DD CEN/TS 16157-1:2011

BSI Standards Publication

Intelligent transport systems — DATEX II data exchange specifications for traffic management and information

Part 1: Context and framework

bsi.

...making excellence a habit.™

National foreword

This Draft for Development is the UK implementation of CEN/TS 16157-1:2011. DD ENV 13106:2000 and DD ENV 13777:2000 were withdrawn in 2007 as ENV 13106:2000 and ENV 13777:2000 expired without producing an EN standard.

DATEX II is founded on a single UML model which is made available in a useable format to all through the website, http://www.datex2.eu/.

This publication is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature. It should be applied on this provisional basis, so that information and experience of its practical application can be obtained.

Comments arising from the use of this Draft for Development are requested so that UK experience can be reported to the international organization responsible for its conversion to an international standard. A review of this publication will be initiated not later than 3 years after its publication by the international organization so that a decision can be taken on its status. Notification of the start of the review period will be made in an announcement in the appropriate issue of *Update Standards.*

According to the replies received by the end of the review period, the responsible BSI Committee will decide whether to support the conversion into an international Standard, to extend the life of the Technical Specification or to withdraw it. Comments should be sent to the Secretary of the responsible BSI Technical Committee at British Standards House, 389 Chiswick High Road, London W4 4AL.

The UK participation in its preparation was entrusted to Technical Committee EPL/278, Road transport informatics.

A list of organizations represented on this committee can be obtained on request to its secretary.

ISBN 978 0 580 72442 8

ICS 35.240.60

This Draft for Development was published under the authority of the Standards Policy and Strategy Committee on 31 October 2011.

Amendments issued since publication

Date	Text affected

TECHNICAL SPECIFICATION

SPÉCIFICATION TECHNIQUE

TECHNISCHE SPEZIFIKATION

CEN/TS 16157-1

October 2011

ICS 35.240.60

Supersedes ENV 13106:2000, ENV 13777:2000

English Version

Intelligent transport systems - DATEX II data exchange specifications for traffic management and information - Part 1: Context and framework

Systèmes de transport intelligents - Spécifications DATEX II d'échange de données pour la gestion du trafic et l'information routière - Partie 1: Contexte et cadre général

Intelligente Transportsysteme - DATEX II Datenaustausch Spezifikationen für Verkehrsmanagement und Informationen - Teil 1: Kontext und Rahmenstruktur

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

Contents

Page

Foreword

This document (CEN/TS 16157-1:2011) has been prepared by Technical Committee CEN/TC 278 “Road transport and traffic telematics”, the secretariat of which is held by NEN.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

This document supersedes ENV 13106:2000, ENV 13777:2000.

As a user of the standard, attention is drawn to the resources of www.datex2.eu. This web site contains related software tools and software resources that aid the implementation of CEN/TS 16157 DATEX II.

According to the CEN/CENELEC Internal Regulations, the national standards organizations of the following countries are bound to announce this Technical Specification: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

Introduction

This Technical Specification defines a common set of data exchange specifications to support the vision of a seamless interoperable exchange of traffic and travel information across boundaries, including national, urban, interurban, road administrations, infrastructure providers and service providers. Standardisation in this context is a vital constituent to ensure interoperability, reduction of risk, reduction of the cost base, promotion of open marketplaces and many social, economic and community benefits to be gained from more informed travellers, network managers and transport operators.

Delivering European Transport Policy in line with the White Paper issued by the European Commission requires co-ordination of traffic management and development of seamless pan European services. With the aim to support sustainable mobility in Europe, the European Commission has been supporting the development of information exchange mainly between the actors of the road traffic management domain for a number of years. In the road sector, DATEX II has been long in fruition, with the European Commission being fundamental to its development through an initial contract and subsequent co-funding through the Euro-Regional projects. With this standardisation of DATEX II there is a real basis for common exchange between the actors of the traffic and travel information sector.

This Technical Specification includes the framework and context for exchanges, the modelling approach, data content, data structure and relationships, communications specification.

This Technical Specification supports a methodology that is extensible.

The European Committee for Standardisation (CEN) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning procedures, methods and/or formats given in this document.

CEN takes no position concerning the evidence, validity and scope of patent rights.

This document (i.e. CEN/TS 16157-1) is targeted towards all stakeholders that want to understand the modelling methodology applied throughout the DATEX II specifications. While this is potentially a wide range of readers, the document addresses specifically those users that intend to extend the DATEX II data model and therefore need to understand – and comply with – the modelling principles, the use of the “Unified Modeling Language” (UML) and other conventions for DATEX II modelling. The subject matter may be difficult to address without some basic background about the historical evolution and main design decisions taken. Users unfamiliar with this background may find a brief summary in Annex A. Users not (yet) familiar with the UML find a brief introduction in the informative Annex B.
Further to the UML modelling, this Technical Specification also defines the mapping of this model to the “eXtensible Markup Language” (XML), used for formatting data in DATEX II data exchanges. XML is the most widely used method nowadays of formatting data for business-to-business data exchange (i.e. centre-to-centre) over the Internet.

