.5 Bag

Values of the EXPRESS data type BAG shall be mapped to the exchange structure as aggregates. Within the aggregate, each instance of the element type shall be encoded as specified in clause 9 for that EXPRESS data type. If the BAG is empty, the list shall be encoded as an XML element with no content.

NOTE If, in a particular entity instance, no value is provided for an OPTIONAL attribute whose data type is a BAG, the attribute is encoded by setting the unset attribute of the XML element representing the list to "true", or by not including any XML element for the OPTIONAL attribute in the exchange structure.

EXAMPLEEntity definition in EXPRESS:

```
ENTITY widget;
  a_numbers: BAG OF INTEGER;
END_ENTITY;
```

Sample entity instance in data section:

```
<Widget id="A">
     <a_numbers>
          <Integer>0</Integer><Integer>1</Integer><Integer>1</Integer><Integer>2</Integer>
     </a_numbers>
</Widget>
<Widget id="B">
     <a_number>
          <Integer>0</Integer><Integer/><Integer>2</Integer>
     </a_number>
</Widget>
```

A: The attribute a_numbers was defined by the collection of numbers 0, 1, 1, 2 in this instance.

B: Syntactically, the instance is correct. However, the instance is incorrect with respect to the definition of BAG in EXPRESS because a BAG is not allowed to have missing members.

9.1.3 Simple defined types

A simple defined type is a type defined by an EXPRESS type declaration in which the underlying type is neither an ENUMERATION nor a SELECT. If the defined type is used in an aggregate or SELECT then an XML element shall be generated for each instance of the type using the keyword of the type.

EXAMPLE Entity definition in EXPRESS:

```
TYPE
  type1 =  INTEGER;
END_TYPE;

TYPE
  type2 =  type1;
END_TYPE;

TYPE
  type3 = LIST [1 : 2] of type2;
END_TYPE;

ENTITY widget;
  a1:  TYPE1;
  a2:  LIST OF TYPE1;
  a3:  TYPE3;
END_ENTITY;
```

Sample entity instance in data section:


```
<Widget id="4">
    <a1>10</a1>
    <a2><Type1>20</Type1><Type1>30</Type1></a2>
    <a3><Type2>40</Type2></a3>
</Widget>
```

a1: No element is necessary for the type.

a2: Type1 is the outermost type for the members of the list so it is used to name the XML element for each item.

a3: Type2 the outermost type for the members of the list so it is used to name the XML element for each item.

9.1.4 Enumeration types

Values of an EXPRESS ENUMERATION data type shall be mapped to the exchange structure as an enumeration data type. 6.2.7 describes the composition of a enumeration data type.

EXAMPLE Entity definition in EXPRESS:

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

ENTITY widget;
  p_colour: primary_colour;     ----------> A
END_ENTITY;
```

Sample entity Instance in data section:

```
<Widget id="A">
    <p_colour>RED</p_colour>
</Widget>
```

A: The value of the attribute p_colour in this entity instance is RED.

9.1.5 Select data types

An EXPRESS select data type defines a list of data types, called the "select-list", whose values are valid instances of the select data type. An instance of a select data type shall be a value of at least one of the data types in the select-list. The value shall be encoded in the exchange structure as determined by the following procedure:

— if the value is an instance of an entity data type in the select-list, it shall be mapped to the exchange structure as an entity instance reference element (see 9.1.6);

— if the value is an instance of a simple defined type in the select-list, it shall be mapped to the exchange structure as specified in 9.1.3;

— if the value is an instance of an enumeration data type in the select-list, it shall be mapped to the exchange structure as as specified in 9.1.3;

— if the value is an instance of a (nested) select data type in the select-list, it shall be mapped to the exchange structure as an instance of that select type as provided in this clause;

— if the value is an instance of a (nested) select data type in the select-list, and the result of the mapping is ambiguous because the branch of the select used to determine the underlying value is meanignful and cannot be determined from the type of the result then the outermost type of the select-branch used for this value shall be addded to an XML attribute of the element called path. If there are multiple names in the path they shall appear in the order outemost to innermost.

NOTE 1 If the data type (in the select-list) which the value instantiates is itself a select data type, then this clause will be used recursively to encode the value. Only instances of entity data types, simple defined types and enumeration data types can actually be encoded (see Example 2).

NOTE 2 According to ISO 10303-11:1994, an instance of a subtype of an entity data type is an instance of the entity data type. So "an instance of an entity data type in the select list" includes instances of subtypes of those entity data types.

NOTE 3 If the entity data types in the select-list are not mutually exclusive, then a value of the select data type may instantiate more than one entity data type in the select-list (see Example 1).

NOTE 4 If the value is not an entity instance, it is an instance of exactly one simple defined-type or enumeration data type. The value may, however, be a valid instance of several (nested) select data types and thereby instantiate more than one type in the original select-list (see Example 2).

EXAMPLE 1 Entity definition in EXPRESS:

```
ENTITY Leader SUBTYPE OF (Employee);
  project: STRING;
END_ENTITY;

ENTITY Manager SUBTYPE OF (Employee);
  unit: STRING;
END_ENTITY;

ENTITY Employee;
  name: STRING;
END_ENTITY;

TYPE Supervisor = SELECT (Manager, Leader);
END_TYPE;


ENTITY Meeting;
  date:       STRING;
  attendees:   SET [2:?] OF Supervisor;
END_ENTITY;
```

Sample data section instances:

```
<Meeting id="id-4">
     <date>14921012</date>
     <attendees>
```



```
        <Leader id="id-1">
            <name>J. Brahms</name>
            <project>Academic Festival</project>
        </Leader>
        <Manager id="id-2">
            <name>S. Ozawa</name>
            <unit>Tokyo Symphony</unit>
        </Manager>
        <Leader-Manager id="id-3">
            <name>G. Verdi</name>
            <project>Aida</project>
            <unit>La Scala</unit>
        </Leader-Manager>
    <attendees>
</Meeting>
```

The second attribute of instance id-4 is the attendees: a SET OF Supervisor. Instance id-1 is a Leader and thus a valid Supervisor. Instance id-2 is a Manager and thus a valid Supervisor. Instance id-3 is a complex instance of both (entity) types Leader and Manager from the select-list of Supervisor. All are mapped according to 9.1.6.

EXAMPLE 2 Entity definition in EXPRESS:

```
TYPE Mass = SELECT (Mass_Spec, Mass_Substitute);
END_TYPE;

TYPE Mass_Spec = SELECT (Measured_Mass, Computed_Mass, Estimated_Mass);
END_TYPE;

TYPE Measured_Mass = REAL;
END_TYPE;

TYPE Computed_Mass = Extended_Real;
END_TYPE;

TYPE Estimated_Mass = REAL;
END_TYPE;

TYPE Mass_Substitute = SELECT(Weight, Estimated_Mass);
END_TYPE;

TYPE Weight = REAL;
END_TYPE;

TYPE Extended_Real = SELECT (FloatingNumber, NotaNumber);
END_TYPE;

TYPE FloatingNumber = REAL(7);
END_TYPE;

TYPE NotaNumber = ENUMERATION OF (plus_infinity,
  minus_infinity, indeterminate, invalid);
END_TYPE;

ENTITY Steel_Bar;
  bar_length: Extended_Real;
  bar_mass:   Mass;
END_ENTITY;
```

Sample instantiation in data section:


```
<Steel_bar id="id-1" >
   <bar_length><Floatingnumber>77.0</Floatingnumber></bar_length>
   <bar_mass><Measured_mass>13.25</Measured_mass></bar_mass>
</Steel_bar>

