INTERNATIONAL STANDARD

**ISO 24531**

First edition
2007-07-01

# Intelligent transport systems — System architecture, taxonomy and terminology — Using XML in ITS standards, data registries and data dictionaries

*Systèmes intelligents de transport — Architecture, taxinomie et terminologie des systèmes — Usage de XML dans les normes, registres de données et dictionnaires de données, en ITS*



Reference number
ISO 24531:2007(E)



ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

# Contents

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 standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. 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 document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 24531 was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

# Introduction

As the exchange of information via the Internet and other wired and wire-free networks develops and expands, the use of XML (extensible markup language) and its variants continues to grow and develop.

XML will be an important tool in the development and operation of intelligent transport systems (ITS) services.

Within XML and its variants, however, there are options. In order to obtain maximum benefit, interoperability and re-use of data within the ITS sector, it is important to implement XML and its variants in a consistent manner.

This International Standard provides the definition of how to use XML and its variants in a consistent and interoperable manner within the ITS sector.

# Intelligent transport systems — System architecture, taxonomy and terminology — Using XML in ITS standards, data registries and data dictionaries

## 1 Scope

The International Standard has been developed to assist developers and users of intelligent transport systems (ITS) standards who wish to use extensible markup language (XML), by providing a consistent definition of the rules and rule references for the use of XML within ITS systems. The scope of the International Standard is to define consistent rules and rule references to provide a framework to be used when implementing XML-based applications in ITS, and particularly, in specifying XML in ITS standards, ITS data registries and ITS data dictionaries. This International Standard also provides guidance and examples in respect of the use of XML in ITS, and the elaboration of XML within the abstract syntax notation one (ASN.1) data definitions required by ISO 14813-6 and ISO 14817.

This International Standard defines:

— Rules concerning the creation of XML Schemas for ensuring interoperability in various types of ITS applications that use XML (Clause 7, normative);

— Rules for using XML for the purpose of reusing XML Schemas (Clause 7, normative);

— Rules concerning registration and management of XML components in data dictionaries and data registries (Clause 8, normative);

— Examples of the use of XML in ITS applications (Annex A, informative);

— Representation of IRI (international resource identifiers) and/or ID-related constructs of this standard (Annex B, informative);

— Schema header template (Annex C, informative);

— Example of registering XML constructs (Annex D, informative);

— Example of automatic generation of an XML Schema from unified modelling language (UML) (Annex E, informative);

— Applying ASN.1 encoding for XML document (Annex F, informative);

— ASN.1 transformation to XML Schema example (Annex G, informative).

NOTE A table of language comparisons (XML, ASN.1, UML) may be found in ISO 14813-6.

## 2 Conformance

This International Standard prescribes a conceptual model; it does not define any single physical implementation. It provides a consistent and interoperable means of achieving interoperability for the



international exchange of information in XML application programs. Regional and national XML Schema have the option of providing additional schemas and variants for use in local situations.

In order to claim conformance with this International Standard, it is only required to design systems and exchange data internationally consistently in accordance with the provisions of this International Standard. No external conformance procedures are proposed or defined in this International Standard, although regional, national and local implementations are free to, and may choose to define and require local conformance procedures.

## 3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 31-0, *Quantities and units — Part 0: General principles*

ISO/IEC 8824-1:2002, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 8825-4:2002, *Information technology — ASN.1 encoding rules: XML Encoding Rules (XER)*

ISO/IEC 8825-5:2004, *Information technology — ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1*

ISO 14813-6, *Transport information and control systems — Reference model architecture(s) for TICS sector — Part 6: Data presentation in ASN.1*

ISO 14817:2002, *Transport information and control systems — Requirements for an ITS/TICS central Data Registry and ITS/TICS Data Dictionaries*

W3C Recommendation, *Extensible Mark-up Language (XML) 1.0, Second Edition*, 6 October 2000

W3C Recommendation, *Namespaces in XML*, 4 January 1999

W3C Recommendation, *XML Schema, Part 1: Structures, Second Edition*, 28 October 2004

W3C Recommendation, *XML Schema, Part 2: Datatypes, Second Edition*, 28 October, 2004

W3C Recommendation, *XML Linking Language (XLink), Version 1*, 27 June 2001

OMG, *XML Metadata Interchange (XMI) Specification, Version 2.0,* 3 May 2003

OMG, *Meta Object Facility (MOF) Specification, Version 2.0*, April 2004

ISOC, RFC 3987: *Internationalized Resource Identifiers (IRIs)*, January 2005

ISOC, RFC 2616: *Hypertext Transfer Protocol — HTTP/1.1*, June 1999

## 4 Terms and definitions

For the purposes of this document, the following terms and definitions shall apply.

**4.1**
**application**
program that reads XML documents and “does something useful” with them

NOTE Applications will normally be interfaced to an XML parser, for example via DOM or SAX.

**4.2**
**ASN.1 application**
application that uses ASN.1 encodings for communication (except XML encoding rules)

**4.3**
**ASN.1 schema**
definition of the content and structure of data using an ASN.1 type definition

NOTE ASN.1 is specified in ISO/IEC 8824.

**4.4**
**association end**
endpoint of an association, which connects the association to a classifier

**4.5**
**attribute**
⟨classifier⟩ feature that describes a range of values those instances of the classifier may hold

**4.6**
**attribute**
⟨element⟩ property

NOTE It is additional information about a piece of data (element). Often attributes are used to pass information about the element and hence can be said to provide metadata for the element. An attribute is a value indicator (=) and the attribute value is specified within the tag (i.e. <H3 align="centre">). Attribute in XML is a name="value" pair that can be placed in the start tag of an element. For XML, all values are quoted with single or double quotes.

**4.7**
**child element**
element contained within another element

NOTE The element containing other elements is a parent element.

**4.8**
**class**
description of a set of objects that share the same attributes, operations, methods, relationships and semantics

**4.9**
**class diagram**
diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships

**4.10**
**constraint**
semantic condition or restriction

NOTE Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML.

**4.11**
**content**
all data between the start tag and end tag of an element

NOTE Content may be made up of mark-up characters and character data.

**4.12**
**content model**
expression specifying what elements and data are allowed within an element

**4.13**
**data concept**
any of a group of data dictionary structures defined in ISO 14817 (i.e. object class, property, value domain, data element concept, data element, data frame, message, interface dialogue, association) referring to abstractions or things in the natural world that can be identified with explicit boundaries and meaning and whose properties and behaviour all follow the same rules

[ISO 14817]

**4.14**
**data dictionary**
organized and constructed (electronic data base) compilation of descriptions of data concepts that provides a consistent means for documenting, storing and retrieving the syntactical form (i.e. representational form) and the meaning and connotation of each data concept

[ISO 14817]

**4.15**
**data element**
⟨data concept⟩ some single unit of information of interest (such as a fact, proposition, observation, etc.) about some (entity) class of interest (e.g. a person, place, process, property, concept, association, state, event)

[ISO 14817]

NOTE A data element is considered to be indivisible in a particular context.

**4.16**
**data frame**
⟨data concept⟩ grouping of data elements primarily for the purpose of referring to the group with a single name, and thereby efficiently reusing groups of data elements that commonly appear together (as an ASN.1 SEQUENCE, SEQUENCE OF, SET, SET OF or CHOICE) in a message specification

**4.17**
**data registry**
store of data, characterized in a consistent manner, as determined according to the provisions of this International Standard, used for a specific purpose (in this case ITS)

[ISO 14817]

**4.18**
**data type**
type of content that an element contains in XML and UML

NOTE An author can specify an element's data type.

**4.19**
**declaration**
create new types (both simple and complex)

**4.20**
**definition**
enable elements and attributes with specific names and types (both simple and complex) to appear in document instances

**4.21**
**document type definition**
rules that define the tags that can be used in an XML file and their valid values

**4.22**
**EBNF**
formal set of production rules that comprise a grammar defining language, such as XML

**4.23**
**element**
⟨XML⟩ logical data structure within an XML document, a piece of data within a file

NOTE An XML element consists of a start tag and end tag, and the information between the tags, which is often referred to as the contents. Start tags and end tags show the beginning and end of an element. A schema that can provide a description of the structure of the data describes elements used in an XML file.

**4.24**
**element**
atomic constituent of the UML model

**4.25**
**end tag**
element delimiter

NOTE In: <foo>this is a bar</foo> the construct </foo> is the end tag. End tags cannot include anything other than the element name and trailing space.

**4.26**
**Internet (uniform) resource identifier**
**IRI**
compact string of characters for identifying an abstract or physical resource

**4.27**
**lexical space**
the set of valid literals for a datatype

**4.28**
**local**
element, group, attribute, attribute group or data types that are not global

**4.29**
**mark-up**
identification of element types and structure within a document

NOTE The mark-up is not actually part of the content, but identifies the components and their roles.

**4.30**
**message**
⟨data concept⟩ grouping of data elements and/or data frames as well as associated message metadata, that is used to convey a complete unit of information

[ISO14817]

**4.31**
**metadata**
data that defines and describes other data

**4.32**
**namespace**
set of unique identifiers

NOTE Namespace is a mechanism to resolve naming conflicts between elements in an XML document when each comes from a different vocabulary. It allows the commingling of like tag names from different namespaces. A namespace identifies an XML vocabulary defined within a uniform resource name (URN). An attribute on an element, attribute or entity

reference associates a short name with the URN that defines the namespace; that short name is then used as a prefix to the element, attribute or entity reference name to uniquely identify the namespace. Namespace references have scope. All child nodes beneath the node that specifies the namespace inherit that namespace. This allows nonqualified names to use the default namespace.

**4.33**
**namespace**
part of the model in which the names may be defined and used

NOTE Within a namespace, each name has a unique meaning.

**4.34**
**namespace prefix**
see prefix

**4.35**
**node**
elements, comments, processing instructions and text in an XML document

NOTE An XML document has a hierarchical structure, described as a tree. The tree has branches connecting at the nodes.

**4.36**
**object class**
⟨data concept⟩ construct used to represent any kind of object (also referred to as an entity) within an ITS/TICS information environment

[ISO 14817]

**4.37**
**OID**
globally unique value associated with an object to identify it unambiguously

**4.38**
**package**
general purpose mechanism for organizing elements into groups

NOTE Packages may be nested within other packages.

**4.39**
**parser (for XML)**
processor that reads an XML document and determines the structure and properties of the data

NOTE If the parser goes beyond the XML rules for conformance and validates the document against an XML Schema (or DTD), the parser is said to be a “validating” parser. A generalized XML parser reads XML files and generates a hierarchically structured tree, then hands off data to viewers and other applications for processing. A validating XML parser also checks the XML syntax and reports errors.

**4.40**
**prefix**
**namespace prefix**
short name to identify uniquely the namespace

**4.41**
**profile**
stereotyped package that contains model elements, which have been customized for a specific domain or purpose using extension mechanisms, such as stereotypes, tagged definitions and constraints

NOTE A profile may also specify model libraries on which it depends and the metamodel subset that it extends.

**4.42**
**property**
named value, with semantic impact, denoting a characteristic of an element

NOTE Certain properties are predefined in the UML; others may be user defined (see tagged value).

**4.43**
**role**
named specific behaviour of an entity participating in a particular context

**4.44**
**schema**
system of representing an information model that defines the data's elements and attributes

**4.45**
**schema processor**
processor to validate schema

**4.46**
**stereotype**
new type of modelling element that extends the semantics of the metamodel

NOTE Stereotypes are based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user defined. Stereotypes are one of three extensibility mechanisms in UML.

**4.47**
**tagged value**
explicit definition of a property as a name-value pair; in a tagged value, the name is referred as the tag

NOTE Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML.

**4.48**
**tags**
text structures that mark up characters which mark the beginning and end of elements within the XML document

**4.49**
**value domain**
⟨data concept⟩ expression of a specific and explicit representation of some information about something of interest within the ITS/TICS domain

[ISO 14817]

**4.50**
**XMI**
XML-based model interchange format for UML models

**4.51**
**XML application**
application that uses XML encoding

**4.52**
**XML OID**
XML representation of an ASN.1 OID

NOTE In the following example, the ASN.1 OID delimiter (white space) changed by a designated delimiter.

EXAMPLE

```
ASN.1 OID : iso standard 24531 schema 1

XML OID (delimiter "_"): iso_standard_24531_schema_1_v_1_._0

XML OID (delimiter "/"): iso/standard/24531/schema/1/v1.0
```

## 5 Symbols and abbreviated terms

**ASN.1**
abstract syntax notation one

**BER**
basic encoding rule

**CER**
canonical encoding rule

**DD**
data dictionary

**DER**
distinguished encoding rule

**DR**
data registry

**DTD**
document type definition

**HTML**
hypertext markup language

**IRI**
internationalized resource identifiers

**ISO**
International Organization for Standardization

**ITS**
intelligent transport system(s)

**OID**
object identifier

**OMG**
object management group

**PER**
packed encoding rule

**RFC**
request for comments

**TICS**
transport information and control system(s)

**UML**
unified modelling language

**URL**
uniform resource locator

**W3C**
World Wide Web consortium

**WSDL**
web services description language

**XHTML**
extensible hypertext markup language

**XMI**
XML metadata interchange

**XML**
extensible markup language

## 6 Document convention

In this International Standard the following documentation conventions are used:

a) The term “schema” with a small “s” is used generically (to include DTDs, XML Schema and in some case ASN.1 schemas), while the term XML Schema or just Schema (capital “S”) refers specifically to schemas authored in accordance with the World Wide Web Consortium (W3C) XML Schema recommendation.

b) For reasons of brevity, not all examples are full schemas. In all prose and examples, the *xs* name space prefix is mapped to the namespace of the XML Schema language http://www.w3.org/2001/XMLSchema, even if no namespace declaration appears in the example. Similarly, the *xsi* and *xmi* namespace prefixes are mapped to the namespace name of Table 1.