1 Scope

This Technical Specification (CEN/TS 16157-1) specifies and defines component facets required to support the exchange and shared use of data and information in the field of traffic and travel.

The component facets include the framework and context for exchanges, the modelling approach, data content, data structure and relationships, communications specification.

This Technical Specification is applicable to:

— traffic and travel information which is of relevance to road networks (non urban and urban);

— public transport information that is of direct relevance to the use of a road network (e.g. road link via train or ferry service).

This Technical Specification establishes specifications for data exchange between any two instances of the following actors:

— Traffic Information Centres (TICs);

— Traffic Control Centres (TCCs);

— Service Providers (SPs).

Use of this Technical Specification may be applicable for use by other actors.

This Technical Specification covers, at least, the following types of informational content:

— road traffic event information – planned and unplanned occurrences both on the road network and in the surrounding environment;

— operator initiated actions;

— road traffic measurement data, status data, and travel time data;

— travel information relevant to road users, including weather and environmental information;

— road traffic management information and information and advice relating to use of the road network.

This part of CEN/TS 16157 specifies the DATEX II framework of all parts of this Technical Specification, the context of use and the modelling approach taken and used throughout these Technical Specifications. This approach is described using formal methods and provides the mandatory reference framework for all other parts.

2 Conformance

This document provides requirements for UML models (as of ISO/IEC 19501:2005) that claim conformance with the DATEX II Technical Specifications. UML models claiming this conformance shall comply with the provisions of the normative clauses and annex of this part. Conformance with metadata constructs is subject to multiplicity requirements stated explicitly in the model or is implicitly defined in provisions of this Technical Specification. Metadata constructs with minimum multiplicity of 1 or more shall be present in any data claiming conformance. Metadata constructs with minimum multiplicity of 0 may be present or may be missing without violating conformance.

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 639-2:1998, *Codes for the representation of names of languages — Part 2: Alpha-3 code*

ISO 3166-1:2006, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO/IEC 14977:1996, *Information technology — Syntactic metalanguage — Extended BNF*

ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

ISO/IEC 19503:2005, *Information technology — XML Metadata Interchange (XMI)*

4 Terms and definitions

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

4.1 Terms and definitions adapted from ISO/IEC 19501:2005

NOTE Definitions have been adapted to meet the particular use of UML within this specification.

4.1.1
association
semantic relationship between classes

4.1.2
association end
endpoint of an association, which connects the association to a class

4.1.3
aggregation
association where the target class is an aggregate; therefore the source class is a part. This characteristic is expressed in UML with an attribute named “aggregation” on the target side Association End being set to “aggregate”

4.1.4
attribute
named slot within a class that describes a range of values that instances of the class may hold

4.1.5
class
description of a set of objects that share the same attributes, relationships, and semantics

4.1.6
composition
association where the target class is a composite; therefore the source class is a part that is strongly owned by the composite and may not be part of any other composite. This characteristic is expressed in UML with an attribute named “aggregation” on the target side Association End being set to “composite”

4.1.7
dependency
dependency states that the implementation or functioning of one or more elements requires the presence of one or more other elements

4.1.8
enumeration
data type whose range is a list of predefined values, called enumeration literals

4.1.9
enumeration Literal
element of the run-time extension of an Enumeration data type

NOTE It has no relevant substructure, that is, it is atomic. The enumeration literals of a particular Enumeration data type are ordered.

4.1.10
generalization
taxonomic relationship between a more general element and a more specific element

NOTE 1 The more specific element is fully consistent with the more general element (it has all of its properties, members, and relationships) and may contain additional information.

NOTE 2 In the metamodel, a Generalization is a directed inheritance relationship, uniting a GeneralizableElement with a more general GeneralizableElement in a hierarchy. Generalization is a subtyping relationship; that is, an Instance of the more general GeneralizableElement may be substituted by an Instance of the more specific GeneralizableElement.