<Steel_bar id="id-2" >
   <bar_mass><Estimated_mass>13.25</Estimated_mass></bar_mass>
        <bar_length><Notanumber>INDETERMINATE</Notanumber></bar_length>
</Steel_bar>

<Steel_bar id="id-3" >
   <bar_length><Floatingnumber>77.0</Floatingnumber></bar_length>
   <bar_mass>
        <Floatingnumber path="Computed_mass">13.25</Floatingnumber>
   </bar_mass>
</Steel_bar>
```

The first attribute of instance id-1 represents an Extended_Real value that is a FloatingNumber. It is mapped to the exchange structure, following 9.1.3 for FloatingNumber, as a value of the simple type REAL.

The second attribute of instance id-1 represents a Measured_Mass value, which is a valid Mass_Spec value and therefore a valid Mass value. It is mapped (by recursive application of 9.1.5, since Mass is a select data type and Mass_Spec is a select data type) as a Measured_Mass, as a value of the simple type REAL.

The first attribute of instance id-2 represents an Estimated_Mass value. This is a valid Mass_Spec value and also a valid Mass_Substitute value and therefore a valid value of Mass. This value actually instantiates both (select) data types in the select-list of Mass.

The second attribute of instance id-2 represents an Extended_Real value that is a NotaNumber value.

The first attribute of instance id-3 is the same as the first attribute of instance #1.

The second attribute of instance id-3 represents a Computed_Mass value, which is a valid Mass_Spec value and therefore a valid Mass value. It is mapped (by recursive application of 9.1.5, since Mass is a select data type and Mass_Spec is a select data type) as Computed_mass (the simple defined type in the select-list). The value of Computed_Mass is an Extended_Real value. Extended_Real is a select data type so a path attribute value is added to show that it was derived from a Computed_mass.

9.1.6 Entity Instance reference element

A entity instance reference element shall be encoded as an XML element with the name of entity type referenced. The XML element shall contain a single XML attribute called href and this attribute shall be used to store the value of the Entity instance reference. There shall be no element content. The entity instance addressed by the reference shall be EXPRESS type compatible with the entity type named by the element.

EXAMPLE

Valid reference element	Meaning

<Point href="#id-12">	References entity instance with identifier id-12 The instance must be a Point or a type compatible with Point
<Point href="file.xml#id-12">	References entity instance with identifier id-12 in document file. that must be an instance of point or a type compatible with Point
<Point href="file.stp#id-10">	References entity instance with identifier id-12 stored in in the ISO 10303-21 "file.stp". The instance must be a point or type compatible with Point.

Invalid reference expressions	Problem
<Point href="other_file.xml">	Fragment identifier missing
<point href="#id-12">	XML element does not have the name of an entity
<Point href="#id-12", unset="true">	XML element has an attibute other than href

9.1.7 Entity Instance reference element short form

If an explicit attribute of an entity is defined by an entity type and the value of that attribute is defined by an entity instance reference element then a short form may be used to encode the reference. The short form shall move the href XML attribute of the entity instance reference element into the XML element representing the XML attribute and delete the entity instance reference element.

EXAMPLE

Valid short reference element	Meaning
<centre href="#id-12">	References entity instance with identifier id-12 The instance must be type compatible with the centre attribute
<end href="file.xml#id-12">	References entity instance with identifier id-12 in document file. The instance must be type compatible with the centre attribute
<knot href="file.stp#id-10">	References entity instance with identifier id-12 stored in in the ISO 10303-21 "file.stp". The instance must be with the knot attribute.

Invalid short reference expressions	Problem
<point href="other_file.xml">	Fragment identifier missing
<Point href="#id-12">	XML element does not have the name of an attribute
<point href="#id-12", unset="true">	XML element has an attibute other than href

EXAMPLE XML elements coded using short form

```
<Product_definition id ="id-1">
    <id>EXAMPLE</id>
    <description/>
    <formation href="#id-2"/>
<Product_definition/>
<Product_definition_formation id="id-2">
    <id>A</id>
    <description>version 1</description>
    <of_product href="#id-3">
```



```
        </product_definition>
    <Product id="id-3">
        <id>1</id>
        <name>widget</name>
        <description/>
        <frame_of_reference/>
    </Product>
```

9.2 Mapping of EXPRESS entity data types

An instance of an EXPRESS entity data type shall be mapped to the exchange structure as an ENTITY_INSTANCE.

As defined by ISO 10303-11:1994, a "simple entity instance" is an entity instance that is not an instance of a subtype of any entity data type. All other entity instances are called "complex entity instances". A simple entity instance shall be mapped as specified in 9.2.1. A complex entity instance shall be mapped as specified in 9.2.5.

NOTE A simple entity instance is an entity instance which is completely described by a *single* EXPRESS entity-declaration. A complex entity instance is an instance whose description involves more than one entity-declaration, even when only one of them contains explicit attributes. A simple entity instance can be an instance of a supertype, as long as it is not an instance of any subtype, but an instance of a subtype is always "complex".

Only the explicit attributes of an EXPRESS entity are mapped to the exchange structure. Special provisions apply to OPTIONAL explicit attributes (see 9.2.2), explicit attributes whose values are entity instances (see 9.2.4), and all redeclarations of explicit attributes (see 9.2.6, 9.2.7, and 9.2.8)

Each explicit attribute shall be mapped as an XML element whose name is defined by the keyword for that attribute. If the XML element has the href attribute then the value of the explicit attribute is described by an entity reference and the element shall have no other attributes and no XML content. If the XML element has content then it shall not have any XML attributes.

The order of the XML elements representing the attributes is not defined by EXPRESS or this part of ISO 10303.

If no value is given for an attribute then the absense of a value shall be indicated by not including any XML content for that attribute in the exchange structure or by including XML content and setting the unset XML attribute of that content to the value "true".

NOTE 1 More than one such provision may apply to the same attribute.

NOTE 2 If an application that wants to include additional XML content (elements or attributes) in an exchange structure it should put that content in a name space that is not part of the exchange structure as defined in the Header element (see 7.2.5.)

9.2.1 Mapping of a simple entity instance

A simple entity instance shall be mapped as an XML element in the exchange structure. The name of the XML element shall be mapped to the keyword defined for the entity as specified in 6.1.1.

Each explicit attribute shall be mapped directly to a nested XML element. If the EXPRESS entity data type has no explicit attributes, then the content of the XML element shall be empty.

NOTE If the XML element representing the entity has no content for an explicit attribute then the exchange structure is not in syntactic conformance. If the XML element has content for an explicit attribute but the XML element representing the attribute has no content then the exchange structure is in syntactic conformance but not AIM or ARM conformance.

EXAMPLE Definition in EXPRESS:

```
TYPE
  primary_colour_abbreviation = ENUMERATION OF (r, g, b);
END_TYPE;

ENTITY widget; ------------------------------>  A
  attribute1: INTEGER; ---------------------->  B
  attribute2: STRING;  ---------------------->  C
  attribute3: LOGICAL; ---------------------->  D
  attribute4: BOOLEAN; ---------------------->  E
  attribute5: REAL; ------------------------->  F
  attribute6: LIST [1 : 2] of LOGICAL;  ---->  G
  attribute7: ARRAY [-1:0]  of INTEGER;  --->  H
  attribute8: PRIMARY_COLOUR_ABBREVIATION; ->  I
END_ENTITY;
```

Sample instance in the data section:

```
<Widget id="id-30">
      <attribute1>1</attribute1>
      <attribute5>1.0</attribute5>
      <attribute2>ABC</attribute2>
      <attribute4>true</attribute4>
      <attribute3>unknown</attribute3>
      <attribute6><Logical>true</Logical><Logical>false<Logical></attribute6>
      <attribute7><Integer>1</Integer><Integer>0</Integer></attribute7>
      <attribute8>R</attribute8>
</Widget>
```

A: The EXPRESS entity name widget is mapped to the WIDGET standard keyword of the data section entity.

B: attribute1 has a value of 1 in this entity instance.

C: attribute2 has a value of ABC in this entity instance.

D: attribute3 has a value of unknown in this entity instance.

E: attribute4 has a value of true in this entity instance.

F: attribute5 has a value of 1.0 in this entity instance.

G: attribute6 is a list of logicals in this entity instance. The list values are:

```
ATTRIBUTE6(1) = true
```



```
ATTRIBUTE6(2) = false
```

H: attribute 7 is an array of integers in this entity instance. The array values are:

```
ATTRIBUTE7(-1) = 1
ATTRIBUTE7( 0) = 0
```

I: Attribute 8 is an enumeration. The attribute contains a value of R.

9.2.2 Mapping of OPTIONAL explicit attributes

The XML element representing an explicit attribute that is declared to be OPTIONAL is not required to have any XML content. When the optional value is supplied, it shall be the content of the element and encoded according to the data type of the attribute, as specified in clause 9.

If the XML element representing an OPTIONAL attribute has the unset attribute set to "true" then the value of the entity instance attribute shall be unset. Any content defined for the XML element representing the entity instance attribute must still be syntactically correct.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY xxx;
  attribute1: REAL;
  attribute2: REAL;
END_ENTITY;

ENTITY yyy;  ------------------------>  A
  attribute1: OPTIONAL LOGICAL; ----->  B
  attribute2: xxx;  ------------------>  C
  attribute3: xxx;  ------------------>  D
  attribute4: OPTIONAL INTEGER; ----->  E
  attribute5: OPTIONAL REAL;  ------->  F
END_ENTITY;
```

Sample entity instances in data section:

```
<Yyy id="id-3">
    <attribute2>
        <Xxx href="#id-2"/>
    </attribute2>
    <attrbute3>
        <Xxx id="id-1">
            <attribute1>1.0</attribute1>
            <attribute2>2.0</attribute2>
        </Xxx>
    </attribute3>
</Yyy>
<Xxx id="id-2">
    <attribute1>1.0</attribute1>
    <attribute2>2.0</attribute2>
</Xxx>
```

A: The EXPRESS entity name yyy is mapped to the Yyy standard keyword of the data section entity.

B: attribute1 does not have a value in this entity instance.

C: attribute2 is a an xxx entity with entity instance #2.

D: attribute3 is a reference to the xxx entity with entity instance #1.

E: attribute4 does not have a value in this entity instance.

F: attribute5 does not have a value in this entity instance.

NOTE When an aggregate or string attribute is defined by an XML element with no content the value of that attribute is set to empty. The XML attribute unset allows this default to be overriden.

9.2.3 Mapping of derived attributes

The derived attributes of an entity shall not be mapped to the exchange structure. When a derived attribute in a subtype redeclares an attribute in a supertype, the mapping used shall be as described in 9.2.6.

9.2.4 Mapping of attributes whose values are entity instances

If an entity instance is specified as an attribute of a second (referencing) entity instance, the first (referenced) entity instance shall be mapped to the exchange structure as XML content (see 9.1) or as an entity instance reference element (see 9.1.6).

If the value is mapped as a referenced entity instance, then the referenced entity instance shall be a member of the entity instance population (see 8.3.1). Only one instance in the population shall contain an entity instance with this identifier. The name of the document containing the instance shall be conisdered to part of the identifier. If no document name is given in the reference then the URL of the current document shall be considered to be part of the identifier. The definition may precede or follow the use of the entity instance identifier as an attribute. The definition need not occur within the same data element or the same document as the entity instance reference element.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY yyy;
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;

ENTITY xxx;
  p0 :  yyy;  ----------->  A
  p1 :  yyy;  ----------->  B
END_ENTITY;
```

Sample entity instances:

```
<Xxx id="id-2">
     <p0>
```



```
            <Yyy href="#id-1"/>
        </p0>
        <p1>
            <Yyy id="id-1">
                <x>1.0</y>
                <y>2.0</y>
                <z>2.0</z>
            </Yyy>
        </p1>
    </Xxx>
```

EXAMPLE Data in two documents:

```
<AIM>
    ...
    <Xxx id="id-2">
        <p0 href="abc.xml#id-1"/>
        <p1>
            <Yyy id="id-1">
                        <x>1.0</y>
                        <y>2.0</y>
                        <z>2.0</z>
            </Yyy>
        </p1>
    </Xxx>
    ...
</AIM>
```

Sample instance in resource document abc.xml

```
    ...
    <Yyy id="id-1">
        <x>10.0</y>
        <y>20.0</y>
        <z>30.0</z>
    </Yyy>
    ...
```

EXAMPLE Referenced data in a ISO 10303-21 file:

```
<AIM>
    ...
    <Xxx id="id-2">
        <p0 href="abc.stp#id-1"/>
        <p1>
            <Yyy id="id-1">
                        <x>1.0</y>
                        <y>2.0</y>
                        <z>2.0</z>
            </Yyy>
        </p1>
    </Xxx>
    ...
</AIM>
```

Sample instance in resource file abc.stp

```
    ...
    #1= YYY(10.0, 20.0, 30.0);
    ...
```


9.2.5 Entities defined as subtypes of other entities

ISO 10303-11 defines instances of an entity having a SUBTYPE clause to be "complex entity instances", in that they may involve attributes from more than one entity-type declaration. This subclause specifies how complex entity instances are mapped to the exchange structure.

Complex entity instances shall be mapped to the exchange structure using one of two mapping rules, internal mapping or external mapping. One mapping rule applies to each instance of a subtype entity.

NOTE 1 The selection of mapping depends on the entity instance rather than the entity type. It is possible for different instances of the same entity data type to use different mappings, depending on whether they are instances of subtypes and which subtypes they instantiate.

NOTE 2 This subclause applies *only* to complex entity instances. It does not necessarily apply to every instance of a supertype entity. In particular, it does not apply to an instance of a supertype that is not an instance of any subtype. Such instances may exist if the supertype is not an abstract supertype and is not itself a subtype of some other entity. Such instances are mapped as provided in 9.2.1.

9.2.5.1 Default selection of mapping

A set of entity data type definitions that are linked by subtype and implicit or explicit supertype expressions define a set of complex entity instance structures, referred to as the evaluated set in annex B of ISO 10303-11. Each member of the evaluated set specifies a list of entity data type names.

Each particular instance of an entity data type corresponds to one member of the evaluated set. The mapping that may be applied to a particular instance depends on the member of the evaluated set to which it corresponds.

To determine the mapping rules to apply to a given entity instance:

a) determine the list of entity data type names that are included in the evaluated set member that corresponds to the entity instance;

b) identify from the list all entity-types that have no subtypes and all entity-types that may have subtypes but for which none of the subtypes appears in the list (evaluated set member) for this instance. The result of the evaluation is the pruned evaluated set;

c) if the pruned evaluated set only contains one entity data type, this entity-type is referred to as the "leaf entity data type" and the internal mapping shall be used. Otherwise the external mapping shall be used.

NOTE At least one entity data type will be identified in step b above.

9.2.5.2 Internal mapping

If the internal mapping is used, the entity instance shall encode the values of the inherited explicit attributes of all supertype entities and the explicit attributes of the leaf entity data type.

This procedure may result in a supertype entity being referenced more than once. In this case all references after the first one shall be ignored.

EXAMPLE 1 An example of a simple subtype/supertype relationship. Entity definition in EXPRESS:

```
ENTITY aa ABSTRACT SUPERTYPE OF (ONEOF(bb,cc));  ------>  A
  attrib_a: zz;  ---------------------------------------->  B
END_ENTITY;

ENTITY bb SUBTYPE OF (aa)
      ABSTRACT SUPERTYPE OF (ONEOF(xx)); ------------>  C
  attrib_b1: yy;  ------------------------------------->  D
  attrib_b2: yy;  ------------------------------------->  E
END_ENTITY;

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

ENTITY xx SUBTYPE OF (bb);
  attrib_x:  REAL;  ----------------------------------->  F
END_ENTITY;

ENTITY zz;
  attrib_z : STRING;
END_ENTITY;

ENTITY yy;
  attrib_1 : REAL;
END_ENTITY;
```

Sample entity instances:

```
<Zz id="i1">
     <attrib_z>ZATTR</attrib_z>
<Zz>
<Yy id="i2">
     <attrib_1>1.0</attrib_1>
</Yy>
<Yy id="i3">
     <attrib_1>2.0</attrib_1>
</Yy>
<Xx>
     <attrib_b1><Yy href="#i2"/></attrib_b1>
     <attrib_b2><Yy href="#i3"/></attrib_b2>
     <attrib_a><Zz href="#i1"/></attrib_a>
     <attrib_x>4.0</attrib_x>
</Xx>);
```

A: Because entity aa is an abstract supertype it does not map to the exchange structure.

B: The attribute attrib_a will map as an inherited attribute in an entity that is directly or indirectly subtyped to the aa entity. In this case, attrib_a is represented by the first attribute of the instance of xx, and refers to zz, entity instance i1.

C: Because entity bb is an abstract supertype it will not map to the exchange structure.

D: The attribute attrib_b1 will map as an inherited attribute in an entity that is directly or indirectly subtyped to the bb entity. In this case, attrib_b1 is represented by the second attribute of the instance of entity xx, and refers to yy, entity instance i2.

E: The attribute attrib_b2 will map to the data section as an inherited attribute in an entity that is directly or indirectly subtyped to the bb entity. In this case, attrib_b2 is represented by the third attribute of the instance of entity xx, and refers to yy, entity instance #3.

F: Attribute attrib_x is represented by its value 4.0 .

EXAMPLE 2 An example of the mapping of a supertype that is not an ABSTRACT supertype. Entity definition in EXPRESS:

```
ENTITY aa SUPERTYPE OF (ONEOF(bb,dd));  -->  A
  attrib_a :  STRING;
END_ENTITY;

ENTITY bb SUBTYPE OF (aa); ---------------->  B
END_ENTITY;

ENTITY cc SUBTYPE OF (bb);  ---------------->  C
  attrib_c :  INTEGER;
END_ENTITY;

ENTITY dd SUBTYPE OF (aa);  ---------------->  D
  attrib_d : REAL;
END_ENTITY;

ENTITY ee ;  ------------------------------>  E
  attrib_e : aa;
END_ENTITY;
```

Sample entity instances:

```
<Aa id="i1">
     <attrib_a>SAMPLE STRING</attrib_a>
</Aa>
<Dd id="i3">
     <attrib_a>XYZ</attrib_a>
     <attrib_d>99.99</attrib_d>
<Dd>
<EE id="i5">
     <attrib_e>
          <Aa href="#i1"/>
     </attrib_e>
</Ee>
<EE id="i6">
     <attrib_e>
          <Bb id="i2">
               <attrib_a>ABC</attrib_a>
```



```
            </Bb>
        </attrib_e>
    </Ee>
    <EE id="i7">
        <attrib_e>
            <Cc id="i3">
                <attrib_a>DEF</attrib_a>
                <attrib_c>123</attrib_c>
            </Cc>
        </attrib_e>
    </Ee>
    <EE id="i8">
        <attrib_e>
            <Aa href="#i4"/>
        </attrib_e>
    </Ee>
```

A: Since it was not an abstract supertype, the supertype entity aa can be instantiated in an exchange structure. Note that it contains only its attrib_a attribute when it is instantiated.

B: The entity bb is a subtype of aa and therefore its instances will contain the attributes of both aa and bb, but since entity bb does not define any attributes the parameter list will only contain attrib_a.

C: The entity cc is a subtype of bb and therefore its instances will contain the attributes of aa, bb, and cc.

D: The entity dd is a subtype of aa and therefore its instances will contain the attributes of both aa and dd.

E: The entity ee references entity aa as an attribute. Therefore, an instance of entity ee may reference or contain any of #1, #2, #3 or #4.

EXAMPLE 3 An example of the mapping of an entity with multiple supertypes in the SUBTYPE OF expression. Entity definition in EXPRESS:

```
ENTITY base SUPERTYPE OF (branch_one,branch_two);  --->  A
  attrib_a :  STRING;
END_ENTITY;

ENTITY branch_one SUBTYPE OF (base); ------------------>  B
  attrib_b :  INTEGER;
END_ENTITY;

ENTITY branch_two SUBTYPE OF (base);  ----------------->  C
  attrib_c :  BOOLEAN;
END_ENTITY;

ENTITY leaf SUBTYPE OF (branch_one, branch_two);  ---->  D
  attrib_d : REAL;
END_ENTITY;
```

Sample entity instance in data section:

```
<Base id="id-A">
    <attrib_a>SAMPLE STRING</attrib_a>
</Base>
<Branch_one id="id-B">
```



```
    <attrib_a>ABC</attrib_a>
    <attrib_b>123</attrib_b>
</Branch_one>
<Branch_two id="id-C">
    <attrib_a>DEF</attrib_a>
    <attrib_c>true</attrib_c>
</Branch_two>
<Leaf id="id-D">
    <attrib_a>XYZ</attrib_a>
    <attrib_b>123</attrib_b>
    <attrib_c>true</attrib_c>
    attrib_d>99.99</attrib_d>
</Leaf>
```

A: Entity base has no supertypes. When instantiated in an exchange structure, it will contain only a value for the attrib_a attribute.

B: The entity branch_one is a subtype of base. When instantiated in an exchange structure, it will contain the inherited attributes of base and the attributes of branch_one.

C: The entity branch_two is a subtype of base. When instantiated in an exchange structure, it will contain the inherited attributes of base and the attributes of branch_two.

D: The entity leaf is a subtype of branch_one and branch_two. When instantiated in an exchange structure, will contain the inherited attributes of branch_one, which includes the attributes of base, and the inherited attributes of branch_two and the attributes of leaf. The attributes of base are only written once.

9.2.5.3 External mapping

The external mapping shall be used if the pruned evaluated set contains more than one member. A complex entity instance shall be formed containing all the attributes of the members of the pruned evaluated set. A name shall be formed for the complex entity instance by joining the names of the entity instances in the set in alphabetical order using hyphens "-".

EXAMPLE 1 An example of the mapping of subtypes related by ANDOR.

Entity definition in EXPRESS:

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

ENTITY bb SUBTYPE OF (aa); ---------->  B
  attrib_b :  INTEGER;
END_ENTITY;

ENTITY cc SUBTYPE OF (aa);  ---------->  C
  attrib_c : REAL;
END_ENTITY;

ENTITY dd ;  ---------------------------->  D
  attrib_d : aa;
END_ENTITY;
```


Sample entity instances:

```
<Bb id="id-A">
    <attrib_b>15</attrib_b>
    <attrib_a>sample string</attrib_a>
</Bb>
<Cc id="id-B>
    <attrib_a>S</attrib_a>
    <attrib_c>3.0</attrib_c>
<Cc>
<Aa-Bb-Cc id="id-C">
    <attrib_a>ASTRID</attrib_a>
    <attrib_b>17</attrib_b>
    <attrib_c>4.0</attrib_c>
</Aa-Bb-Cc>
<Dd id="id-D1">
    <attrib_d>
        <aa href="#id-1"/>
    </attrib_b>
</Dd>
<Dd id="id-D2">
    <attrib_d>
        <aa href="#id-2"/>
    </attrib_b>
</Dd>
<Dd id="id-D3">
    <attrib_d>
        <aa href="#id-3"/>
    </attrib_b>
</Dd>
<Aa id="id=E">
    <attrib_a>ABC</attrib_a>
</Aa>
```

A: id-1 is an instance of aa and bb combined.