**Table 1 — Namespace prefix and associated namespace**

| Namespace prefix | Associated Namespace |
|---|---|
| *xs* | http://www.w3.org/2001/XMLSchema |
| *xmi*, *xsi* | http://www.omg.org/2001/XMLSchema-instance |

EXAMPLE <xs:simpleType name = “car”/>

In this case, it is understood that

```
<xs:schema xlmns:xs = “http://www.w3.org/2001/XMLSchema”>
```

has already been declared.

c) Even if no end tag appears in the example, assume the end tag is declared at the appropriate place.

d) All examples are only for the purpose of explanation therefore informative. All IRIs in example are virtual with the exception of http://www.w3.org/2001/XMLSchema.

e) Throughout this International Standard, in accordance with ISO 31-0, decimal separators will be a point on the line.

## 7 Requirements

Figure 1 shows the scope of XML functionality in the ITS sector.

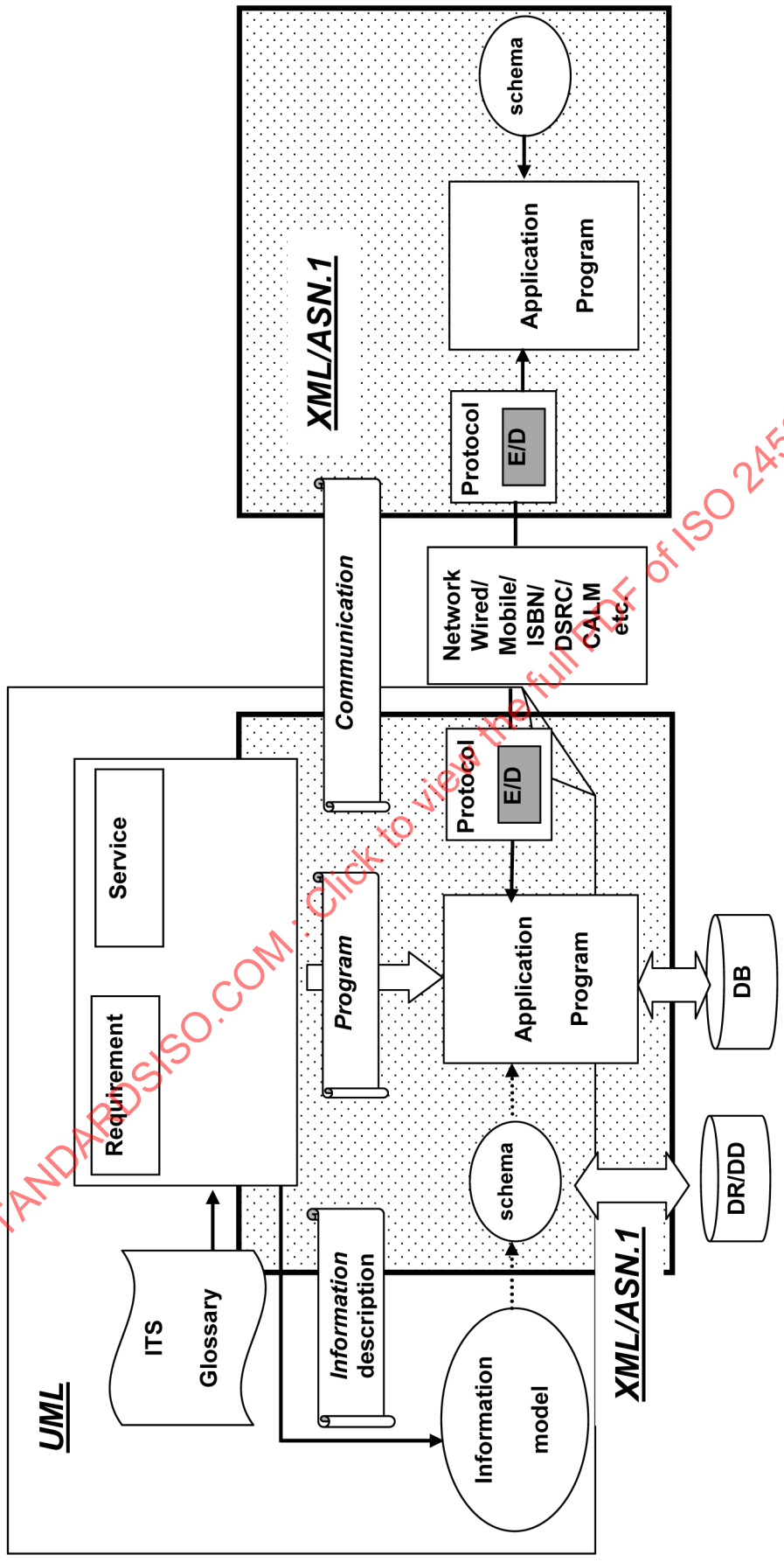



Figure 1 — XML functionality

## 7.1 Required conditions

Intelligent transport systems are evolving social infrastructure systems that offer various functions. To achieve their benefits, the following exchanges occur:

— information exchanges between various countries organizations including web services usages;

— information exchanges between different ITS functional areas;

— information exchanges between ITS related industries' systems; and

— information exchanges through various networks.

## 7.2 Required items

From the viewpoint of ITS information technology, the following items are required:

— formal method to define precise and unambiguous ITS vocabularies;

— registration and management rules for XML components, management and maintenance rules (ITS data registries and ITS data dictionaries);

— formal method to define dialogues and messages;

— expandable and reusable vocabularies and programs;

— ways to support various networks (wired/mobile/DSRC/digital broadcast /CALM, etc.);

— efficient encoding method;

— automatic generation of XML Schemas (or DTDs) from UML; and

— rules for the automatic transformation between ASN.1 module to/from XML Schemas.

## 7.3 Rules for using XML in ITS standards

Rules for using XML and its variants in ITS standards are given below. Rules are described following usual Schema development process. 7.3.1 defines the rules for designing Schema and Clause defines the rules for Schema component definition.

A strictly conforming implementation shall be strictly conforming to the "REQUIREMENT" set.

### 7.3.1 Rules for XML Schema design

#### 7.3.1.1 Use of the W3C XML Schema language

The W3C XML Schema language shall be used as the descriptive language for schemas when using XML for International Standards.

NOTE Although there are other descriptive languages for schemas, such as RELAX NG, use of the W3C XML Schema language is overwhelmingly predominant in applications written in XML. The W3C XML Schema language is used to enable interoperability with other applications.

#### 7.3.1.2 Use of XML Schemas in preference to DTDs

XML Schema language shall be used to create schemas.

NOTE Within XML, DTDs are allowed to be used to create schemas. DTDs are simpler than XML Schema and can provide a good fit for many text-oriented applications; however, ITS applications are generally not just text-oriented and XML Schemas provide rich datatype definitions that improve the precision of vocabulary definition and document validation.

Also, the XML schema language supports “Namespace” that facilitates easier reuse of vocabulary. ITS applications shall therefore use XML Schemas.

#### 7.3.1.3 Use of subschema, linear dependency graphs

In order to promote reusability and code management, large schemas shall be constructed of smaller reusable packages wherever possible, and there shall be a strictly linear dependency graph between subschemas.

NOTE XML allows cyclic and other complex references.

#### 7.3.1.4 Explicit indication of version attribute and encoding attribute in XML declaration statement

When creating XML documents in ITS applications, the XML declaration statement shall be declared with the XML version number attribute and encoding attribute explicitly indicated.

Because Schema syntax may be revised, the version number of a Schema shall be explicitly defined when the Schema is created to ensure interoperability.

EXAMPLE

```
<?xlm version = ‘1.0’ encoding = ‘UTF-8’ ?>
```

#### 7.3.1.5 Declaration of version attribute in Schema elements

The version attribute shall be explicitly declared in Schema elements.

It is highly likely that applications that use XML will be revised, including extension of functions and deletion of unnecessary functions. In such cases, the Schema is generally revised. It shall be clearly indicated which version of a Schema is currently being used.

EXAMPLE

```
<xs:schema xlmns:xs=”http://www.w3c.org/2001/XLMSchema”
xmlns =”http://www.example.com/aa”
targetNamespace=”http://www.rxample.com/aa”
version = “1.0”>
...
</xs:schema>
```

NOTE Namespace and targetNamespace values are virtual.

#### 7.3.1.6 Schema versioning rule

Released Schemas shall provide a version number in the form n.m (e.g. 1.2), and drafts also shall provide a version number of the form n.ma (e.g. 1.2b).

#### 7.3.1.7 Major version number rule

The major version number (n) shall be revised upwards when the change from the previous version of Schema will cause existing documents to fail to validate.

#### 7.3.1.8 Minor version number rule

The minor version number (m) shall be revised upwards when the change to the Schema will result in all existing documents continuing to validate. However, some new documents, which validate against the new version will fail against the old version.

#### 7.3.1.9 Version letter rule

The version letter (a) shall be revised upwards every time a new draft is issued.

Indicating the version of a Schema is good practice and helps prevent problems caused by people accidentally working with incorrect Schema versions. Therefore a common versioning rule and indicating the Schema version attribute shall be used.

EXAMPLE

Listing 7-1 Version 1.0 document

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema targetNamespace='http://www.iso.org/aa' version= '1.0' xmlns ='http://www.iso.
org' xmlns:xs='http://www.w3.org/2001/XMLSchema'>
   <xs:element name='temperature' type='xs:decimal/>
</xs:schema>
```

Listing 7-2 Version 1.1 document

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema targetNamespace='http://www.iso.org/aa' version= '1.1' xmlns ='http://www.iso.
org' xmlns:xs='http://www.w3.org/2001/XMLSchema'>
   <xs:element name='temperature' type='xs:decimal/>
   <xs:element name='humidity' type='xs:decimal minOccurs="0" />
</xs:schema>
```

NOTE "Humidity element" is added to the new version Schema. Therefore, the new version conformant document could be invalid to the old version Schema, if the "humidity" is added. So the minor version number is incremented (in this example to 1.1).

Listing 7-3 Version 2.0 document

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema              targetNamespace='http://www.iso.org/aa'              version='2.0'
xmlns:temp='http://www.iso. org' xmlns:xs='http://www.w3.org/2001/XMLSchema'>
   <xs:element name='temperature' type='xs:decimal' />
   <xs:element name='windVelocity' type='xs: decimal'/>
</xs:schema>
```

New mandatory element "windVelocity" is added; therefore old version document fails to validate. Therefore, new Schema shall be the incremented major version number.

#### 7.3.1.10 Schema ID attributes in the Schema element

The Schema ID attribute of the Schema element shall normally be used to indicate the identity of the Schema. The Schema ID attribute shall always use XML OID (delimiter "_"). (See Annex B.) The XML ID attribute is given its value from the following rule: First three letters of ID are "iso", after which follows number or identifier of the ASN.1 OID. In the case of ASN.1 OID string include a white space, as a delimiter of the OID node, thus "_" is used for XML ID. See ISO/IEC 8824-1 Annex D .

NOTE 1 Using XML OID as ID attribute of the Schema clarifies the application area and facilitates easy search of the Schema and its constructs from the DR/DD.

NOTE 2 The lexical space of XML ID differs from that of the ASN.1 OID.

EXAMPLE

```
<xs：schema …… id="iso_standard_24531_schema_1"……>
```

#### 7.3.1.11 Explicit indication of *targetNamespace* in Schema elements

The *targetNamespace* shall be explicitly declared and shall be explicitly defined in Schema by “http://"host"/"XML OID (delimiter "/”).

NOTE The namespace system facilitates use of the same names in many different fields without any naming collisions. The use of XML in various ITS fields may lead to the same names being used even though they have different meanings. The t*argetNamespace* shall therefore be explicitly defined to facilitate the use of names without any collisions. In respect of the term “host”, see IETF's rfc2616. The system manager decides the host.

EXAMPLE

```
<xs:schema targetNamespace = “http://www.example.com/iso/standard/24531/schema/1/v1.0”>
</xs:schema>
```

#### 7.3.1.12 Application of the *targetNamespace*+“xsd” for the *schemaLocation*

The targetNamespace “.xsd” naming convention shall be used as the *schemaLocation*.

NOTE The use of this convention will make it possible to infer the contents of a Schema from the file name.

EXAMPLE