4.1.11
multiplicity
in the metamodel a MultiplicityRange defines a range of integers. The upper bound of the range cannot be below the lower bound. The lower bound must be a nonnegative integer. The upper bound must be a nonnegative integer or the special value unlimited, which indicates there is no upper bound on the range

4.1.12
package
grouping of model elements

4.1.13
stereotype
stereotype concept provides a way of branding (classifying) model elements so that they behave in some respects as if they were instances of new virtual metamodel constructs. These model elements have the same structure (attributes, associations, operations) as similar non-stereotyped model elements of the same kind. The stereotype may specify additional constraints and tag definitions that apply to model elements. In addition, a stereotype may be used to indicate a difference in meaning or usage between two model elements with identical structure

4.1.14
tagged value
tagged value allows information to be attached to any model element in conformance with its tag definition. Although a tagged value, being an instance of a kind of ModelElement, automatically inherits the name attribute, the name that is actually used in the tagged value is the name of the associated tag definition. The interpretation of tagged values is intentionally beyond the scope of UML semantics. It must be determined by user or tool conventions that may be specified in a profile in which the tagged value is defined

4.2 Other terms and definitions

4.2.1
binary (association)
association that connects exactly <u>two</u> classes

4.2.2
globally unique identifier
GUID
Globally Unique Identifier that is unique in space and time, i.e. no other object will ever have the same identifier at any other place and at any time

4.2.3
lower camel case
LCC
Camel Case describes the practice of concatenating compound phrases without whitespace in between where phrases are delimited by upper case letters. Lower Camel Case describes the case where the initial letter is lower case, e.g. as in lowerCamelCase

4.2.4
model element
generic term for any construct of metadata used within a model to specify a particular aspect or element of this model

4.2.5
Platform Independent Model
PIM
model of aspects of an information system (e.g. the data model) that is independent of any technical platform used to implement the model. Concrete implementations can be derived from the platform independent model by platform specific models or mappings

4.2.6
platform specific model
PSM
model of aspects of an information system (e.g. the data model) that is linked to a specific technological platform (e.g. a specific programming language or data transfer syntax)

4.2.7
upper camel case
UCC
Camel Case describes the practice of concatenating compound phrases without whitespace in between where phrases are delimited by upper case letters. Lower Camel Case describes the case where the initial letter is upper case, e.g. as in UpperCamelCase

4.2.8
unique resource identifier / locator
URI / URL
character string of well defined structure used to uniquely identify a resource. If that string is actually pointing at a resource accessible via the Internet, it is called a Unique Resource Locator

4.2.9
extensible markup language
XML
set of rules for encoding electronic documents define by the World Wide Web Consortium W3C. Although developed for documents, it is today widely used for data exchange in general, usually in conjunction with an XML Schema Definition

4.2.10
XML metadata interchange
XMI
XML based specification for the interoperable exchange of metadata. It is today most commonly used to exchange UML models between UML tools. XMI is specified in ISO/IEC 19503:2005

4.2.11
XML schema definition
XSD
XML Schema Definition is a formal description of the allowed content of an XML document that claims compliance to the schema. XML Schema Definitions allow for formal validation of syntactical compliance of instance documents

5 Symbols and abbreviated terms

GUID	Globally Unique identifier
LCC	Lower Camel Case
PIM	Platform Independent Model
PSM	Platform Specific Model
UCC	Upper Camel Case
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
XSD	XML Schema Definition

6 General conventions and requirements

6.1 Metamodelling

The DATEX II modelling methodology (see Annex C) uses the Unified Modeling Language (UML), version 1.4.2 as specified in UML ISO/IEC 19501:2005. UML provides a vast set of modelling elements that are not all used for DATEX II data modelling. Further to the **selection** of UML modelling elements, this clause also provides requirements for DATEX II modelling regarding the **use** of these elements. Models that claim to comply with this specification may use these UML elements but shall comply with all provisions regarding the use of these elements. Annex B provides a brief introduction into the UML constructs used for DATEX II, although the authors recognise that there is plenty – presumably better – introductory material available to learn about UML in general and would like to refer the reader to these resources for further study.

Note that this clause no provisions are made regarding the existence and use of other UML elements. Thus, compliant models may use these other elements, but they have no defined semantics in the framework of this Technical Specification.

DATEX II compliant models may use the following metaclasses and metaattributes from the UML "Core" package:

— Class

  - Class.name
  - Class.isAbstract
  - Class.feature
  - Class.association
- Association
  - Association.connection
- AssociationEnd
  - AssociationEnd.name
  - AssociationEnd.aggregation
  - AssociationEnd.multiplicity
  - AssociationEnd.qualifier
  - AssociationEnd.participant
- Attribute
  - Attribute.name
  - Attribute.multiplicity
  - Attribute.type
- Enumeration
  - Enumeration.name
- EnumerationLiteral
  - EnumerationLiteral.name
- Generalization

DATEX II compliant models may use the following metaclasses and metaattributes from the UML "Extension Mechanisms” package:

- Tagged Value
  - TaggedValue.name
  - TaggedValue.dataValue
- Stereotype
  - Stereotype.name

DATEX II compliant models may use the following metaclasses and metaattributes from the UML "Data Types" package:

- Multiplicity
- Multiplicity.Range
  - MultiplicityRange.lower
  - MultiplicityRange.upper

DATEX II compliant models may use the following metaclasses and metaattributes from the UML "Model Management" package:

- Package
  - Package.name

Whenever one of these UML constructs is used in a UML model seeking compliance with this Technical Specification, all provisions contained in this specification governing the use of this particular construct shall be adhered to.