B: id-2 is an instance of aa and cc combined.

C: id-3 is an instance of aa, bb and cc combined.

D: The entity dd references entity aa as an attribute. Therefore, an instance of entity dd may legally reference any of id-1, id-2 or id-3.

E: The non-abstract supertype aa can be instantiated, and the internal mapping applies because the evaluated set contains only one member.

EXAMPLE 2 An example of the mappings of a more complicated subtype/supertype graph. Entity definition in EXPRESS:

```
ENTITY x;
  attrib_x : INTEGER;
END_ENTITY;

ENTITY a ABSTRACT SUPERTYPE OF(ONEOF(b,c));  -->  A
  attrib_a : x  ------------------------------>  B
END_ENTITY;
```



```
ENTITY b SUPERTYPE OF(d ANDOR e)
     SUBTYPE OF (a);
 attrib_b : REAL;  ----------------------------> B
END_ENTITY;

ENTITY c SUBTYPE OF (a);  ---------------------> C
 attrib_c : REAL;
END_ENTITY;

ENTITY d SUBTYPE OF (b);  ---------------------> D
 attrib_d : x;
END_ENTITY;

ENTITY e ABSTRACT SUPERTYPE
       SUBTYPE OF (b);  ------------------------> A
 attrib_e : x;  ------------------------------> B
END_ENTITY;

ENTITY f SUPERTYPE OF (h);
 attrib_f : x;  ------------------------------> B
END_ENTITY;

ENTITY g SUBTYPE OF (e);  ---------------------> E
 attrib_g : INTEGER;
END_ENTITY;


ENTITY h SUBTYPE OF (e,f);  -------------------> E
 attrib_h : INTEGER;
END_ENTITY;
```

A: Since entity a and e are abstract supertypes they cannot occur on the exchange structure as independent instances.

B: Since attrib_a, attrib_b, attrib_e and attrib_f are attributes of supertype entities, they will be mapped as inherited attributes if a subtype is mapped using the internal mapping. They will be mapped as attributes of the complex instance if a subtype is mapped using the external mapping.

C: Since entity c participates in an ONEOF operation and its supertype participates in no supertype operation, it will use the internal mapping.

D: The mapping of d will depend on the structure of the pruned evaluated set in which it appears.

E: Since entities g and h both have a supertype (entity e) that participates in an ANDOR operation. their mapping will depend on the structure of the pruned evaluated set in which they appear.

EXAMPLE 3 An entity instance showing internal mapping.

```
<C id="id-2">
     <attrib_c>2.0</attrib_c>
     <attrib_a>
          <X id="id-1">
               <attrib_x>1</attrib_x>
          </X>
     </attrib_a>
</C>
```


A: The evaluated set of 'id-2' is [c & a]. The pruned evaluated set is [c] and therefore uses the internal mapping.

EXAMPLE 4 Entity instance showing internal mapping:

```
<D id="id-2">
    <attrib_c>2.0</attrib_c>
    <attrib_a>
        <X id="id-1">
            <attrib_x>1</attrib_x>
        </X>
    </attrib_a>
    <attrib_d>
        <X id="id-4">
            <attrib_x>1</attrib_x>
        </X>
    </attrib_d>
</D>
```

A: The evaluated set of id-2 is [a & b & d] and the pruned evaluated set is [d] so it is internally mapped.

B: The attribute of a with name attrib_a is inherited by entity data type d.

C: attrib_b is inherited by entity data type d.

D: attrib_d is defined in entity data type d.

EXAMPLE 5 Entity instance showing external mapping:

```
<D-H id="id-2">
    <attrib_a>
        <X id="id-1">
            <attrib_x>1</attrib_x>
        </X>
    </attrib_a>
    <attrib_b>9.0</attrib_b>
    <attrib_c>2.0</attrib_c>
    <attrib_d>
        <X href="#id-1">
    </attrib_d>
    <attrib_e>
        <X href="#id-1">
    </attrib_e>
    <attrib_f>
        <X href="#id-1">
    </attrib_f>
    <attrib_h>4</attrib_h>
</D>
```

A: Since entity instance id-2 has the evaluated set [a & b & d & e & f & h] and the pruned evaluated set has more than one leaf (d and h), the external mapping is used.

9.2.5.4 Attributes with the same name.

An entity instance can have inherit two or more attributes with the same name. In the internal mapping one of the attributes may be defined locally and the other may be inherited from a super-type, or both of the atttributes may be inherited from (different super-types). In the external mapping two or more of the types in the pruned evaluated set may define attributes with the same name.

If an entity instance has two or more attributes have the same name then each shall be qualified with the name of the entity that defines that attribute. The qualification shall consist of the name of the entity followed by the period character "." and the name of the attribute. The qualified name shall be used to define the name of the XML element that represents the attribute in the exchange structure.

EXAMPLE 1 An example of the mapping of subtypes with conflicting attributes related by ANDOR.

Entity definition in EXPRESS:

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

ENTITY bb SUBTYPE OF (aa); ---------------->  B
  attrib_sn :  INTEGER;
END_ENTITY;

ENTITY cc SUBTYPE OF (aa);  ---------------->  C
  attrib_sn : REAL;
END_ENTITY;

ENTITY dd SUBTYPE OF (bb, cc);  ---------------->  C
  attrib_sn : BOOLEAN;
END_ENTITY;
```

Sample entity instance in data section:

```
<Bb id="id-A">
      <attrib_sn>15</attrib_sn>
      <attrib_a>sample string</attrib_a>
</Bb>
<Cc id="id-B>
      <attrib_a>S</attrib_a>
      <attrib_sn>3.0</attrib_sn>
<Cc>
<Aa-Bb-Cc id="id-C">
      <attrib_a>ASTRID</attrib_a>
      <Bb.attrib_sn>17</Bb.attrib_sn>
      <Cc.attrib_sn>4.0</Cc.attrib_sn>
</Aa-Bb-Cc>
<Dd id="id-D">
      <attrib_a>ASTRID</attrib_a>
      <Bb.attrib_sn>17</Bb.attrib_sn>
      <Cc.attrib_sn>4.0</Cc.attrib_sn>
      <Dd.attrib_sn>true</Dd.attrib_sn>
</Aa-Bb-Cc>
```

A: only one attribute has the name attrib_sn.

B: only one attribute has the name attrib_sn.

C: two types in the pruned evaluated set described attributes with the name attrib_sn so they are each qualified with the name of the entity that defines the attribute.

D: entity dd inherits two attributes with the name attrib_sn and also defines a local attribute with that name. All three attributes are qualifed with the name of their defining entity type.

9.2.6 Explicit attributes redeclared as DERIVEd

If a subtype entity redeclares an attribute of its supertype using the DERIVE clause then there is no effect on the encoding. The redeclared attribute is not encoded in anyway.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY point;
  x : REAL;
  y : REAL;
  z : REAL;
END_ENTITY;

ENTITY point_on_curve SUBTYPE OF (point);
  u : REAL;
  c : curve;
DERIVE
  SELF\point.x : real := fx(u, c);
  SELF\point.y : real := fy(u, c);
  SELF\point.z : real := fz(u, c);
END_ENTITY;

ENTITY curve;
  attr : STRING;
END_ENTITY;
```

Sample entity instance in data section

```
<Point_on_curve id="i1">
    <u>0.55</u>
    <c>
        <Curve id="i2">
            <attr>curve_attribute</curve>
        </Curve>
    </c>
<Point_on_curve>
```

9.2.7 Attributes redeclared as INVERSE

If a subtype entity redeclares an attribute of its supertype using the INVERSE clause, there is no effect on the encoding. The redeclared attribute is not encoded in any way.