```
http://www.its.org/iso/standard/24531/schema/1/v1.0a.xsd
```

#### 7.3.1.13 Elements or attributes

Schemas shall be designed so that elements are the main holders of information content in the XML instances. Attributes are more suited to holding ancillary metadata — simple items providing more information about the element content. If the element containing the attribute has element content, any attributes shall normally apply to all the descendant elements but are not mandatory.

NOTE Unlike elements, attributes cannot hold structured data. For this reason, elements are preferred as the principal holders of information content. However, allowing the use of attributes to hold metadata about an element’s content (for example, the format of a date, a unit of measure or the identification of a value set) can make an instance document simpler and easier to understand. If an attribute were allowed to qualify other attributes, there could arise cases with multiple attributes where it would not be clear whether an attribute was qualifying the element or one of several attributes. Therefore, attributes are not allowed to qualify other attributes.

#### 7.3.1.14 Data type or element declarations

A component may be defined as a data type if either:

— it is to be used with different element names in different contexts; or

— it is expected that further data types will be derived from it.

A component shall be defined as an element if:

— there is no intention to derive new components from it; and

— the element is to be used with its name unchanged.

NOTE There are many circumstances in which an element should be used with its name unchanged. For example, if a “Unique LOCATION” always has the name “Unique LOCATION”, its semantics will be known and two systems using the same element will be known to be using the same definition. It is therefore possible to build a dictionary of element names with known interoperable semantics. However, there are other circumstances where it is not appropriate to allocate a name to an element at the time a Schema is developed. For example, an address could have several meanings and so be used with different names. An address should therefore be defined as a global data type. The other circumstance for choosing between an element and a data type to define a component is if there is an intention to derive other components from it. By only using data types in this instance, we simplify understanding of Schemas by only having a single

inheritance mechanism and avoiding use of *xs:redefine* for this purpose. In some cases in a Schema, it is appropriate to define both a data type and an element. The element is then available with known fixed semantics for reuse and the data type available for appropriate modification.

#### 7.3.1.15 Use of complex types

In order to enable future reuse of compound elements, or to allow for the use of inheritance in its own definition, complex structures shall be used for elements that represent all significant (that is semantically discrete) objects, including request and response themselves.

#### 7.3.1.16 Global definitions

Schema documents should make available globally those component definitions that are either:

— re-used within the Schema; or

— to be made available for re-use in other Schemas (import, include).

NOTE The main reason for this approach is to limit the effect of change. By keeping component definitions local, it is easy to control who else uses these definitions and so limit the impact of change.

#### 7.3.1.17 Use of documentation element and language attribute

When documenting a Schema, the documentation element shall be used rather than XML comments.

NOTE The “documentation” element contains a human readable annotation to a Schema. Using “documentation” element, “source” attribute supplementing outer documentation information and “*xml:lang”* attribute support multi-natural language description.

### 7.3.1.18 Containing schema header in every schema

The Schema header shall be defined with reference to Table 2.

**Table 2 — Schema header item**

| Header Item | | Description |
|---|---|---|
| **General information** | | |
| | Schema name | Described by ID attribute |
| | Functional area | Classified functional area name by ISO 14813 or common area |
| | Description | Plain text description of the type of information described by the Schema |
| | Version number | Version number of the released or draft Schema |
| | Current status | (Released/ Draft) |
| | Standard title | Related standard title of the Schema, if extant |
| | Standard number | Related standard number of the Schema, if extant |
| | Detail explanation IRI | Detail explanation IRI of the Schema, if extant |
| | Namespace | targetNamespace of the schema (See 7.3.1.11) |
| | Schema location | URL of the Schema (See 7.3.1.12) |
| | Validated by | Schema validation software name |
| | UML location | UML (CLASS Diagram) of the Schema, if extant |
| | DR/DD information | DR/DD Information of the Schema, if extant |
| | Code table IRI | Code table IRI of the Schema, if extant |
| | Enumeration IRI | Enumeration IRI, if extant |
| **Change history** | | |
| | Version number | Release or draft version number |
| | Date | Version number change date |
| | Description of change | Plain text description |
| **Other Schemas imported** | | |
| | Schema name | Imported Schema names, if extant |
| | Description | Plain text description of the type of information imported by the Schema |
| | Namespace | Namespace of the imported Schema |
| | Schema location | Schema location of the imported Schema |
| **Other Schemas included** | | |
| | Schema name | Included Schema names |
| | Description | Plain text description of the type of information included by the Schema |
| | Schema location | Schema location of the included schema |
| **Point of contact** | | |
| | Name | Name of person to contact with questions regarding the Schema |
| | e-mail address | E-mail address to contact with questions regarding the Schema |

NOTE 1 By adhering a header to the schema, the schema becomes an interoperability tool and promotes reuse of the schema, because analysts can read it and understand the meaning and derivation of various XML components. It also expected to increase the maintainability of the system.

NOTE 2 Using Annex C Schema header template, 7.3.1.17 conformant header can be authored.

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema ……>
<xs:annotation>
    <xs:documentation xml:lang="en">
<HeaderStart>********** ********** ********** ********** **********</HeaderStart>
            <generalInformation>
                <schemaName>iso standard 14532 schema header</schemaName>
                <functionalArea>ITS common</functionalArea>
                            <description>This schema is a template for creating ISO
            24531conformant schema header </description>
                <versionNumber>1.0a</versionNumber>
                <currentStatus>Draft</currentStatus>
                <standardTitle>Using XML in ITS standard, data registries and data
dictionaries</standardTitle>
                <standardNumber>ISO 24531</standardNumber>
    <detailExplanationIRI>http://www.example.com/iso/standard/24531/schema/1/v1.0a.xsd</de
tailExplanationIRI>
    <nameSpace>http://www.example.com/iso/standard/24531/schema/1</nameSpace>
                <validatedBy>ABC Schema editor v6.0</validatedBy>
    <UMLLocation>http://www.example.com/iso/standard/schema/1/v1.0a.cls</UMLLocation>
                <DR_DDInformation>all data concept, including this schema are registered to
the ISO 14817 data registry</DR_DDInformation>
                <codeTableIRI>None</codeTableIRI>
                <enumerationIRI>None</enumerationIRI>
            </generalInformation>
            <changeHistory>
                <change>None</change>
            </changeHistory>
            <otherSchemasImported>
                <importedSchema>None</importedSchema>
            </otherSchemasImported>
            <otherSchemasIncluded>
                <schemaNames>iso_standard_schema_2</schemaNames>
                <description>general information template</description>
    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/2/v1.0a.xsd</schemaL
ocation>                <schemaNames>iso_standard_schema_3</schemaNames>
                <description>change history template</description>
    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/3/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_4</schemaNames>
                <description>other schema imported template</description>
    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/4/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_5</schemaNames>
                <description>other schema included template</description>
    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/5/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_6</schemaNames>
                <description>contact information template</description>
    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/6/v1.0a.xsd</schemaL
ocation>
            </otherSchemasIncluded>
            <otherSchemasIncluded/>
            <pointOfContact>
                <Name>A B</Name>
                <mail>a_b@its.com</mail>
                <Name>X Y</Name>
                <mail>x.y@tc20</mail>
            </pointOfContact>
<HeaderEnd>********** ********** ********** ********** **********</HeaderEnd>
       </xs:documentation>
    </xs:annotation>
…
</schema>
```

#### 7.3.1.19 External table reference

Where an external table is provided, reference IRI shall be included.

NOTE Because a code table is often extended, reference to a maintained external table means that there is no need for revision with each change to the code table.

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<wind xmlns:xlink="http://www.example.com/1999/xlink"
  xlink:type="simple"
  xlink:title="wind direction code table"
xlink:href="http://www.example.com>
    NNW
</wind>
```

NOTE *href* address is hypothetical.

#### 7.3.1.20 *xs:redefine*

*xsd:redefine* shall not be used.

NOTE This is to avoid pervasive side-effects in reused components, and to increase clarity and readability.

#### 7.3.1.21 Accessibility of the Schema file

The Schema file shall be accessible while an application is in use.

When an instance document is input, the Schema syntax is designed such that validation is performed regardless of whether the processor references the Schema or not. Ordinarily, however, a processor references the Schema in performing the validation process, and it is thought that such a method shall normally be provided. The basis for doing that is the Schema files. Accordingly, the Schema file shall be accessible so long as the application is in use.

### 7.3.2 Rules for XML Schema component

#### 7.3.2.1 Adoption of consistent naming conventions

Consistent naming conventions should be adopted concerning the names of XML elements/attributes defined by users.

NOTE Adopting consistent naming conventions is desirable for enhancing schema readability. The following type of method exists with respect to naming conventions:

— carType (camel case convention)

— Car - Type

— car_type

— car.type, etc.

The camel case convention is the naming convention adopted for the W3C XML Schema language (e.g. *complexType*, *maxOccurs*, etc.). Excluding cases where Schemas have already been created, use of the camel case convention is desirable, considering its familiarity to users, among other factors.

#### 7.3.2.2 Avoidance of name collision in one namespace

Within any namespace, a name shall be used only once and shall not be repeated even if it is a different type of data.

In a Schema, elements, attributes and data types are managed in different symbol spaces. Therefore, even if same names are used in an element and attribute, it is legal in XML (see next example).

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema/1/v1.0">
    <xs:attribute name="name" type="xs:Name"/>
    <xs:element name="name" type="xs:Name"/>
</xs:schema>
```

Using same names, however, prevents readability of the Schema and causes problems in ISO 14817 DR/DD data concept management. Thus, different names shall be used within any one namespace.

#### 7.3.2.3 Differentiation between simple data type names, complex data type names and element names

The names of complex data types shall end with the text string "*Structure*".

The name of simple data types shall end with the text string "*Type*".

#### 7.3.2.4 Revision of version attribute in Schema elements when the Schema is revised

When changing the Schema, the version number shall be changed and information of the new schema version shall be registered to the ISO 14817 data registry and/or data dictionary.

Schema changes need to be reported to applications in order to ensure operation without any contradictions occurring.

#### 7.3.2.5 Description of annotations and comments

Where necessary, annotations and comments shall be provided.



# 8 Rules for registration and management of XML Schema constructs in data registry (DR) and/or data dictionaries (DDs)

## 8.1 Objectives of Schema constructs registration and management

Schemas are the basis for the development of XML applications. One task that takes a lot of time in the development of interoperable XML applications is the determination of the Schema. ITS applications interface with other ITS sub-applications, extensibility shall be considered so that once a program has been written, and the Schema does not have to be revised. Reusing existing Schema enhances productivity, interoperability and system lifetime extension.

## 8.2 Why use ISO 14817 data registry/data dictionary (DR/DD)?

OMG XMI 2.0, ISO/IEC 8825-4, and ISO/IEC 8825-5 eliminate barriers of languages on the standard bases, so one language information can be reused by another language. No language is superior for every situation and a suitable language should be selected by application circumstances. Wherever possible, the use of a common, sector-wide DD/DR is recommended.

NOTE 1 If DR/DD is created corresponding to each language, it is hard to reuse other language information. Therefore, this International Standard promotes an information portal site for TC 204 applications with a single DR/DD.

NOTE 2 It is important to understand that ISO 14817 concepts can be mapped to a number of different description languages, so that it is possible to have an ISO 14817 registry with a focus on UML, or an ISO 14817 registry with a focus on XML Schema.

NOTE 3 In order to conform with ISO 14813-6 and ISO 14817, DR/DD provides ASN.1-oriented meta-attributes, but it is well abstracted to use Schema registry/dictionary independent of language. In addition, ISO 14817 has a meta-attribute URL (IRI) supporting to refer to the outer source. This enables standard developer/schema designer/application developer to see other language information and if necessary the stakeholder can transform it to her/his developing language.

NOTE 4 Schema (ASN.1 Module and XMI representation) is not only an information model representation but also a system developing start point. Using Schema files, pointed by DR/DD's URL (IRI), a system developer could use tools directly.

Standard developer/schema designer/application developer sometimes need information model at a glance like UML diagrams, DR/DD might not fit this purpose. This International Standard mandates to provide a Schema header for this purpose (See 7.3.1.18) when describing the Schema.

Figure 2 depicts relationships between XML Schema constructs and ISO 14817 DR/DD.

NOTE 5 ASN.1<->XML transformation is done if necessary.

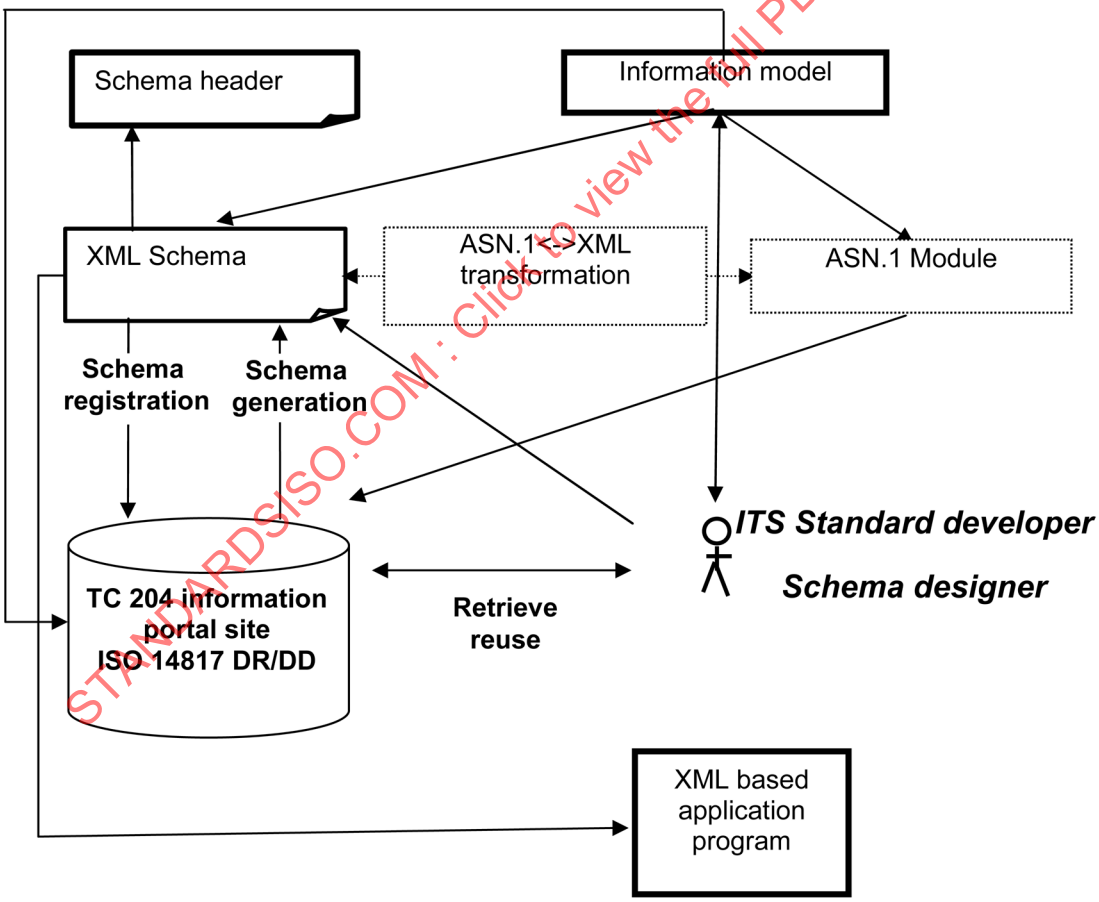





**Figure 2 — Relationships of XML Schema constructs and ISO 14817 DR/DD**

NOTE 6 Information model transformation to ISO 24531 conformant XML Schema sometimes need a profile (see Annex E).