6.2 Naming conventions

The following conventions apply:

a) The following UML constructs used in this Technical Specification may have a name that is used by the DATEX II metamodel: AssociationEnd, Attribute, Class, Enumeration, EnumerationLiteral, Package, Stereotype, TaggedValue. This name is specified via a “name” metaattribute from that the metaclasses inherit from the abstract Core:ModelElement metaclass.

   If such a name metaattribute is provided, it shall begin with a letter, followed by none or more letters or digits. A name is case sensitive.

   In formal terms, a DATEX II name shall comply with the following definition using Extended Backus Naur Form as of ISO/IEC 14977:1996.

```
name ::=     letter ,  { letter  | digit }

letter ::=   “A” | “B” | “C” | “D” | “E” | “F” | “G” | “H” | “I” | “J” | “K” | “L” | “M” | “N” |
             “O” | “P” | “Q” | “R” | “S” | “T” | “U” | “V” | “W” | “X” | “Y” | “Z” | “a” | “b” |
             “c” | “d” | “e” | “f” | “g” | “h” | “i” | “j” | “k” | “l” | “m” | “n” | “o” | “p” |
             “q” | “r” | “s” | “t” | “u” | “v” | “w” | “x” | “y” | “z”

digit ::=    “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”
```

b) The following DATEX II names shall be in Upper Camel Case notation, i.e. they shall start with an upper case letter (‘A’ to ‘Z’), they shall consist of one or more logical components, and each component itself shall again begin with an upper case letter (Example: Component1Component2Component3):

   Class, Enumeration, Package

c) The following DATEX II names shall be in Lower Camel Case notation, i.e. they shall start with a lower case letter (‘a’ to ‘z’), they shall consist of one or more logical components, and each component starting with the second component shall begin with an upper case letter (Example: component1Component2Component3):

   Attribute, AssociationEnd, EnumerationLiteral, Stereotype, TaggedValue

d) In all DATEX II names acronyms should be avoided, but in cases where they are used, their capitalization shall be modified to comply with the provisions in 6.2 b) and c).

NOTE In some provisions of this Technical Specification, default names are defined by turning names of other constructs from Upper Camel Case to Lower Camel Case. This transformation is strictly defined as turning only the first character from upper case to lower case. Example: a proper class name may be "ANameExample". If referred by another class via an aggregation without role name and qualifier, the XML schema encoding of the enclosing class (see Annex D) will contain an attribute that will implicitly be named "aNameExample".

e) The names of the following DATEX II UML constructs shall be unique within their own object category in the whole model:

 Attribute, Class, Enumeration, Package

 Since the scope of attributes is limited to the containing class, this uniqueness requirement is not ensured by the use of UML or a UML tool. Thus, the DATEX II specifications provide their own definition of semantic identity for this particular case:

 In the case of Attributes this means that if two distinct UML Attributes have the same name, the "dataValue" of their corresponding "definition" TaggedValue shall be identical and their "type" metaattribute shall also have the same value (ref.7.2 h))).

6.3 Notational conventions

Text in italics is used for terms that are defined in Clause 4 (Terms and definitions) of this document. Examples are stereotype or XML schema.

Text in double quotes is used for reference to features of the underlying UML profile, especially stereotypes like “datatype” or TaggedValues like “definition” or “schemaType”, and for elements quoted from the DATEX II data model, e.g. "D2LogicalModel".

7 Platform independent model rules

7.1 General

The DATEX II modelling methodology implies a certain structure of a UML model that seeks to claim compliance with this specification. This clause states the requirements that UML models shall comply with in terms of constraints on the use of certain UML constructs. Note that no statements are made concerning other UML constructs, i.e. models that contain other UML constructs may still claim compliance with this Technical Specification, as long as all requirements for the named UML constructs are met.

Most provisions in this clause have been directly deduced from the metamodelling approach taken for DATEX II, which is explained further in informative Annex C. Nevertheless, others have been decided deliberately to guide users, improve modelling quality and avoid ambiguity.

The provisions in this clause address the particular requirements for the platform independent model, i.e. they do not address requirements for mapping DATEX II models to specific transfer syntax for exchange of data. Such requirements are dealt with in later clauses of this Technical Specification.