9.2.8 Attributes redeclared as explicit attributes

If a subtype entity redeclares an attribute of one of its supertypes as an explicit attribute then it shall be encoded using the rules defined for its redeclared type as specified in 9.2.5, applying the mapping specified in clause 9 for the redeclared data type of the attribute.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY aaa SUPERTYPE OF (ONEOF (bbb, ccc));
  a1    : NUMBER;
  a2    : curve;
INVERSE
  a3    : SET OF mmm FOR m1;
END_ENTITY;

ENTITY bbb SUBTYPE OF (aaa);
  SELF\aaa.a1   : INTEGER;
  b             : REAL;
END_ENTITY;

ENTITY ccc SUBTYPE OF (aaa);
  SELF\aaa.a2   : line;
INVERSE
  SELF\aaa.a3   : SET [1:2] OF mmm FOR m1;
END_ENTITY;

ENTITY curve;
  ...
END_ENTITY;

ENTITY line SUBTYPE OF (curve);
  ...
END_ENTITY;

ENTITY mmm;
  m1    : aaa;
END_ENTITY;
```

Sample instantiation in data section:

```
<Line id="i1">
      ...
</Line>
<Curve id="i2">
      ...
</Curve>
<Bbb id="i3">
      <a1>1.0</a1>
      <a2>
            <Curve href="#i2">
      </a2>
      <b>0.5</b>
</Bbb>
<Ccc id="i4">
      <a1>1.5</a1>
      <a2>
            <Line href="#i1">
      </a2>
</Ccc>
```


9.2.9 Entity local rules

Entity local rules, WHERE rules and UNIQUE rules, shall not be mapped to the exchange structure.

9.2.10 Mapping of INVERSE attributes

Attributes within the INVERSE clause shall not be mapped to the exchange structure.

9.2.11 Encoding of short names

Short names shall have no effect on the encoding.

9.3 Mapping of the EXPRESS element of SCHEMA

The EXPRESS element of SCHEMA shall not be mapped to the exchange structure. The name of the schema that specifies entities that appear in an exchange structure shall be mapped to the header section of the exchange structure by use of an instance of the **exchange_schema** entity data type as specified in 7.2.3.

9.4 Mapping of the EXPRESS element of CONSTANT

The EXPRESS element of CONSTANT shall not be mapped to the exchange structure.

NOTE The existence of multiple references to the same constant is not preserved when that data is mapped to the exchange structure.

9.5 Mapping of the EXPRESS element of RULE

The EXPRESS element of RULE shall not be mapped to the exchange structure.

9.6 Remarks

Remarks shall not be mapped to the exchange structure.

10 Mapping of Application Object instances to the exchange structure

This clause describes how application objects instances are mapped to the exchange structure.

Section 5.1 of an Application Protocol defines mapping paths for the Application Objects in the protocol. These mapping paths define constraints that must be meet by instances in the exchange structure in order for them to properly represent instances of the Application Object. This clause defines an encoding for these mapping paths so that conformance checking can be defined for application object instances independently of, or in addition to, AIM conformance.

NOTE Conformance checking of Application Objects enables modular reuse of application protocol data.

NOTE Conformance checking of all the Application Object instances in an exchange structure is not sufficient to ensure AIM conformance because additional constraints on data sharing between Application Objects instances are defined by the Integrated Resource parts.

10.1 Mapping of Application Object

An application object instance shall be mapped as an XML element in the exchange structure. The application object name shall be mapped to the name of the XML element as specified in 6.1.3. The content of the element shall consist of XML elements representing the attributes so the Application Object encoded using the rules described in 10.2. No order is defined for these attributes.

NOTE 1 The identity of an Application Object instance is defined by the entity instance that represents the AIM element assigned to that Application Object by the mapping table, not by an indentifier on the element representing the Application Object instance.

NOTE 2 Application Object instances reference each other in-directly when the last element in a mapping path is the root instance of another Application Object instance of the correct type.

10.2 Mapping of Application Object attribute

If an Application Object attribute is mapped to the exchange structure, it shall be mapped as an XML element with the name of application object attribute name as specified in 6.1.4

Both the explicitly defined attributes of the Application Object instance and the inherited attributes of the Application Object instance can be mapped.

If an Application Object attribute is mapped to the exchange structure then a mapping path must be mapped to the exchange structure as the XML conent of the XML element representing the Application Object attribute using the rules described in 10.3.

NOTE 1 There is no requirement to map an Application Object attribute. If the attribute is mapped then the representation of that attribute can be ARM conformance checked. If it is not mapped then its representation cannot be ARM conformance checked.

NOTE 2 Appliction Protocols frequently include ARM models. However, there is no requirement for these models to be defined in EXPRESS.

EXAMPLE Application Object definition in a mapping table

MANUFACTURING_FEATURE	shape_aspect	41		
its_id	shape_aspect.name			

manufacturing_feature to workpiece (as its_workpiece)	PATH			shape_aspect shape_aspect.of_shape -> product_definition_shape <= property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition product_definition

Sample instantiation in ARM data section:

```
<MANUFACTURING_FEATURE>
    <its_id>
    ...
    </its_id>
    <its_workpiece>
    ...
    </its_workpiece>
    ...
</MANUFACTURING_FEATURE>
```

10.3 Mapping of Application Object attribute path

The mapping path of an Application Object attribute shall be encoded as a sequence of reference elements in the order given in the mapping table. Each reference element shall be an entity instance reference or an attribute value reference.

An entity instance reference shall be encoded as described in 9.1.6

A attribute value reference shall be encoded as an XML element with the name of the attribute and no XML content.

The first item in each path for an Application Object instance shall be an entity instance reference to the same entity instance. This instance is called the root of the Application Object instance.

If the mapping defines a path to another Application Object instance then the last item in the path shall be an entity instance reference to the root of that other Application Object instance.

If the mapping path defines a path to a value or values, then the path shall terminate with attribute value references for each of those values. There shall be one attribute value for each value defined by the path. All of the attribute values in the path shall be attributes of the last entity instance reference given in the path. There shall be no additional entity instance references in the path after the first attribute value reference has been given.

If the mapping path as defined in the Application Protocol allows alternate paths to be used to represent the Application Object attribute instance then the exchange structure for this instance shall encode the chosen path.

NOTE 1 Some application protocols define paths that terminate in multiple values. If a path is to more than one value then the path will end with more than one attribute value reference. However, all of the attribute value references will belong to the same entity instance which will be the last entity instance referenced in the path.

NOTE 2 Some application protocols define paths that terminate in values that are described by entity instances belonging to one of the integrated resource parts. The path will terminate with attribute value reference. However, if accessed the attribute will contain another entity instance and not a primitive value.

NOTE 3 An entity instance can be the root of more than one application object instance if all the application object instances have different types. A system that uses root entity instances to navigate the ARM structure of an Application Protocol needs to know which type of application object is being referenced by each path.

EXAMPLEMapping Path for its_id attribute instance of a Manufacturing Feature