## 8.3 ISO 14817 registration/management of Schema constructs

Schema has many possible constructs. This subclause designates which Schema constructs shall be registered in ISO 14817 DR/DD (together with its rationale).

Figure 3 depicts the selection scheme of the registering constructs.

NOTE This scheme is made from a practical (rather than a theoretical) perspective.

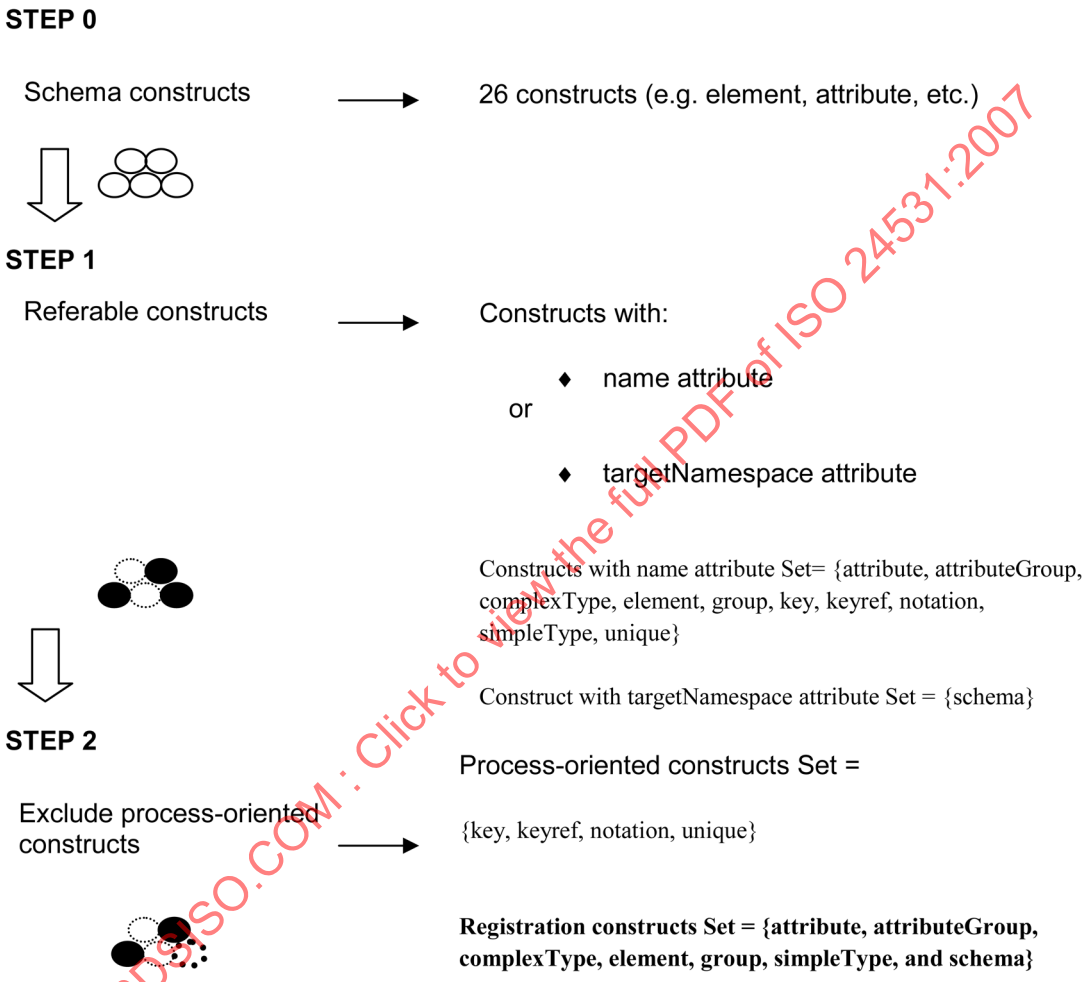



**Figure 3 — Registration/management target Schema constructs**



**STEP 1**

From the 26 possible Schema constructs, one is selected; this provides the construct with a name or *targetNamespace* attribute.

**STEP 2**

From STEP 1 set, this International Standard eliminates process-oriented constructs. The reason to eliminate process-oriented constructs are rarely referenced from outer Schema (see next EXAMPLE).

NOTE As seen in the example below, the name attribute of *xs:notation* is only used to define *graphicalFormat*. From the viewpoint of "necessary and sufficient" of DR/DD registration, *xs:notation* is excluded.

EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:notation name="jpeg" public="image/jpeg" system="file:///user/bin/xv"/>
    <xs:notation name="gif" public="imag/gif" system="file:///user/bin/xv"/>
    <xs:simpleType name="graphicalFormat">
        <xs:restriction base="xs:NOTATION">
            <xs:enumeration value="jpeg"/>
            <xs:enumeration value="gif"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

## 8.4 Schema constructs mapping to the ISO 14818 constructs

In this subclause, mapping scheme of Schema constructs to ISO 14817 constructs is described. The concrete mapping rule of the Schema construct is defined in subclause 8.5.

Most Schema construct properties can map directly to the ISO 14817 construct meta-attribute. In the occasional cases where this is not possible, the ISO 14817 URI meta-attribute shall be set to the Schema location, so stakeholders can directly see the declaration (see Figure 4).

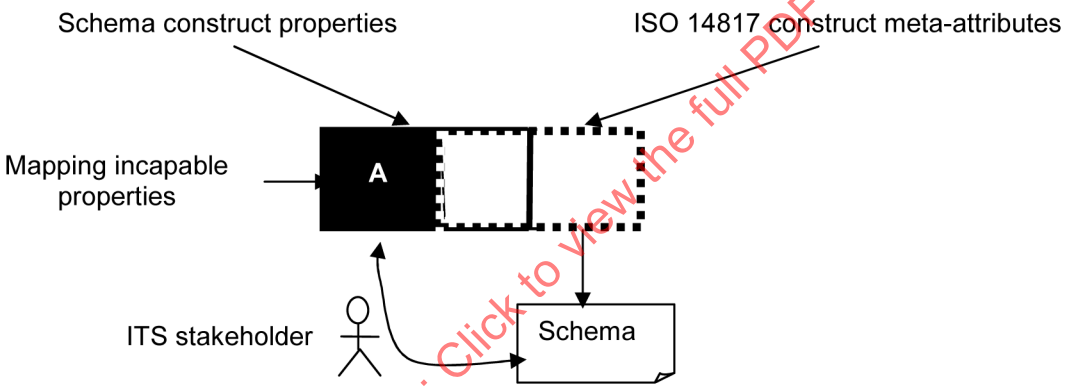



**Figure 4 — URL mapping**

NOTE Anonymous type treatment.

Listing 8-1 is an example of anonymous *complexType*.

EXAMPLE Listing 8-1 Anonymous type

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="data">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:float">
                    <xs:attribute name="measureTime" type="xs:date"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

The *xs:complexType* has no name attribute because this *complexType* is used only for the declaration of “data”. This anonymous style is the Schema developer’s intension. Thus, this *xs:complexType* need not register. If the stakeholder wants this *xs:complexType* to reuse, no need to register, method of listing 8-2 can be applied.