7.2 Numbered list of requirements

a) DATEX II models may use UML Classes. UML Classes shall have a “definition” UML TaggedValue.

b) UML Classes may have a “datatype” UML Stereotype assigned.

c) UML Classes in DATEX II models may have UML Attributes.

d) UML Attributes shall have a “definition” UML TaggedValue.

e) UML Attributes shall have an assigned “type” element. The assigned type shall be a UML Class with UML Stereotype “datatype” (Note that built-in UML types are not allowed) or it shall be a UML Enumeration. If the assigned type is either “Reference” or “VersionedReference”, the UML Attribute shall have a “targetClass” UML TaggedValue, which shall provide a name of a UML Class that has an “identifiable” or “versionedIdentifiable” Stereotype assigned, respectively.

f) UML Attributes shall have an “order” UML TaggedValue. This order shall be a non-negative integer and all order values of attributes of the same UML Class shall be unique within this UML Class.

g) UML Attributes may have a “multiplicity” element attached. In case multiplicity is not provided explicitly, a default value of "1..1" is used.

h) UML Attributes names have a global name scope in DATEX II, i.e. two UML Attributes with the same name shall have the same definition and type values.

i) DATEX II models may contain UML Enumerations. These UML Enumerations may contain UML EnumerationLiterals.

j) UML Enumerations shall have a “definition” UML TaggedValue assigned.

k) UML EnumerationLiterals shall have a “definition” UML TaggedValue assigned.

l) UML EnumerationLiterals shall have an “order” UML TaggedValue assigned. This order shall be a non-negative integer and all order values of UML EnumerationLiterals of the same UML Enumeration shall be unique within this UML Enumeration.

m) For all UML Classes that do not have a “datatype” UML Stereotype assigned, the following UML constructs shall be unique:

— UML Attribute “names”

— “qualifiers” of UML AssociationEnds directly connected to the UML Class

— “names” of remote UML AssociationEnds of UML Associations connected to the class

— names of UML Classes connected via the source side of an UML Association without an UML AssociationEnd name on the source end – with their UML Class name’s first letter turned to lower case

n) UML Classes may be declared to be “abstract”.

o) UML Classes may have an “identifiable” UML Stereotype assigned or they may have a “versionedIdentifiable” UML Stereotype assigned. They shall not have both stereotypes assigned.

p) UML Classes which themselves or any of their ancestor classes (i.e. a direct UML Generalization, UML Generalization of its UML Generalization, etc.) have an “identifiable” or “versionedIdentifiable” UML Stereotype assigned shall not have a “datatype” UML Stereotype assigned to themselves or their ancestor classes.

q) UML Classes shall not have two or more UML Generalizations.

r) DATEX II models may use UML Associations between UML Classes. UML Associations shall be binary. UML Associations may have a “multiplicity” element, may have a “definition” UML TaggedValue and may have a (role) “name” attached to their source side UML AssociationEnd. If multiplicity is not provided explicitly for a source side UML AssociationEnd, the default is "1..1". If a role name is not provided for a source side UML AssociationEnd, the default assumption is the name of the source side UML Class with the first character turned to lower case.

s) The target side UML AssociationEnd of UML Associations in DATEX II models may have their "aggregation" attribute set to either "aggregate" or "composite". If the value of this attribute is "none", the association is not governed by this specification and does not have to be compliant to the requirements provided in this document.

t) Note that this clause means that arbitrary associations which are neither aggregations nor compositions are allowed in a DATEX II model but do not have semantics in the scope of DATEX II.

u) UML Associations in DATEX II models may have a qualifier on their target side association end.

v) UML Associations in DATEX II models shall have an “order” UML TaggedValue assigned to their target side UML AssociationEnd. The target order value shall be a non-negative Integer and shall be different from any other value of any other UML Association that shares the same UML Class for its target side UML AssociationEnd (i.e. all target association ends ending in the same class shall have distinct order values).

w) If two or more UML Associations have the same source and target UML Classes, they shall have “definition” and source side UML AssociationEnd “name” values.

8 Predefined model elements

8.1 General

Besides regulations for the use of UML constructors and a UML profile providing additional metainformation via tagged values and stereotypes, the DATEX II modelling methodology furthermore stipulates a certain top level model structure for all compliant UML models. These clauses are mainly motivated by the need to create a well defined structure for DATEX II tools aiming at supporting users.

It is in principle possible to create models in accordance to Clauses 6 and 7 (general and platform independent model related clauses) – even including the mapping to XML Schema Definition described in Annex D – that do not comply with this clause. Nevertheless, note that such a model cannot claim full compliance with this Technical Specification and thus may not work with tools requiring full compliance.

8.2 Top Level model packages and classes

The following Top Level model packages and classes apply:

a) DATEX II compliant UML models shall have one single top level UML package named "D2LogicalModel". This top-level package shall contain a UML Class of the same name, "D2LogicalModel".

b) The UML Class "D2LogicalModel" shall have a “rootElement” UML TaggedValue with “dataValue” equal to “d2LogicalModel”.

c) The UML Class “D2LogicalModel” shall also have a UML TaggedValue named “modelBaseVersion” that has a value that corresponds to the DATEX II model version identifier. The version in accordance to this Technical Specification shall have the fixed value "2". The class shall furthermore have a UML Tagged Value named "version" with a value of the full version, including minor versions. The value of minor versions conformant to this Technical Specification shall have the form 2.n, where "n" is the minor version number.