```
<MANUFACTURING_FEATURE>
    <ITS_ID>
        <Instanced_feature-pocket href="#id1-1"/>
        <name/>
    </ITS_ID>
    <ITS_WORKPIECE>
        <Instanced_feature-pocket href="#id1-1"/>
        <Product_definition_shape href="#id1-38"/>
        <Product_definition href="WORKPIECE_38.xml#id1-1"/>
    </ITS_WORKPIECE>
    ...
</MANUFACTURING_FEATURE>
```

The first mapping path ends in a value defined in the Instanced_feature-pocket entity instance.

The second mapping path ends in an entity instance that is the root of another Application Object. The definition of the AP requires this application object to be a workpiece.

EXAMPLE Entity definition in EXPRESS:

```
ENTITY person ABSTRACT SUPERTYPE;
  name    : STRING;
  personal_address: address;
END_ENTITY;

ENTITY male SUBTYPE OF (person);
END_ENTITY;

ENTITY female SUBTYPE OF (person);
END_ENTITY;

ENTITY business;
  bussiness_name    : STRING;
  business_category : category;
  business_address  : address;
END_ENTITY;

TYPE category= ENUMERATION OF (FACTORY, OFFICE); END_TYPE;

ENTITY employment;
  employee   : person;
  employer   : business
END_ENTITY;
```



```
ENTITY marriage;
    husband  : male;
    wife     : female;
END_ENTITY;
```

Application Object definition for a mailing list for a fishing catalog. The catalog is only interested in males. If the male works in an office then the mailing address is mapped to the office address. If the male does not work in an office then his personal address is used.

MAIL_TARGET	male	41		person => male
name	person.name			person.name
spouse_name	person.name			person <- marriage.husband marriage marriage.wife ->person person.name
address #1 if works in an office #2 if works in factory	PATH			#1: (person <- employment.person employment employment.employer -> business {business.business_category = "office"} business.address) #2 (person {person <- employment.person employment.employer -> business business.business_category = "factory"} person.address)

Sample instantiation in AIM data section:

```
<AIM>
    <Male id="id-1">
        <name>Jack</name>
        <personal_address href="personal_data.xml#Jack"/>
    </Male>
    <Male id="id-2">
        <name>Bill</name>
        <personal_address href="personal_data.xml#Bill"/>
    </Male>
    <Female id="id-3"
        <name>Jill</name>
        <personal_address href="personal_data.xml#Jill"/>
    </Female>
    <Business id="id-4">
        <name>Big Office Inc.<name>
        <business_address href="office_data.xml#id-1"/>
        <business_category>OFFICE</business_category>
    </Business>
    <Business id="id-5">
```



```
            <name>Little Factory Inc.<name>
            <business_address href="factory_data.xml#id-1"/>
            <business_category>FACTORY</business_category>
        </Business>
        <Employment id="id-7">
            <employee href="#id-1">
            <employer href="#id-4">
        </Employment>
        <Employment id="id-8">
            <employee href="#id-2">
            <employer href="#id-5">
        </Employment>
        <Employment id="id-9">
            <employee href="#id-3">
            <employer href="#id-4">
        </Employment>
    </AIM>
```

Sample instantiation in ARM data section:

```
    <ARM>
        <MAILING_TARGET>
            <name>
                <Male href="#id-1">
                <name/>
            <name/>
            <spouse_name>
                <Male href="#id-1">
                <Marriage href="personal_data#JackAndJill">
                <Female href="id-3">
                <name/>
            </spouse_name>
            <address>
                <Male href="#id-1">
                <Business href="#id-4">
                <Address href="business_data.xml#id-1">
            <address/>
        </MAILING_TARGET>
        <MAILING_TARGET>
            <name>
                <Male href="#id-1">
                <name/>
            <name/>
            <address>
                <Male href="#id-1">
                <Address href="address_data.xml#Bill">
            <address/>
        </MAILING_TARGET>
    </ARM>
```

NOTE The first mailing target has a spouse and the address used is his office.

NOTE The second mailing target has no spouse and the address used is his personal address.

NOTE The marriage of Jack and Jill is stored in an entity instance belonging to a different AIM section. If the exchange structure does not define a complete Application Protocol instance, then this entity instance can only be found using the entity instance reference given in the path.

10.4 Mapping path validation

A mapping path shall be validated by:

- verifiying that each entity instance referenced in the path exists in the exchange structure

- verifying that each entity instance referenced in the path has the values and relationships required by the mapping table. The meaning of these value and relationship requirements is as described in Section 5.1 of each protocol.

NOTE 1 If all the mapping paths are valid then the Application Object has a valid representation that can be resued and stored in a library. For example, it is anticipated that manufacturing systems will begin to build libaries of Application Objects representing common manufacturing features such as round_holes.

NOTE 2 There may be additional constraints on the Application Object defined elsewhere in the AP. For instance in the MAILING LIST example, presumably there is a constraint defined somewhere defined in the AP that allows the spouse_name to be optional. Unfortunately there is consistent specification for identifying how these constraints apply to individual Application Objects at the present time so conformance checking of these constraints is only possible on the AIM level.

NOTE 3 Individual exchange processing systems may be able to check Application Object instances for conformance to an EXPRESS model defined for the ARM but this checking is not part of this edition of this standard.

11 Conformance Classes

This clause defines the conformance classes for this part of ISO 10303. Three types of conformance are described and used to define six conformance classes.

11.1 Syntactic Conformance

Syntactic conformance with respect to the EXPRESS schema defined in the header requires:

— All of the entity instances in the exchange structure are defined by the EXPRESS schema;

— All of the entity instances in the exchange structure have values defined for all of their required attributes;

— All of the attributes (required and optional) are set to values of an allowed type;

— All of the XML elements in the exchange structure are assigned one of meanings described in this specification;

— All of the XML elements in the exchange structure that represent attributes are assigned to an entity instance;

— All of the XML elements in the exchange structure that represent values are assigned to an attribute.

11.2 AIM Conformance

AIM conformance with respect to the EXPRESS schema defined in the header requires:

— The exchange structure meets the requirements of syntactic conformance;

— All of the local rules of all of the entity instances evaluate to true or unknown;

— All of the global rules of the EXPRESS schema evaluate to true or unknown for the population as specified in 8.3.1.

11.3 ARM Conformance

ARM conformance with respect to the EXPRESS schema defined in the header requires:

— The EXPRESS schema identifies an Application Protocol;

— Each Application Object in the exchange structure is an Application Object of the Application Protocol;

— Each attribute of each Application Object is one of the attributes (explicit or inherited) identified for that object in the Application Protocol;

— The mapping path of each Application Object attribute is one of the paths defined for that attribute by the mapping table;

— Each entity instance referenced in the mapping path is a valid instance with the type allowed by the mapping table;

— Each entity instance identified by the mapping path has all the values required by the mapping table including all the forward and backward references between instances required by the mapping table and all the constant values required by the mapping table;

— Each value attribute identified by the mapping table is an attribute of the last entity instance referenced in the path.

11.4 Conformance Class 1

An exchange exchange structure shall be in conformance class 1 if:

— The exchange structure consists of a single document with exactly one header element and exactly one AIM element;

— All of the entity instance identifiers in the exchange are restricted entity instance identifiers as specified in 6.2.5;

— The exchange structure meets the requirements of syntactic conformance.

NOTE This is the XML equivalent of Conformance Class 1 of ISO 10303-21..

11.5 Conformance Class 2

An exchange structure shall be in conformance class 2 if:

— The exchange structure consists of a single document with exactly one header element and exactly one AIM element;

— All of the entity instance identifiers in the document are restricted entity instance identifiers as specified in 6.2.5;

— The exchange structure meets the requirements of AIM conformance.

NOTE This is the XML equivalent of Conformance Class 2 of ISO 10303-21.

11.6 Conformance Class 3

An exchange structure shall be in conformance class 3 if:

— The exchange structure meets the requirements of syntactic conformance.

NOTE This conformance class allows instance data to be distributed between multiple data elements and resource elements. The data elements can be AIM elements and ARM elements. The resource elements can be in XML documents and ISO 10303-21 files.

11.7 Conformance Class 4

An exchange structure shall be in conformance class 4 if:

— The exchange structure meets the requirements of AIM conformance.

NOTE This conformance class also allows instance data to be distributed between multiple data elements and resource elements. The data elements can be AIM elements and ARM elements. The resource elements can be in XML documents and ISO 10303-21 files.

11.8 Conformance Class 5

An exchange structure shall be in conformance class 5 if:

— The exchange structure meets the requirements of ARM conformance.

NOTE This conformance class also allows instance data to be distributed between multiple data elements and resource elements. The data elements can be AIM elements and ARM elements. The resource elements can be in XML documents and ISO 10303-21 files.

11.9 Conformance Class 6

An exchange structure shall be in conformance class 6 if:

— The exchange structure meets the requirements of AIM and ARM conformance.

NOTE This conformance class also allows instance data to be distributed between multiple data elements and resource elements. The data elements can be AIM elements and ARM elements. The resource elements can be in XML documents and ISO 10303-21 files.

Annex A
(informative)

Guidlines for translating ISO 10303-21 files to ISO10303-29

Annex C
(normative)

Information object registration

C.1 Document identification

In order to provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 10303 part(21) version(3) }

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

C.2 Schema identification

In order to provide for unambiguous identification of the header_section_schema in an open information system, the object identifier

{ iso standard 10303 part(21) version(3) object(1) header-section-schema(1) }

is assigned to the header_section_schema schema (see clause 7). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex D
(informative)

Guidlines for writing XML Schema

Annex E
(normative)

Protocol Implementation Conformance Statement (PICS) proforma

The Protocol Information and Conformance Statement (PICS) Proforma supports the conformance assessment of implementations requesting evaluation to this part of ISO 10303. This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preparation for conformance testing by a testing laboratory.

All implementors shall provide answers to the questions provided in E.1 and E.2.

E.1 Conformance to specified function

Check as many as are appropriate.

E.1.1 Entity instance encoding

Does the implementation support Conformance Class One?

___ for reading ___ for writing

Does the implementation support Conformance Class Two?

___ for reading ___ for writing

E.1.2 Short name encoding

Does the implementation support short names for entities

___ for reading ___ for writing

Does the implementation support short names for select type values

___ for reading ___ for writing

Does the implementation support short names for enumeration values

___ for reading ___ for writing

E.1.3 String encoding

Does the implementation support the \X\ encoding for 8-bit bytes?

___ for reading, and if so, what is the binary representation used by the implementation?

___ for writing

Does the implementation support the \S\ and \P\ encoding for ISO 8859 characters?

___ for reading, and if so, what is the binary representation used by the implementation?

___ for writing

Does the implementation support the \X2\ encoding for ISO 10646 characters?

___ for reading, and if so, what is the binary representation used by the implementation?

___ for writing

Does the implementation support the \X4\ encoding for ISO 10646 characters?

___ for reading, and if so, what is the binary representation used by the implementation?

___ for writing

E.2 Implementation limits

What is the maximum number of schemas that be referenced by an exchange structure?

What is the maximum number of data sections that may exist within an exchange structure?

What is the maximum number of entity instances that may exist within a data section?

What is the maximum number of entity instances that may exist within an exchange structure?

What is the maximum value (or binary representation used by the implementation) for entity instance identifiers that is supported?

What are the maximum and minimum values (or binary representation used by the implementation) for EXPRESS INTEGER that is supported?

What is the limit of EXPRESS REAL precision (or binary representation used by the implementation) that is supported?

What is the maximum length of EXPRESS STRING that is supported?

What is the maximum length of EXPRESS BINARY that is supported?

What is the maximum number of elements that may appear in the encoding of an aggregate?

What is the maximum nesting depth that may appear in the encoding of nested aggregates?

Annex F
(informative)

Example of a complete CC 2 exchange structure

F.1 Introduction

An example EXPRESS schema definitions, a table of short names and an exchange structure are presented below. This EXPRESS schema does not reflect the contents of any part of ISO 10303.

F.2 Example schema

The EXPRESS schema definitions used for the exchange structure example.