EXAMPLE Listing 8-2 Global declaration

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:complexType name="dataStructure" >
            <xs:simpleContent>
                <xs:extension base="xs:float">
                    <xs:attribute name="measureTime" type="xs:date"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
        <xs:element name="data" type="dataStructure" />
</xs:schema>
```

Schema constructs mapping to ISO 14817 constructs are summarized in Table 3.

**Table 3 — Schema constructs mapping scheme for ISO 14817 constructs**

| Similarity comparison table | | | | Schema constructs -> ISO 14817 construct mapping |
|---|---|---|---|---|
| **XML type** | **Schema constructs** | **ISO 14817 constructs** | **(UML constructs)** | |
| Simple type | xs:element<br>xs:attribute | data element | attribute datatype | simple type (xs:element/xs:attribute) -> data element |
| | xs:simpleType | property | datatype | xs:simpleType -> property |
| Complex type | xs:element<br>xs:group<br>xs:attributeGroup<br>xs:complexType | object class | class | complex type -> object class |
| Schema type | xs:schema | object class | class | schema type -> object class |
| | (web services WSDL) | interface dialog | (message in sequence diagram) | |
| | (xs:nameSpace) | | package | |



## 8.5 Registration and management rules

### 8.5.1 Referencing of DR or DD when creating Schemas

When creating Schemas, a check shall be made to confirm whether or not the Schema elements are already registered in the ISO 14817 data registry and data dictionary.

NOTE This procedure is necessary in order to enable portal sites to keep the vocabulary.

### 8.5.2 Registering items not registered in DR/DD

If an XML component has not been registered, the Schema and/or elements that have name attributes shall be registered in the appropriate DR/DD registration of Schemas.

### 8.5.3 Registering Schemas

Schemas shall be registered as an ISO 14817 object class. When creating new Schemas or changing their version number, the Schema shall be registered with its new URL (IRI). The content of a new URL (IRI) is its *schemaLocation*.

### 8.5.4 Use Schema ID attribute as ISO 14817 *ObjectClassTerm*

All descriptive names shall be unique.

NOTE In ISO 14817, the descriptive name is the key to registered data concept, so it must be unique. Using the Schema ID attribute and unique name (see 7.3.2.2) uniqueness of the data concept is assured.

### 8.5.5 Registration of Schema simple type as data element

Simple type Schema constructs shall be registered as ISO 14817 data element types.

NOTE Simple type does not include what are called attributes or child elements in XML. Accordingly, they are registered as ISO 14817 data elements.

### 8.5.6 Registration of Schema complex type as ISO 14817 object class

Schema complex type elements shall be registered as ISO 14817 object class.

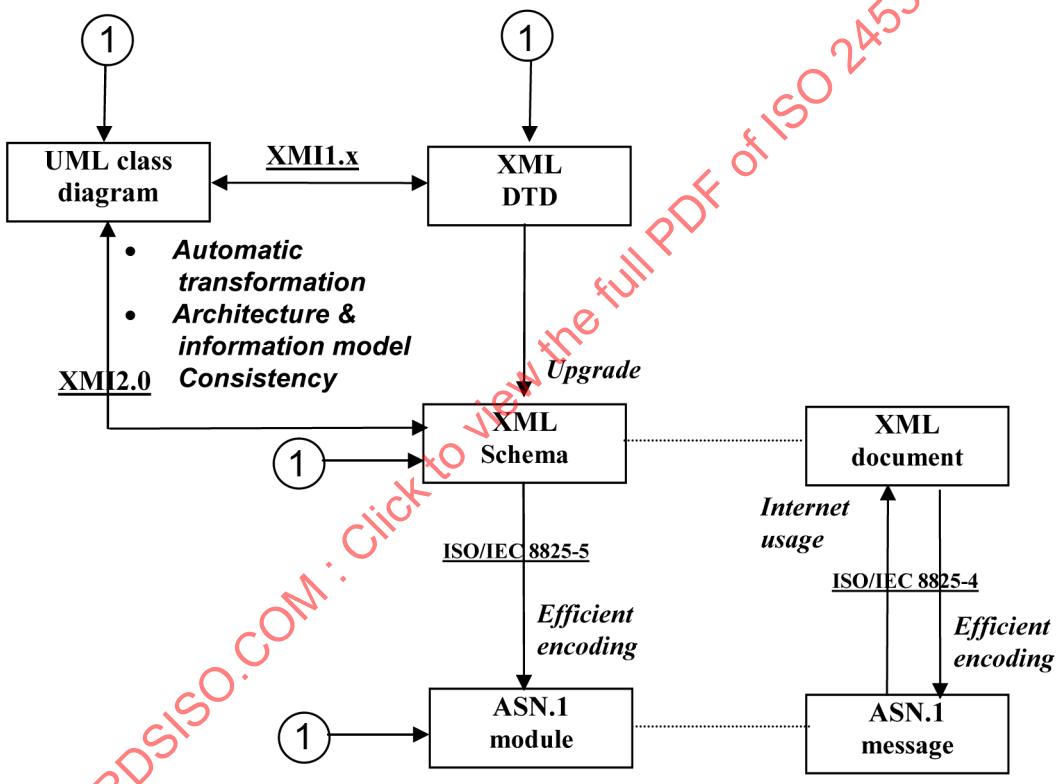
NOTE Schema complex type includes child elements or attributes. Accordingly, they need to be registered as an ISO 14817 *objectClass*.

# Annex A
(informative)

# Model/document transformation

## A.1 Model/document transformation among XML, UML and ASN.1

UML, XML Schema, XML DTD and ASN.1 have their own target area of usage, and no language is superior for all domains. Mastering every language is impractical, so it is helpful to use standardized language transformation supported by software tools. Figure A.1 shows various language transformation standards.




**Key**

1 designing starting point

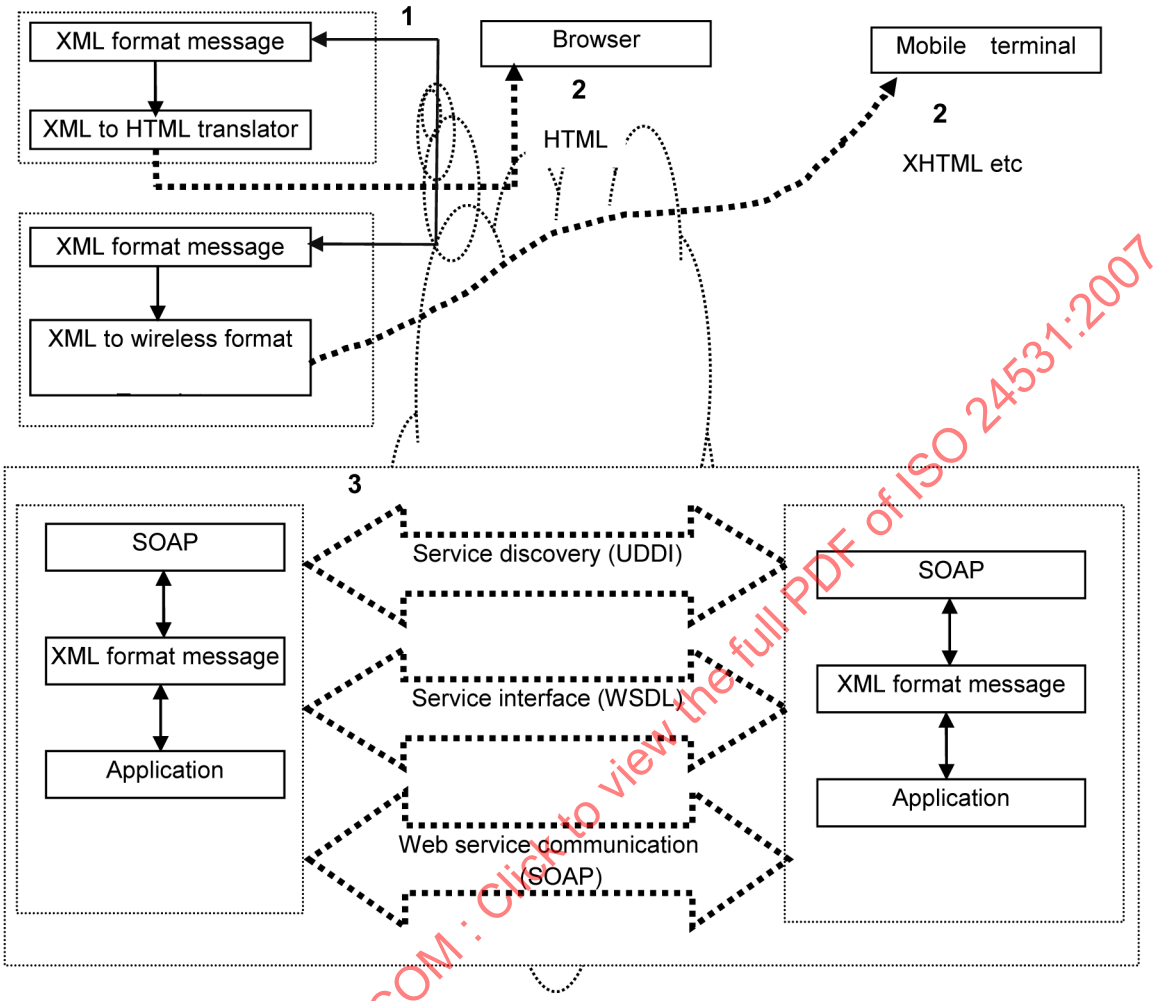
NOTE Typical objectives in italics; related standards underlined.

**Figure A.1 — Language transformation standards and typical objectives of transformation**

Mutual transformation yields great profits for increasing productivity through effective reuse of resources. In Annex E, UML and XML transformation is described, and in Annex G, ASN.1 and XML transformation is described.

## A.2 XML applications in ITS

Figure A.2 depicts representative XML applications in ITS.




**Key**

1 message interface
2 internet browsing
3 web services

**Figure A.2 — XML application in ITS**



### A.2.1 Message interchange

XML does not depend on hardware and operating systems. It is frequently used as a middleware in heterogeneous circumstances. Because XML Schema gives more rigorous validation of the message structure than DTD, usage of the Schema is required. Because an XML processor validates the message, developing of new application program for checking data is not necessary. Changing the interface might cause relevant program modification, so it is required to make expandable Schema. To achieve this requirement, the design of Schema and description are separate processes. Defining the message structure using UML is often used, because it enables the interface visibly and eases understanding among the stakeholders. In Annex E, automatic transformation from UML package/class diagram to XML Schema is described. If a national ITS architecture is described by UML with sufficient detail of granularity, it is possible to create XML Schema automatically, and this enables strict reconciliation with the architecture, and reverse engineering maintenance of the architecture is possible.

By cooperating with other ITS applications and ITS adjacent applications (e.g. e-commerce, e-government, etc.), the effect is increased. Messages using validated Schema make application cooperation possible. If relevant XML Schema already exist, this International Standard recommends such reuse because it brings interoperability between applications. Once the Schema is registered, it can be reused by the “include” or “import” mechanisms of XML. A UML class diagram makes it easier to find reusable Schema.

NOTE XML message interchange is used for various ITS applications. For example, message interchange between traffic control centres, weather information, event information, telemetric services, and operational data exchange for commercial vehicles, are typical usages of XML messages.

### A.2.2 Internet browsing

XML documents are easily transformed to HTML or XHTML by using the XML application interface “*DOM*” or “*XSLT*”. A typical ITS application of Internet browsing is a traveller information system.

### A.2.3 Web services

Web services are anticipated to be one of the main applications for the use of XML. The W3C defines web services as follows: “*A web service is a software system identified by a URI (RFC 2396), whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols.*”

The term “web service” describes a standardized way of integrating web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to make message, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, web services allow organizations to communicate data without intimate knowledge of each other’s IT systems behind the firewall.

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, web services are not tied to any one operating system or programming language. Compared to EDI or CORBA, it is said to be easier, and flexible coordination is possible. In ITS, much application including traveller information, as well as fundamental data exchange Web services, could be applied.

# Annex B
(informative)

# ISO/TC 204 representation of IRI (URI) and/or ID related constructs

Table B.1 summarizes representation of IRI and/or version ID related constructs.

**Table B.1 — ISO 24531-related representation of IRI and/or version ID constructs**

| XML construct | Requirement | ISO TC 204 representation rule | Example |
|---|---|---|---|
| xs:anyURI | physically accessible by populated softwares | http_URI | http://www.example.com/iso/standatd/24531/schema/1/v1.0 |
| xs:id | globally unique | XML OID (delimiter "_") | iso_standard_24531_schema_1_v_1_._0 |
| xs:targetNamespace | globally unique | " http://"host"/XML OID (delimiter "/") " | http://www.example.com/iso/standard/24531/schema/1/v1.0 |
| xs:schema Location | globally unique<br>physically accessible by populated softwares<br>1-to-1 mapping to the Schema version | xs:targetNamespace" .xsd" | http://www.example.com/iso/standard/24531/schema/1/v1.0a/xsd |

NOTE 1 The host in the example is virtual. TC 204 determines the host.

NOTE 2 In an XML instance, the document version of Schema can be identified by the *xsi:schemaLocation* attribute.

# Annex C
(informative)

# Schema header template

The normative part of this International Standard specifies that a header and mandatory items as described in 7.3.1.18 be attached to a Schema. To make a Schema header easily and correctly, this annex shows how an example Schema header template is presented (in Listing C-1 to C-6).

**Listing C-1 Schema header template (main part)**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="qualified" elementFormDefault="qualified"
    id="iso_standard_24531_schema_1_v_1_._0_a"
targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    version="1.0a" xmlms="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:annotation>
        <xs:documentation xml:lang="en">
<HeaderStart>********** ********** ********** ********** **********</HeaderStart>
    <generalInformation>
        <schemaName>iso standard 14532 schema header</schemaName>
        <functionalArea>ITS common</functionalArea>
        <description>This schema is a template for creating ISO 24531conformant schema
header </description>
        <versionNumber>1.0a</versionNumber>
        <currentStatus>Dtaft</currentStatus>
        <standardTitle>Using XML in ITS standard, data registries and data
dictionaries</standardTitle>
        <standardNumber>ISO 24531</standardNumber>
    <detailExplanationIRI>http://www.example.com/iso/standard/24531/schema/1/v1.0a.xsd</de
tailExplanationIRI>
                <nameSpace>http://www.example.com/iso/standard/24531/schema/1</nameSpace>
                <validatedBy>ABC Sxhema editor v6.0</validatedBy>

    <UMLLocation>http://www.example.com/iso/standard/schema/1/v1.0a.cls</UMLLocation>
                <DR_DDInformation>all data concept, including this schema are registered to
the ISO
                    14817 data registry</DR_DDInformation>
                <codeTableIRI>None</codeTableIRI>
                <enemerationIRI>None</enemerationIRI>
            </generalInformation>
            <changeHistory>
                <change>None</change>
            </changeHistory>
            <otherSchemasImported>
                <importedSchema>None</importedSchema>
            </otherSchemasImported>
            <otherSchemasIncluded>
                <schemaNames>iso_standard_schema_2</schemaNames>
                <description>general information template</description>

    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/2/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_3</schemaNames>
                <description>change histry template</description>

    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/3/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_4</schemaNames>
                <description>other shaema inported template</description>
```

```
    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/4/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_5</schemaNames>
                <description>other schema included template</description>

    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/5/v1.0a.xsd</schemaL
ocation>
                <schemaNames>iso_standard_schema_6</schemaNames>
                <description>contact information template</description>

    <schemaLocation>http://www.iso.standard/iso/standard/23531/schema/6/v1.0a.xsd</schemaL
ocation>
            </otherSchemasIncluded>
            <otherSchemasIncluded/>
            <pointOfContact>
                <Name>A B</Name>
                <mail>a_b@its.com</mail>
                <Name>X Y</Name>
                <mail>x.y@tc20</mail>
            </pointOfContact>
<HeaderEnd>********** ********** ********** ********** **********</HeaderEnd>
    </xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="file:/I:/iso/standard/24531/schema/2/v1.0a.xsd"/>
    <xs:include schemaLocation="file:/I:/iso/standard/24531/schema/3/v1.0a.xsd"/>
    <xs:include schemaLocation="file:/I:/iso/standard/24531/schema/4/v1.0a.xsd"/>
    <xs:include schemaLocation="file:/I:/iso/standard/24531/schema/5/v1.0a.xsd"/>
    <xs:include schemaLocation="file:/I:/iso/standard/24531/schema/6/v1.0a.xsd"/>
    <xs:element id="iso_standard_24531_schema_1_dataConcept_1" name="schemaHeder">
    <xs:complexType>
            <xs:sequence>
                <xs:element fixed="********** ********** ********** ********** **********"
                    name="HeaderStart" type="xs:string"/>
                <xs:element name="generalInformation"/>
                <xs:element name="changeHistory"/>
                <xs:element name="otherSchemasImported"/>
                <xs:element name="otherSchemasIncluded"/>
                <xs:element name="pointOfContact"/>
                <xs:element fixed="********** ********** ********** ********** **********"
                    name="HeaderEnd" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Listing C-2 General Information part (included in main Schema)**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="iso_standard_24531_schema_2_v_1_._0_a" version="1.0a"
    targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    xmlns="http://www.example.com/iso/standard/24531/schema/1"
elementFormDefault="qualified"
    attributeFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:annotation>
        <xs:documentation>  Schema  header  is  omittedfor  clarifying  the  example
</xs:documentation>
    </xs:annotation>
    <xs:element id="iso_standard_24531_schema_2_dataConcept_1" name="generalInformation"/>
    <xs:complexType                          id="iso_standard_24531_schema_2_dataConcept_2"
name="generalInformationStructure">
        <xs:sequence>
            <xs:element id="iso_standard_24531_schema_2_dataComcept_3" name="schemaName"
                type="xs:string"/>
            <xs:element                          id="iso_standard_24531_schema_2_dataConcept_4"
name="functionalArea"
```

```
                type="xs:string"/>
            <xs:element id="iso_standard_24531_shema_2_dataConcept_5" name="description"
                type="xs:string"/>
            <xs:element id="iso_standard_24531_schema_2_dataConcept_6" name="versionNumber"
                type="versionNumberStructure"/>
            <xs:element id="iso_standard_24531_schema_2_dataConcept_7" name="currentStatus"
                type="currentStatusType"/>
            <xs:element    id="iso_standard_schema_2_dataConcept_8"    name="standardTitle"
type="xs:string"/>
            <xs:element                        id="iso_standard_24531_schema_2_dataConcept_9"
name="standardNumber"
                type="xs:string"/>
            <xs:element                        id="iso_standard_24531_schema_2_dataConcept_10"
name="detailExplanationIRI"
                type="IRIType"/>
            <xs:element   id="iso_standard_24531_schema_2_dataConcept_11"   name="nameSpace"
type="IRIType"/>
            <xs:element                        id="iso_standard_24531_schema_2_dataConcept_12"
name="schemaLocation"
                type="xs:anyURI"/>
            <xs:element id="iso_standard_24531_schema_2_dataConcept_13" name="validatedBy"
                type="xs:string"/>
            <xs:element id="iso_standard_24531_schema_2_dataConcept_14" name="UMLLocation"
                type="URIType"/>
            <xs:element                        id="iso_standard_24531_schema_2_dataConcept_15"
name="DR_DDInformation"
                type="xs:string"/>
            <xs:element id="iso_standard_24531_schema_2_dataConcept_16" name="codeTableIRI"
                type="IRIType"/>
            <xs:element                        id="iso_standard_24531_schema_2_dataConcept_17"
name="enemerationIRI"
                type="IRIType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="versionNumberStructure">
        <xs:choice>
            <xs:element name="draftVersionNumber">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="d{1}\.\d{1}[a-z]{1}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="releasedVersionNumber">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="d{1}\.\d{1}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
    <xs:simpleType name="currentStatusType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Draft"/>
            <xs:enumeration value="Released"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="IRIType">
        <xs:choice>
            <xs:element fixed="None" name="other" type="xs:string"/>
            <xs:element name="IRI" type="xs:anyURI"/>
        </xs:choice>
    </xs:complexType>
</xs:schema>
```