NOTE The model base version “2” denotes the second iteration of the second generation of DATEX specifications, denoted “DATEX II”. The Arabic version number “2” is not to be mixed up with the Roman “II” used to give this generation a name that distinguishes it from the EDIFACT-based “DATEX” standard developed in the 1990ies, finally resulting in the meanwhile withdrawn CEN ENV 13106:2000 and ENV 13777:2000.

d) The UML Class “D2LogicalModel” may have UML TaggedValues named “extensionName” and “extensionVersion” that contain the name of the extension(s) contained in the model – if any – and a corresponding version identifier. These values shall be provided by the creator of the model.

e) The DATEX II top level package "D2LogicalModel" shall have at least two sub-packages named "General" and "PayloadPublication".

f) The "PayloadPublication" package shall contain at least one abstract UML Class named "PayloadPublication". It may contain further packages and classes.

g) The "D2LogicalModel" class in the "D2LogicalModel" package shall have an aggregation association to the "PayloadPublication" class, with multiplicity "0..1" on the part side.

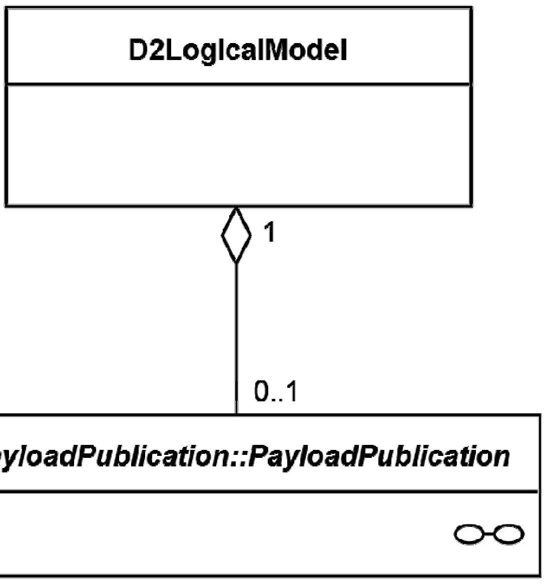


Figure 1 — The D2LogicalModel package

Note that in conjunction with 8.2 b) this provides a well defined entry structure into a DATEX II XML publication, which always starts with a top level "d2LogicalModel" object that may contain at most one concrete instance of a class specialized from "PayloadPublication".

h) The class "PayloadPublication" shall have the following structure:

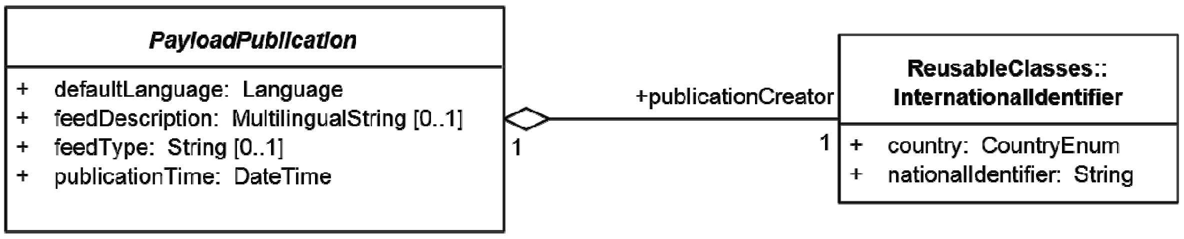

PayloadPublication
+ defaultLanguage: Language
+ feedDescription: MultilingualString [0..1]
+ feedType: String [0..1]
+ publicationTime: DateTime
+publicationCreator
1
1
ReusableClasses::
InternationalIdentifier
+ country: CountryEnum
+ nationalIdentifier: String

Figure 2 — The PayloadPublication package

The UML Class “PayloadPublication” shall also have a UML TaggedValue named “definition” with a content of "A payload publication of traffic related information or associated management information created at a specific point in time that can be exchanged via a DATEX II interface".

The class "PayloadPublication" further has one aggregation association to another class with name "InternationalIdentifier". The class “InternationalIdentifier” shall have a UML TaggedValue named “definition” with a content of "An identifier/name whose range is specific to the particular country".

Annex E provides the normative values for the “definition” and “order” UML TaggedValues of the attributes of classes “PayloadPublication” and "InternationalIdentifier".