```
SCHEMA example_geometry;

TYPE length_measure = NUMBER;
END_TYPE;

ENTITY geometry
SUPERTYPE OF (ONEOF(point));
END_ENTITY;

ENTITY point
SUPERTYPE OF (ONEOF(cartesian_point))
SUBTYPE OF (geometry);
END_ENTITY;

ENTITY cartesian_point
SUBTYPE OF (point);
  x_coordinate  : length_measure;
  y_coordinate  : length_measure;
  z_coordinate  : OPTIONAL length_measure;
END_ENTITY;

TYPE edge_or_logical = SELECT (edge, edge_logical_structure);
END_TYPE;

ENTITY topology
SUPERTYPE OF (ONEOF(vertex, edge, loop));
END_ENTITY;

ENTITY vertex
SUBTYPE OF (topology);
  vertex_point : OPTIONAL point;
END_ENTITY;

ENTITY edge
```



```
SUBTYPE OF (topology);
  edge_start   : vertex;
  edge_end     : vertex;
END_ENTITY;

ENTITY edge_logical_structure;


  edge_element : edge;
  flag : BOOLEAN;
END_ENTITY;

ENTITY loop
SUPERTYPE OF (ONEOF(edge_loop))
SUBTYPE OF (topology);
END_ENTITY;

ENTITY edge_loop
SUBTYPE OF (loop);
  loop_edges : LIST [1:?] OF edge_or_logical;
END_ENTITY;

END_SCHEMA;
```

F.4 Example exchange structure

The following is an example of a complete exchange structure. Note that since the schema is only an example, there is no schema registration id present in the schema_name attribute of the **file_schema** entity instance in the header section.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('THIS FILE CONTAINS A SMALL SAMPLE STEP MODEL'),'3;1');
FILE_NAME('EXAMPLE STEP FILE #1',
'1992-02-11T15:30:00',
('JOHN DOE',
'ACME INC.',
'METROPOLIS USA'),
('ACME INC. A SUBSIDIARY OF GIANT INDUSTRIES','METROPOLIS USA'),
'CIM/STEP VERSION2',
'SUPER CIM SYSTEM RELEASE 4.0',
'APPROVED BY JOE BLOGGS');
FILE_SCHEMA(('EXAMPLE_GEOMETRY'));
ENDSEC;
DATA;
/*
    THE FOLLOWING 13 ENTITIES REPRESENT A TRIANGULAR EDGE LOOP
*/
#1=CPT(0.0,0.0,0.0);    /* THIS IS A CARTESIAN POINT ENTITY */
#2=CPT(0.0,1.0,0.0);
#3=CPT(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,.F.);   /* THIS IS AN EDGE LOGICAL STRUCTURE ENTITY */
#22=ED_STRC(#18,.F.);
#23=ED_STRC(#16,.T.);
#24=ED_LOOP((#21,#22,#23));   /* THIS IS AN EDGE LOOP ENTITY */
/*
    OTHER SYNTACTICAL REPRESENTATIONS WERE POSSIBLE. THE PREVIOUS
    EXAMPLE IS REPRESENTATIVE OF ONE POSSIBLE APPROACH.
*/
ENDSEC;
END-ISO-10303-21;
```

NOTE This example exchange structure has been edited to aid readability. Unnecessary *spaces* have been added to aid readability.

Annex G
(informative)

Example of a complete CC 4 exchange structure

Index