### Listing C-3 Change History part (included in main Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="iso_standard_24531_schema_3_v_1_._0_a" version="1.0a"
    targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    xmlns="http://www.example.com/iso/standard/24531/schema/1"
attributeFormDefault="qualified"
    elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="file:/I:/iso/standard/24531/schema/2/v1.0a.xsd"/>
    <xs:element id="iso_standard_24531_schema_3_dataConcept_1" name="changeHistory"
        type="changeHistoryStructure"/>
    <xs:complexType                              id="iso_standard_24531_schema_3_dataConcept2"
name="changeHistoryStructure">
        <xs:choice>
            <xs:element fixed="None " name="change" type="xs:string"/>
            <xs:element name="history" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element id="iso_standard_24531_schema_2_dataConcept_3"
                            name="versionNumberStructure"/>
                        <xs:element            id="iso_standard_24531_schema_3dataConcept_4"
name="Date"
                            type="xs:date"/>
                        <xs:element id="iso_standard_24531_schema_3_dataConcept_5"
                            name="discriptionOfChange" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
</xs:schema>
```

### Listing C-4 OtherSchemasImported part (included in main Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="iso_standard_24531_schema_4_v_1_._0_a" version="1.0a"
    targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    xmlns="http://www.example.com/iso/standard/24531/schema/1"
attributeFormDefault="qualified"
    elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element id="iso_standard_24531_schema_4_dataConcept_1" name="otherSchemasImported"
        type="otherSchemasImportedStructure"/>
    <xs:complexType id="iso_standard_24531_schema_4_dataConcept_2"
        name="otherSchemasImportedStructure">
        <xs:choice>
            <xs:element fixed="None" name="importedSchema" type="xs:string"/>
            <xs:element name="importedList" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element          id="iso_standard_24531_schema_4_dataConcept_3"
name="schemaNames"/>
                        <xs:element          id="iso_standard_24531_schema_4_dataConcept_4"
name="Description"
                            type="xs:string"/>
                        <xs:element          id="iso_standard_24531_schema_4_dataConcept_5"
name="nameSpace"
                            type="xs:anyURI"/>
                        <xs:element name="schemaLocation" type="xs:anyURI"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
</xs:schema>
```