The possible ISO 3166-1 codes related to “country” attribute values are fixed in an enumeration type with the following literals: be; bg; ch; cs; cy; cz; de; dk; ee; es; fi; fo; fr; gb; gg; gi; gr; hr; hu; ie; im; is; it; je; li; lt; lu; lv; ma; mc; mk; mt; nl; no; pl; pt; ro; se; si; sk; sm; tr; va; other;

All other types referred to in this clause are defined in 8.3.

i) The "General" package shall have at least one sub-package named "DataTypes", which again shall have at least one sub-package named "Generic".

j) This "Generic" package shall at least contain two classes named "Reference" and “VersionedReference”, with a "datatype" stereotype assigned and a "definition" tagged value that has a value of:

   "A reference to an identifiable managed object where the identifier is unique. It comprises an identifier (e.g. GUID) and a string identifying the class of the referenced object."

   for “Reference” and

   “A reference to an identifiable version managed object where the combination of the identifier and version is unique. It comprises an identifier (e.g. GUID), a version (NonNegativeInteger) and a string identifying the class of the referenced object.”

   in case of “VersionedReference”.

The "DataTypes" package and all packages contained therein shall contain only UML Classes with the "datatype" stereotype.

8.3 Basic datatypes

Besides the "Reference" and “VersionedReference” data types, the "Generic" sub-package of the "DataTypes" package shall contain at least the ”datatype” stereotyped UML Classes, with according definitions and XML Schema Definition mappings, as described in Annex E.

9 Extension Rules

9.1 General

DATEX II models enable application specific extensions. These extensions may implement innovative concepts, and while they may happily reuse data types, enumerations, components and even identifiable entities from an existing model, they may not seek any type of system level interoperability with systems being implemented without being cognizant of this particular extension. Such extensions are denoted as level C extensions.

In other scenarios, extensions may only seek to add some limited amount of application specific business logic whilst at the same time requiring backward compatibility with an existing model. “Compatibility” here means system level interoperability, i.e. for systems exchanging XML messages a valid instance of an extended model shall always be also a valid instance for the core model. This level of interoperability is in DATEX II denoted as level B extension. The levels' names actually indicate a compatibility hierarchy where the top level (maximum compatibility) is denoted as level A, where both interacting system use an identical model.

9.2 Numbered list of requirements

a) A model that is conforming to this Technical Specification may be extended. Extensions may either seek backwards compatibility to an existing model (denoted 'core model' in this clause), or they may create a new model not compatible to any previous model, but nevertheless using the methodology provided within this Technical Specification and – potentially – reusing classes taken from other, existing models.

   A compatible extension is denoted within this Technical Specification as a level B extension.

   Non-compatible extensions are denoted as level C extensions.

b) All extensions shall fully comply with all other rules presented so far in this document.

c) An extended model shall provide extension name and version number in two tagged values called "extensionName" and "extensionVersion" on the “d2LogicalModel” element and on any other root level elements (defined using a “rootElement” tagged value), that shall be usable in conjunction with extended elements.

d) Classes belonging to an extension and having a superclass not belonging to the extension (i.e. extension classes that inherit from the core model) shall have an "extension" tagged value with values either "levelb" or "levelc".

   Extensions that do <u>not</u> add new root classes (i.e. classes that have a “rootElement” tagged value) are called “level B extensions”. These extensions shall set the “extension” tagged values to “levelb”. They are backwards compatible with the standard model on message level.

   Extensions that introduce new root classes are called “level C extensions” and shall set the “extensions” tagged value to “levelc”.

e) Classes belonging to an extension may not become superclasses of classes in the core model, i.e. specializations from a class from the extension to a class in the core model may not be added to the model.

f) UML Associations may be added to the extended model that have a core model class on their source end and an extension class on their target end. Thus, existing classes from the core model may become components from containers in the extensions model (class reuse), but classes from the extensions shall not become components of existing containers in the core model.

g) Data types and enumerations of the core model may be reused in extensions.

Annex A
(informative)

History and Background of DATEX II

A.1 Introduction

In the late nineties – shortly before the DATEX specifications that had been elaborated by the DATEX Task Force about five years earlier eventually became the endorsed CEN prestandards CEN ENV 13106:2000 and ENV 13777:2000 – implementers already knew about deficiencies in the DATEX specifications that made DATEX difficult and expensive to use. This usually resulted in poor performance and unreliable data exchange. The first pilot implementations had exposed these problems and systems from different vendors usually were not interoperable. At this time R&D activities like the 5th framework program project TRIDENT and the TEN-T funded project COURIER had already started looking into potential improvements, which all circulated around two basic main suggestions:

— To clearly separate the **content** data model (i.e. the traffic engineering application domain model – the **What?**) from the **data exchange** related specifications that stipulated how this information should be exchanged between software systems (the information and communication technology solution domain model – i.e. the **How?**)

— To adopt the distinction between an abstract platform independent model (PIM) and its concrete implementation(s) in (a) specific target platform(s) as (a) platform specific model(s) (PSM), which is a basic principle of the Model Driven Architecture (MDA) approach (see the website of the Object Management Group for details on MDA).