### Listing C-5 OtherSchemaIncluded part (included in main Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="iso_standard_24531_schema_5_v_1_._0_a" version="1.0a"
    targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    xmlns="http://www.example.com/iso/standard/24531/schema/1/v1/0a"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    attributeFormDefault="qualified">
    <xs:element id="iso_standard_24531_schema_5_dataConcept_1" name="otherSchemasIncluded"
        type="otherSchemaIncludedStructure"/>
    <xs:complexType name="otherSchemaIncludedStructure">
        <xs:choice>
            <xs:element     fixed="None"     id="iso_standard_24531_schema_5_dataConcept_3"
name="including"
                type="xs:string"/>
            <xs:sequence maxOccurs="unbounded">
                <xs:element                  id="iso_standard_24531_schema_5_dataConcept_4"
name="schemaNames"
                    type="xs:string"/>
                <xs:element                  id="iso_standard_24531_schema_5_dataConcept_5"
name="Description"
                    type="xs:string"/>
                <xs:element                  id="iso_standard_24531_schema_5_dataConcept_6"
name="schemaLocation"
                    type="xs:anyURI"/>
            </xs:sequence>
        </xs:choice>
    </xs:complexType>
</xs:schema>
```

### Listing C-6 PointOfContact part (included in main Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="iso_standard_24531_schema_6_v_1_._0_a"
    targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0a"
    xmlns="http://www.example.com/iso/standard/24531/schema/1/v1/0a"
elementFormDefault="qualified"
    attributeFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element                  id="iso_standard_24531_schema_6_dataConcept_1_v_1_._0_a"
name="pointOfContact"/>
    <xs:complexType                                          name="pointOfContactStructure"
id="iso_standard_24531_schema_6_dataConcept_2">
        <xs:sequence maxOccurs="unbounded">
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="mail" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

# Annex D
(informative)

# Example of registering an XML construct

## D.1 General

As described in Clause 8, the ITS data registry/data dictionary is a “portal site” for ITS information, that promotes the reuse of data concepts. Reuse of data concepts can increase productivity and provide added value to the ITS business by improving portability of data between systems. It is therefore necessary when creating a new data concept to register this with an ISO 14817-compliant data registry/data dictionary. This annex gives an example of how to register a created XML schema constructs in the DR/DD. Figure D.1 shows a package diagram forming the basis for creating the schema to be registered.

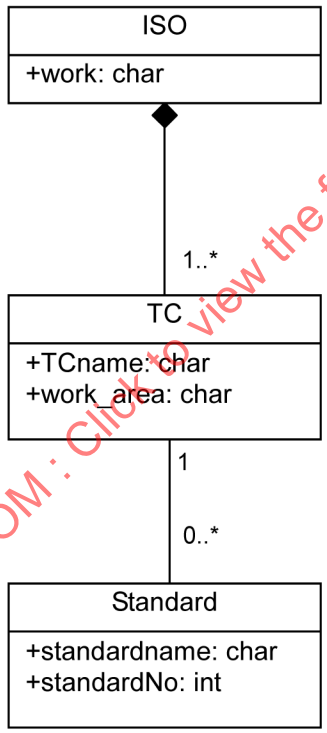



**Figure D.1 — Package diagram**

The following is the corresponding XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="iso_standard_24531_schema_1_v_1_._0"
    targetNamespace="http://www.example.com/iso/standard/24531/schema/1/v1.0"
    xmlns="http://www.example.com/iso/standard/24531/schema/1/v1.0"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="ISO" type="ISOStructure"/>
    <xs:complexType name="ISOStructure">
        <xs:sequence>
            <xs:element name="ISO.work" type="xs:string"/>
            <xs:element name="ISO.TCStructure">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" ref="TC"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="TC" type="TCStructure"/>
    <xs:complexType name="TCStructure">
        <xs:sequence>
            <xs:element name="TC.TCname" type="xs:string"/>
            <xs:element name="TC.work_area" type="xs:string"/>
            <xs:element name="TC.Standard">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element maxOccurs="unbounded" minOccurs="0" ref="Standard"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="Standard" type="StandardStructure"/>
    <xs:complexType name="StandardStructure">
        <xs:sequence>
            <xs:element name="Standard.standrdname" type="xs:string"/>
            <xs:element name="Standard.standardNo" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

When this schema is created, the following data construct must be registered if they have not already been registered in the DR/DD:

— the schema itself;

— attribute;

— attributeGroup;

— complexType;

— elements;

— group.

How Schema constructs mapping rule to the ISO 14817 data concept is shown in Table D.1.

**Table D.1 — Schema construct mapping rule to the ISO 14817 data concept**

| | | ISO 14817 DR/DD data concept | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Interface dialog** | **Message** | **Data frame** | **Data element** | **Object class** | **Data element concept** | **Property** | **Value domain** |
| Schema construct | Schema | | | | | ○ | | | |
| | attribute | | | | ○ | | | | |
| | attributeGroup | | | | | ○ | | | |
| | Element*2 | | | | ○ | ○ | | | |
| | group | | | | | ○ | | | |
| | simpleType | | | | ○ | | | | |
| | complexType | | | | | ○ | | | |

When an XML element is a simple type, it is mapped to the ISO 14817 data element; in the case of a complex type, it is mapped to an ISO 14817 object class.

## D.2 Schema registration example

A schema header describes the schema property, as shown in 7.3.1.18 and in Annex C. The schema itself, however, forms an interface between systems, and it is preferable to be able to be accessible directly to an *xsd* file. It is therefore the Schema URL that shall be written in an ISO 14817-compliant DR/DD. An example is shown in Table D.2.

**Table D.2 — Example of Schema registration**

| ISO 14817 meta-attribute | ISO 14817 registration value |
|---|---|
| Descriptive name | ISO-standard-24531-schema-1-v-1-.-0 |
| ASN.1 object identifier | {iso standard 24531 schema 1 } |
| Uniform resource locator | http://www.example.com/iso/standard/24531/schema/1/v1.0.xsd |
| Definition | Example of Schema type registration |
| Descriptive name context | ISO-24531 |
| Data concept type | Object class |
| International Standard | ISO 24531 |
| Abstract | False |
| Referenced data element | ISO-standard-24531-schema-1-v-1-0.ISO<br>ISO-standard-24531-schema-1-v-1-0.ISOStructure<br>ISO-standard-24531-schema-1-v-1-.-0.TC<br>ISO-standard-24531-schema-1-v-1-.-0.TCStructure<br>ISO-standard-24531-schema-1-v-1-.-0.Standard<br>ISO-standard-24531-schema-1-v-1-.-0.StandardStructure |

# Annex E
(informative)

# Example of automatic generation of an XML schema from UML

## E.1 General

This annex provides a UML profile for XML Schema (stereotype, tagged value and constraint set) and an example of how to automatically create a XML schema from a UML class diagram that conforms with the Schema requirement of this International Standard. Some software tools support UML class diagram transformation to XML Schema with user (/user group) defined profile.

## E.2 The reasons to describe UML Class Diagram <-> XML Schema automatic transformation

OMG's XMI standard gives production rules to transform a UML class model to a W3C Schema. This specification may be implemented by software vendors. Over 30 vendors have developed XMI1.x. Tools allow data modellers to start working a conceptual level in UML, obviating the tedious aspects of schema description based on W3C Schema syntax. In most cases, XML Schema is used for data exchange amongst distributed heterogeneous systems. One of the most time-consuming processes of system development is the definition of the interface message. Using UML graphical presentation enables collaboration with ITS business experts and system developers. A key condition for the success of the ITS application will depend on this type of collaboration. Creating reusable ITS content components is also important to enable rapid software development. Using UML helps to find reusable ITS content component easily. After completion of the modelling, software tools can generate XML Schema.

## E.3 UML profile for XML Schema

A UML profile for XML Schema is a set of stereotypes, tagged values and constraints that enables transformation from a UML package/class diagram to a W3C XML Schema. See Figure E.1.

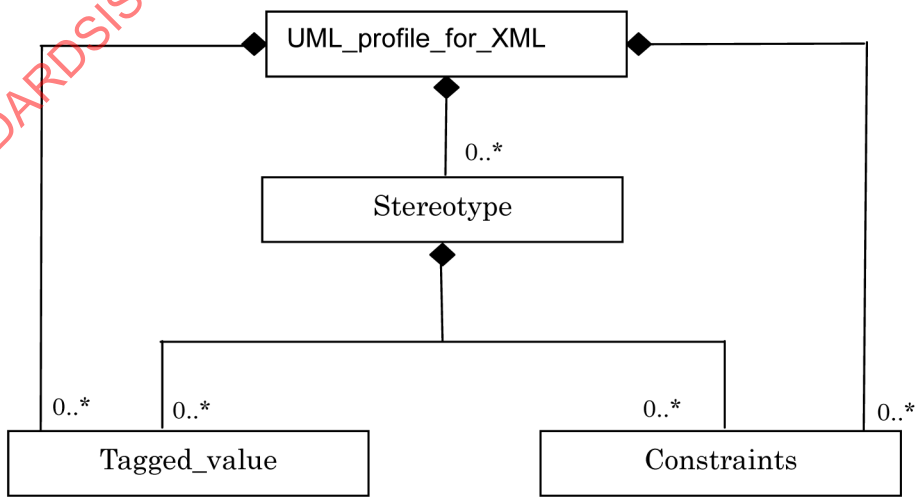



**Figure E.1 — UML profile for XML transformation**

There are two OMG standards in XMI.

XMI1.x (XMI1.0 and XMI1.1) defines DTD production rules from UML class/package diagrams and XMI2.0 defines XML Schema production rules from UML class diagram. The rules are formally defined in EBNF. Because this International Standard requires use of XML Schema, the application of XMI2.0 is required

XMI2.0 defines two transformation methods: default transformation and tailoring transformation. Default transformation generates valid Schema, but it might cause inconformity of the Schema to this International Standard [Figure E.2 a)]. XMI2.0 also gives tagged value set for tailoring Schema [Figure E.2 b)], but it is not sufficient to create conformant Schema to this International Standard. Thus, output Schema can be transformed using XML editor to make this Schema standard conformant. Another approach is extension of XMI2.0 tagged values to generate this standard conformant Schema. In a software developing or maintenance process, Schema to UML are very important. Figure E.2 also shows reverse process of Schema to UML diagram. Selection of the method is a stakeholder's choice.

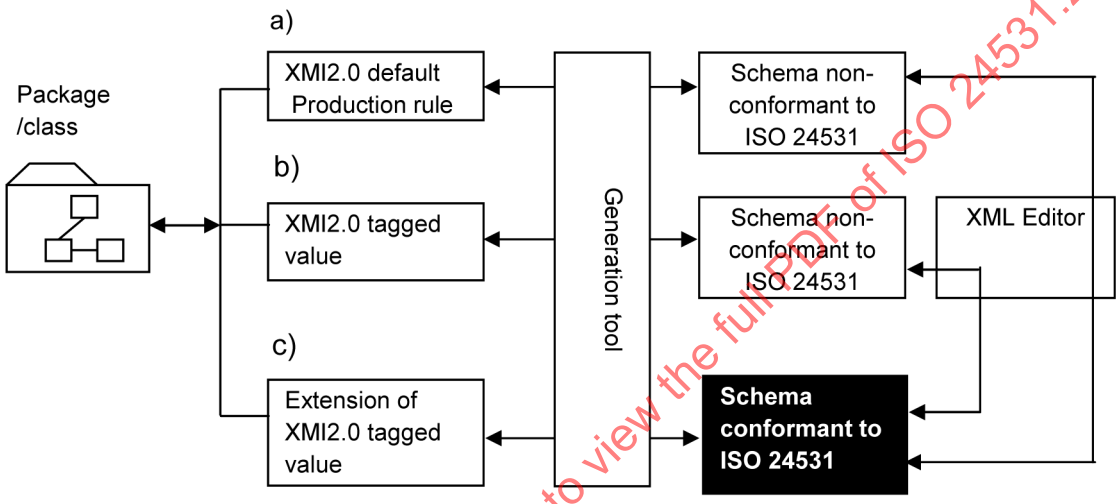



**Figure E.2 — UML <-> XML Schema Transformation**

## E.4 Transformation example

Method a) and b) of the Figure E.2 are straightforward; therefore, this clause only describes how to apply method c).

A profile is created by the following method.

— Enumerate UML and XML meta model of Schema and UML. MOF 2.0/XMI mapping specification, v2.1, Clause 5 refers to this procedure.

— Specified transformation condition was determined through:

  — enumerating the functionality that is included in XML but not included UML;

  — enumerating the syntactic difference between XML and UML;

  — enumerating the requirement of final form of the Schema including Clause 7 requirement.

— Stereotypes were specified through the following principle:

  — if tagged value already existed in XMI2.0, that was used;

  — number of stereotype and tagged value were minimized;

  — in naming of the profile, the top three letters are “XSD” except UML predefined stereotype were constructed.

    EXAMPLE

      <<XSDelement>> Stereotype to designate transform UML attribute to XML element. (XSD means XML Schema Definition)

      <<enumeration>> Stereotype predefined in UML.

  — for easy designation of stereotype in UML each stereotype and tagged value has scope.

    NOTE 1 There are three kinds of scope: “Package scope”, “Class scope” and “Construct scope”. “Package scope” is applied to all UML model constructs (all classes, all associationEnds, and all UML attributes); “Class scope” to class, association end, and all UML attributes, and so on.

    NOTE 2 The lower scope stereotype (e.g. Class scope) overrides the upper scope (e.g. Package scope).

  — for tagged value, the default tagged value is given.

    NOTE 3 Default tagged value is a value if stereotype tagged value is not declared.

In taking this profile, it is also relevant to consult [1] and [2] in the Bibliography.

Figure E.3 shows a stereotype set of the profile, and Table E.1 shows the tagged value of the stereotype placed in alphabetical order. In the second column of Table E.1, the default values are underlined. The default value is the value if this tagged value is not set. The tagged value can be used on its own, without designating a stereotype.

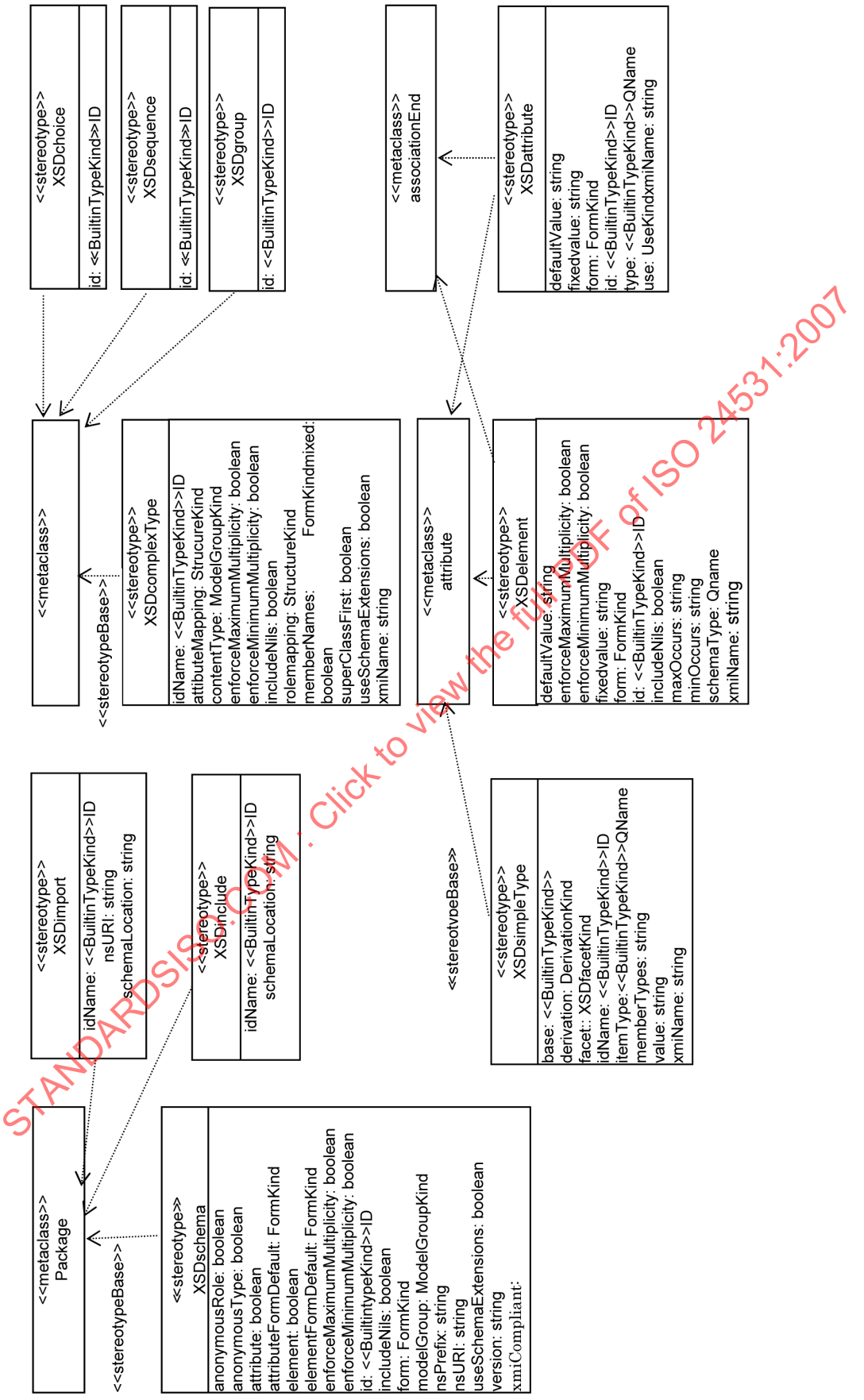





**Figure E.3 — UML profile for XML Schema**

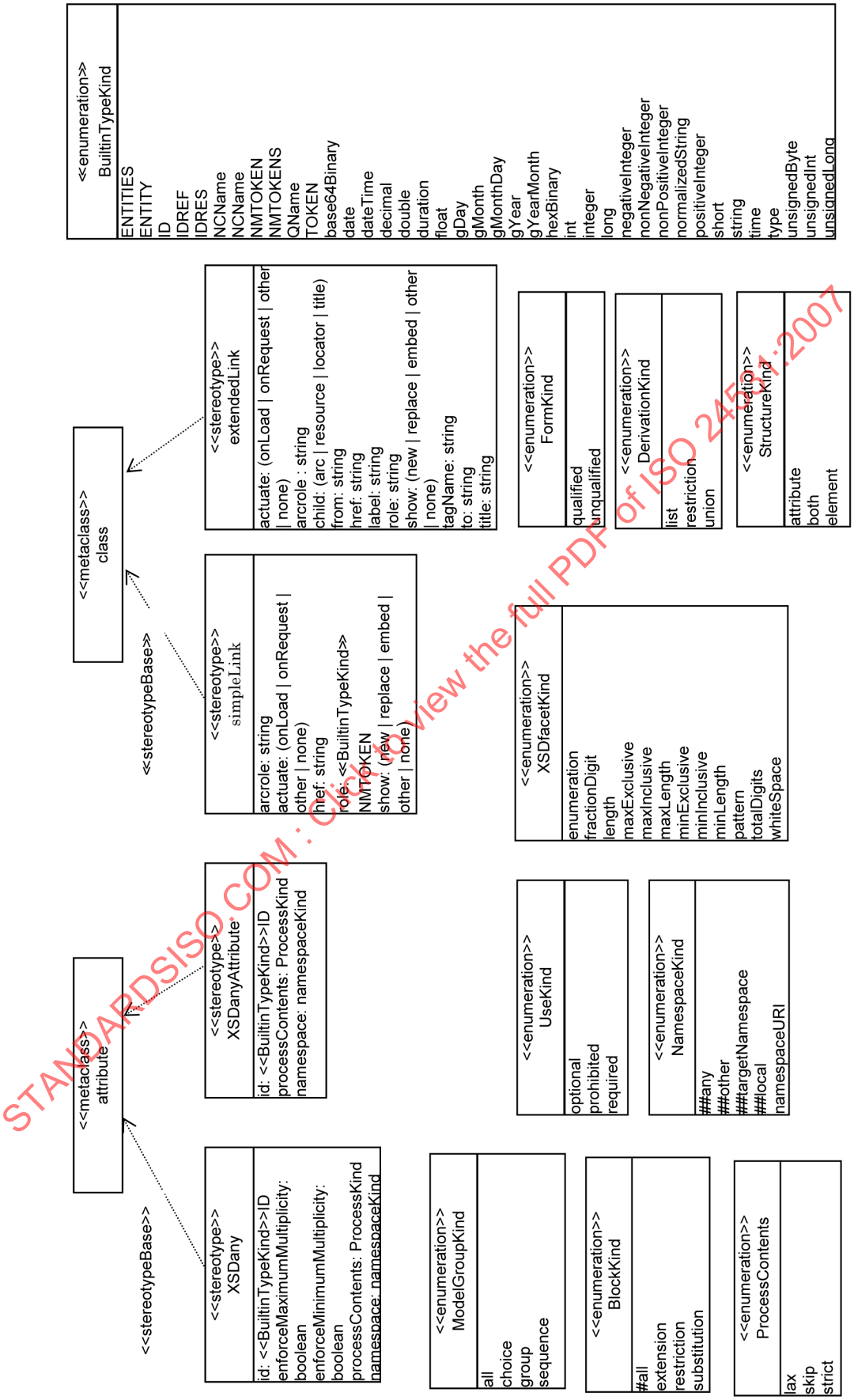





**Figure E.3** (*continued*)

**Table E.1 — Stereotype tagged value**

| Tag name | Tagged value | Description | Applied UML constructs | Scope | | | Note |
|---|---|---|---|---|---|---|---|
| | | | | Pack-age | Class | Const-ruct | |
| actuate | (onLoad \| onRequest \| other \| none) | The `actuate` attribute is used to communicate the desired timing of traversal from the starting resource to the ending resource (*See Xlink*). | Class | | X | | |
| anonymousRole | (true \| false) | The class type of this attribute or *associationEnd* will be embedded within the *complexType* definition for the owner class, omitting the role or attribute name element type wrapper. | Package, class | X | X | X | Realizes requirement 7.3.1.15 |
| anonymousType | (true \| false) | The class type of this attribute or *associationEnd* will be anonymous for XML documents defined by the generated Schema (that is the element defined by the attribute or *associationEnd* will contain child elements from the class but not the class element type). | Class, associationEnd | X | X | X | Realizes requirement 7.3.1.15 |
| arcrole | string | The arc `role` attribute indicates a property of the arc (*See XLInk*). | Class | | X | | |
| attribute | (true \| false) | If true, serialize the construct as an XML attribute. | Attribute, AssociationEnd | X | X | X | Tag defined in XMI2.0 False realizes requirement 7.3.1.13 |
| attributeForm-Default | (qualified \| unqualified) | Specifies whether local attribute declarations are qualified or unqualified. | Package, class, attribute | X | X | X | |
| childType | string | Specifies extended link type *child.* | Class | | X | | |
| contentType | (complex \|simple \| any \| mixed) | Defines the Schema content type. | Class, attribute, AssociationEnd | X | X | X | Tag defined in XMI2.0 |
| defaultValue | string | Specifies the default value for UML attribute. | Attribute | | | X | Tag defined in XMI2.0 |
| derivation | (restriction \| list \|union) | Select the kind of the *simpleType* derivation. | Attribute | | | X | |
| element | true | If true, serialize the construct as an XML element. | Attribute, AssociationEnd | X | X | X | Tag defined in XMI2.0 Realizes requirement 7.3.1.13 |
| elementForm | (qualified \| unqualified) | Specifies whether local element declarations are qualified or unqualified. | Package, class, construct | X | X | X | |
| enforce Maximum-Multiplicity | (false \| true) | If true, enforce maximum multiplicities; otherwise, they are “unbounded”. | Attribute, AssociationEnd | X | X | X | Tag defined in XMI2.0 |
| enforceMinimum -Multiplicity | (false \| true) | If true, enforce minimum multiplicities; otherwise, they are “0”. | Attribute, AssociationEnd | X | X | X | Tag defined in XMI2.0 |
| facet | FacetKind See Figure E.3 | Facets are used to restrict the set of values a *datatypes* with different value range to derive from other types. | Attribute | | | X | |

**Table E.1** (*continued*)

| Tag name | Tagged value | Description | Applied UML constructs | Scope | | | Note |
|---|---|---|---|---|---|---|---|
| | | | | Pack-age | Class | Const-ruct | |
| fixedValue | string | Specifies the fixed value for attributes. | Attribute | | | X | Tag defined in XMI2.0 |
| form | (qualified \| unqualified) | Specifies the value of the form attribute for attributes. Other valid values are *qualified* and *unqualified*. | Attribute, AssociationEnd | X | X | X | Tag defined in XMI2.0 |
| fractionDigits | positiveInteger | Specifies the maximum number of decimal digits to the right of the decimal point. | Attribute | | | X | |
| from | string | Extended link originate. | | | X | | |
| href | true | If true, use the *href* attribute rather than the *idref* attribute for links within a document. | AssociationEnd | X | X | X | Tag defined in XMI2.0 |
| IdName | (string \| xmi:id) | The value is the name of the ID attribute. | Class | X | X | X | Tag defined in XMI2.0 Realizes requirement 7.3.1.10 |
| includeNils | true | If false, do not serialize nil values. | Attribute | X | X | X | Tag defined in XMI2.0 |
| label | string | The `label` attribute provides a way for an `arc`-type element to refer to it in creating a traversal arc (*see Xlink*). | Class | | X | | |
| length | nonnegative-Integer | Defines the number of characters in a string-based type, the number of octets in binary-based type, or the number of items in a list-based type. | Attribute | | | X | |
| maxExclusive | number | Specifies an exclusive upper bound of the value space of the type. | Attribute | | | X | |
| maxInclusive | number | Specifies an inclusive upper bound of the value space of the type. | Attribute | | | X | |
| maxLength | nonNegativeInteger | Defines the maximum number of characters in a string-based type, the maximum number of octets in a binary-based type, or the maximum number of list-based type. | Attribute | | | X | |
| maxOccurs | nonNegativeInteger | Specifies the maximum number of times this element may appear in the context in which the declaration appears. | Attribute | | | X | |
| memberNames | (qualified \| unqualified) | Determines whether the UML *attribute* and *AssociationEnd* names are qualified by the UML class name. | Attribute, AssociationEnd | X | X | X | |
| minInclusive | number | Specifies an inclusive lower bound of the value space of the type. | Attribute | | | X | |

**Table E.1** (*continued*)

| Tag name | Tagged value | Description | Applied UML constructs | Scope | | | Note |
|---|---|---|---|---|---|---|---|
| | | | | Pack-age | Class | Const-ruct | |
| minLength | nonNegative Integer | Defines the minimum number of characters in a string-based type, the minimum number of octets in a binary-based type, or the minimum number of list-based type. | Attribute | | | X | |
| minOccurs | nonNegative Integer | Specifies the minimum number of times this element may appear in the context in which the declaration appears. | Attribute | | | X | |
| mixed | (true \| false) | If set to true, this attribute specifies that the content model of the complex type may contain text and element children. | Package, class | X | X | | |
| modelGroup | all \| sequence \| choice | Indicates the model group used when generating *complexType* definitions for Schema. | Attribute, AssociationEnd | X | X | | |
| namespace | ##any \| ##other \| ##targetNamespace \| ##local \| namespaceURI | Specifies whether a schema processor should find schema information and validate the elements appearing in place of the wildcard. | | | | X | |
| nsPrefix | string | The namespace prefix of the UML package; this is used in schemas. (Any legal XML prefix may be used in documents.) | Package | X | | X | Tag defined in XMI2.0 |
| nsURI | string | The namespace URI of the UML package. | Package | X | | X | Tag defined in XMI2.0<br>Realizes requirement 7.3.1.11 |
| processContents | lax \| skip \| strict | If *contentType* is any. This tag is used to specify the value of the *processContents* attribute of the any element. | Attribute | | | X | Tag defined in XMI2.0 |
| remoteOnly | true | If set on one end of bi-directional relationship, only serializes that end if it is remote. | AssociationEnd | X | X | X | Tag defined in XMI2.0 |
| role | string | The `role` attribute indicates a property of the resource (*see XLInk*). | Class | | X | | |
| roleMapping | (element \| attribute) | Indicates the default for generating UML association roles as either element or attribute. | AssociationEnd | | | X | |
| schemaLocation | string | Identifies the location of the schema to be imported. | Package | X | | | |

**Table E.1** (*continued*)

| Tag name | Tagged value | Description | Applied UML constructs | Scope | | | Note |
|---|---|---|---|---|---|---|---|
| | | | | Pack-age | Class | Const-ruct | |
| schemaType | XML Built-in dataType | The name of a *datatype* defined in the XML *Schema Datatype* specification. | Attribute | | | X | Tag defined in XMI2.0 |
| show | (new \| replace \| embed \| other \| none) | The `show` attribute is used to communicate the desired presentation of the ending resource on traversal from the starting resource (*see Xlink*). | Class | | X | | |
| superClassFirst | true | If true, serialize the super class content first. | Class | X | X | X | Tag defined in XMI2.0 |
| tagName | string | Tag name used for child element of extended type element. | Class | | X | | |
| title | string | The *title* attribute is used to describe the meaning of a link or resource in a human-readable fashion. | Class | | X | | |
| to | string | Extended link type destination. | | | X | | |
| totalDigit | positiveInteger | Specifies the maximum number of decimal digits for types derived from number. | Attribute | | | X | |
| use | (optional \| required \| prohibited ) | Specifies whether the attribute is optional, required, or prohibited. | Attribute | | | X | |
| useSchema-Extensions | true | If true, use schema extensions to represent inheritance in the MOF metamodel. | Class | X | X | X | Tag defined in XMI2.0 |
| value | string | Defines regular expression. | Attribute | | | X | |
| version | string | Version of the schema. | Package | X | | | Realize requirement 7.3.1.6 |
| whiteSpace | (preserve \| replace \| collapse) | Defines rules for *whiteSpace* normalization. | Attribute | | | X | |
| xmiCompliant | (true \| false) | Determines whether the required XMI attributes and wrapper elements are generated for this Schema. | Package | X | | | |
| xmiName | string | Provides an alternate name from the UML name for writing to XMI. Useful in cases where the UML name has characters that conflict with XML. This value is used rather than the UML name. | Class, attribute, AssociationEnd | | | | Tag defined in XMI2.0 |

### E.4.1 Real example

In this clause, generation of tailored XML Schema is given applying the profile for hypothetical "*RoadConditionMessage*" package. "*RoadConditionMessage*" consists of three kinds of information, namely "*Location"*, "*WeatherCondition"* and "*RoadSurfaceCondition*". Content of the above information is depicted in Figure E.5. Assumption of independent measurement of *WeatherCondition* and *RoadSurfaceCondition*, both have time attribute. Stereotyped package and class is shown in Figure E.5. To achieve Figure E.4 format for "*RoadConditionMessage*", position tag is used on each class.

Listing E-1 is the result of tailored Schema.

**Listing E-1 Customized Schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" id="iso_standard_24531_schema_1"
targetNamespace="http://www.example.com/" version="1.0a"
xmlns:rc="http://www.example.com">


    <xs:element name="location" id="iso__standard__24531_schema_1_element_1"
type="rc:LocationStructure"/>
    <xs:complexType name="LocationStructure">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="latitude" type="xs:string"/>
            <xs:element name="longitude" type="xs:string"/>
            <xs:element name="height" type="xs:float"/>
        </xs:sequence>
        <xs:attribute name="locationID" type="xs:integer" use="required"/>
        <xs:attribute name="heightUnit" type="xs:string" fixed="m" use="required"/>
    </xs:complexType>
    <xs:element name="weatherCondition" id="iso_standard_24531_schema_1_element_2"
type="rc:WeatherConditionStructure"/>
    <xs:complexType name="WeatherConditionStructure">
        <xs:sequence maxOccurs="unbounded" minOccurs="1">
            <xs:element name="weather" type="xs:string"/>
            <xs:element name="windDirection" type="xs:string"/>
            <xs:element name="windVelocity" type="xs:float"/>
        </xs:sequence>
        <xs:attribute name="Time" type="xs:dateTime" use="required"/>
        <xs:attribute name="WindVelocityUnit" type="xs:string" fixed="m/sec"/>
    </xs:complexType>
    <xs:element name="roadSurfaceCondition" id="iso_standard_24531_schema_1_element_3"
type="rc:RoadSurfaceConditionStructure"/>
    <xs:complexType name="RoadSurfaceConditionStructure">
        <xs:sequence maxOccurs="unbounded" minOccurs="1">
            <xs:element name="coefficientOfFriction" type="rc:CoefficientOfFriction"/>
        </xs:sequence>
        <xs:attribute name="time" type="xs:dateTime" use="required"/>
    </xs:complexType>
    <xs:simpleType name="CoefficientOfFriction">
        <xs:restriction base="xs:float">
            <xs:maxExclusive value="1"/>
            <xs:minExclusive value="0"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```



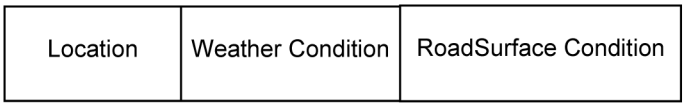



**Figure E.4 — RoadConditionMessage structure**

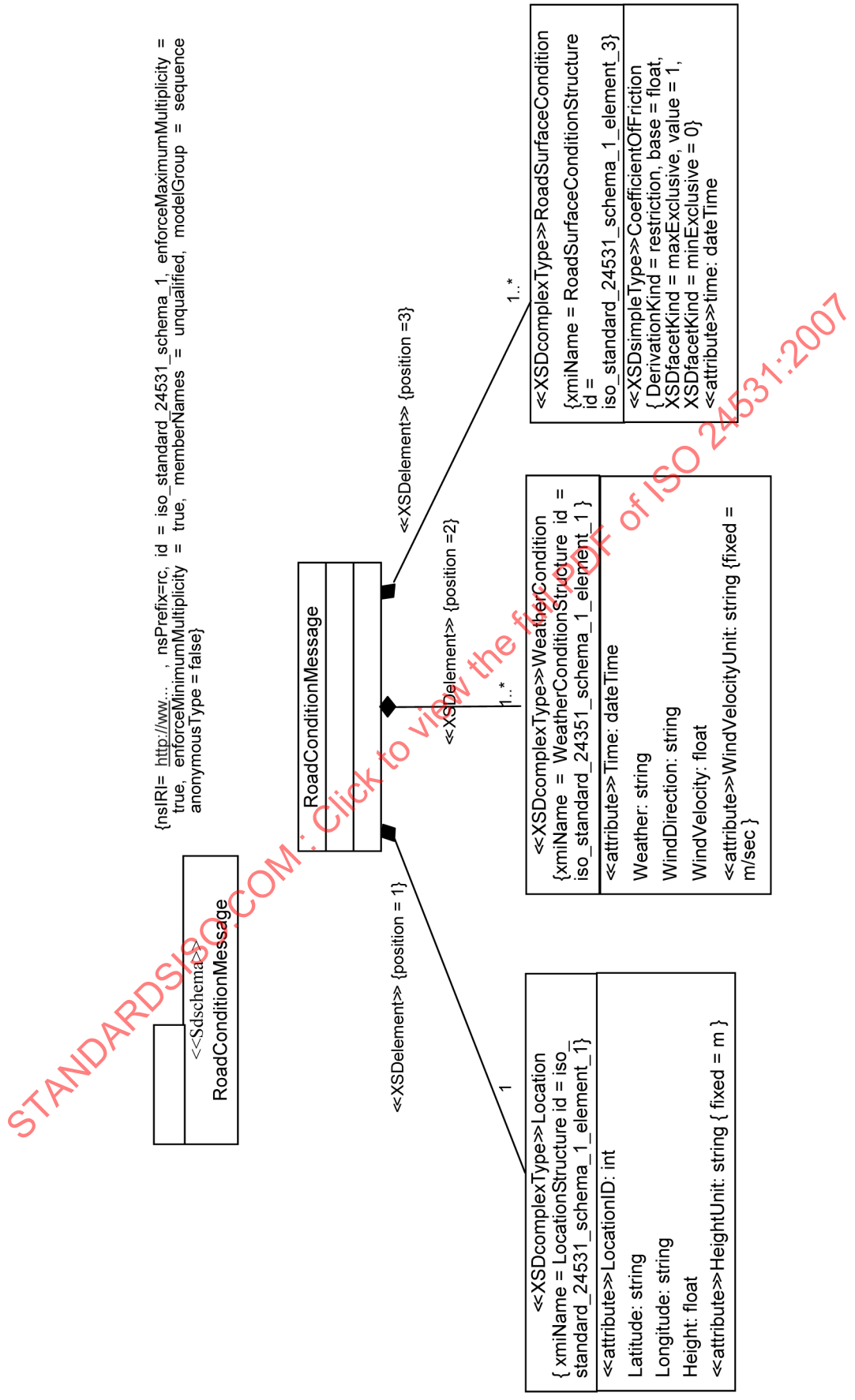





**Figure E.5 — RoadCondition constructs customization**

# Annex F
(informative)

# Applying ASN.1 encoding for XML document

XML encoding is generally less efficient than using ASN.1 (BER, CER, DER. PER).

The main reasons are:

— ASN.1 encodes data “bit based”, but XML sends data “character based”.

— Contrary to XML, ASN.1 encodes tag value efficiently in bit base or, in some cases where this does not exist, XML always needs a start tag and end tag as a delimiter. Normally, because the tag represents the meaning of the data, it tends to become long.

In the case of a narrow band network, e.g. cellular network or DSRC, efficient encoding is essential. Usually, mobile network bit error rate is larger than that in a wired network. Because the transmission error rate is proportional to the message length, it is critical to minimize the message length for transaction processes such as electronic fee collection. To respond to this need, ASN.1 encoding may be applied to the XML message in some cases.

Whether applying ASN.1 encoding or not, from the viewpoint of required resource of CPU power and response time, [4] and [5] provide some example XML and various (including ASN.1 encoding) encoding size comparisons.

This annex provides examples of three methods of effective encoding.

**Method 1: Construct application in ASN.1** (see Figure F.1)

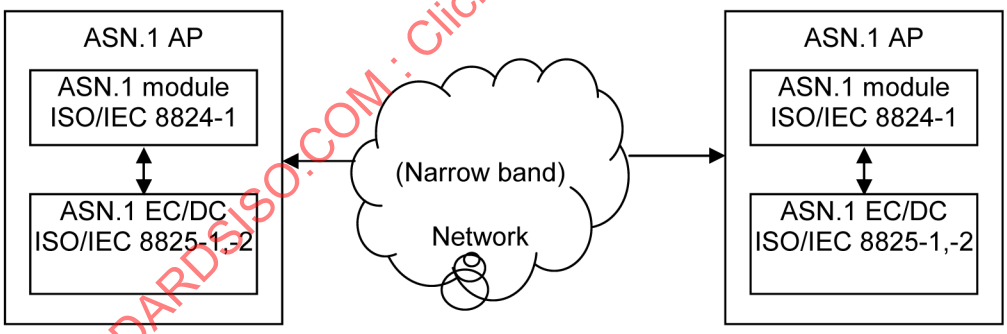



**Figure F.1 — Efficient encoding using ASN.1 encoding**



**Method 2: Generate ASN.1 module from XML schema** (see Figure F.2)

If an XML Schema already exists, an ASN.1 module is created using ISO/IEC 8825-5, and *Method 1* is applied.

In this case, a new ASN.1 application development is required.

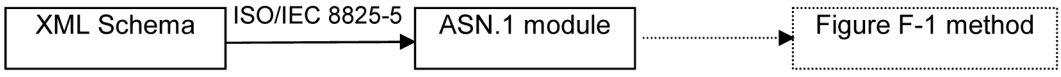



**Figure F.2 — Use of XML Schema transformation**