Both recommendations follow the principle of separation of concerns in order to make DATEX II more robust and more manageable, and also intend to separate the more persistent, abstract, application domain oriented specifications from the short innovation cycles of information and communication technology platforms.

A.2 Separation of payload content and exchange

TRIDENT and COURIER – the two R&D projects that first aimed at improving DATEX – had both concluded that the DATEX-Net specifications in particular – but also the DATEX dictionary in a few places – were mixing up aspects of modelling the content domain of travel and traffic related information with data constructs required by the exchange mechanisms that aimed at exchanging this information.

In the best case, the consequence of this was that the DATEX specifications became cumbersome for users, since the information they looked for from one domain was often veiled by many other regulations with relevance only for the other domain. Implementers of the data exchange mechanisms found it difficult to find the few data exchange related attributes amongst the many (200+) attributes from the traffic engineering domain. Traffic engineers and application designers sometimes stumbled over seemingly redundant attributes that looked at the same thing from the different viewpoints of the domains (i.e. there is a potentially substantial time gap between the start in time of a traffic situation on the road and the point in time that a database record was created to capture the information about this traffic situation). In the worst case, this led to non-interoperable systems.

These considerations went along with the general observation that a more specific modelling of the data model underlying DATEX data exchange was needed. The considerations about appropriate modelling technologies actually led to the obvious conclusions that more or less independent UML models should be provided for the traffic engineering content of DATEX messages, and the exchange mechanisms used to effectively and efficiently exchange this content between systems.

Since the use of UML was finally agreed as the tool to specify the payload content (and also the exchange mechanisms, although on a different level of detail), it was again a more or less obvious decision to implement both models in one UML model database, with those UML packages related to the exchange model being aware of – and actually using elements from – the payload content model, but not the other way round, i.e. the content payload modelling is entirely unaware of the exchange model).

Readers trying to confirm this relationship between the Exchange and Payload packages when looking at the final DATEX II v2.0 UML model will make the observation that on the top level, there are two more. Non-empty packages not mentioned yet: General and Management.

The General package reflects the fact that data concepts in DATEX II can be reused throughout the model. This does hold particularly throughout the various parts of the content model, where common data concepts like data types or data structures can be defined at one place and then be (re-)used throughout the model. Nevertheless, it is also possible to reuse these concepts in other packages, like for instance the Exchange package. Therefore, concepts like reusable classes, data types, enumerations and location references have been collected separately from the payload content in the General package on the top level.

During the work on the first draft of DATEX II – which was carried out by an expert consortium contracted by DG-TREN in a project called D2 – it became apparent that there was a need for some particular metadata constructs that appeared to be half-way between the Payload content and the Exchange specifications. These data concepts were mainly addressing the management of data in the client's database, based on triggers from the supplier, covering concepts well known already from DATEX like indicators for ending a situation, cancellations or for conveying the status of records related to client specific filters inside the supplier. Clearly, these concepts were not addressing traffic information as such, but rather describing the means to properly handle the information. On the other hand, they were not directly related to the exchanged artefacts of serialised content payload, i.e. they were artefacts of a higher level of abstraction than those found in the Exchange package. The final conclusion was to create a third package at the top level of the model that covers all metadata related to the Management of exchanged information.

A.3 Modelling approach: abstract specification and platform mappings

Following the recommendations from R&D (TRIDENT, COURIER), but also based on the TRIDENT assessment carried out by the DATEX Technical Committee (TC) and the input from the Centrico OTAP demonstrator, the DATEX TC recommended that DATEX II should have a rich, structured data model formally specified in UML. The DATEX Data Dictionary, which so far had contained the data concept definitions of DATEX, could then be generated automatically from the data model, e.g. via a report generator built into the tool used to maintain the model or via a software package working on the UML model exported in the interoperable XMI (XML Metadata Interchange) format. Furthermore, the data model could then be used for creating mappings of the model to specific implementation platforms via Platform Specific Models (PSM), using the Model Driven Architecture approach. This procedure has the advantage that all the required steps can be implemented as software tools and are carried out automatically, allowing for quick and cheap adaptation of the PSM whilst ensuring consistency. The experiences with DATEX had shown that manual mappings of sizable models are unmanageable and usually inevitably lead to inconsistencies.

When starting to create a DATEX II data model in UML, it quickly became apparent that UML offers a vast variety of mechanisms that would not all be required for the DATEX II modelling exercise. On the other hand UML was a fairly generic tool, and applying it for DATEX II modelling required some additional clarifications, conventions and agreements on semantics that are not to be found in general purpose UML documentation. These need to be understood by users that want to fully understand and make full use of the features offered by DATEX II, e.g. when introducing national, regional or application specific extensions to the DATEX II standardised model in an interoperable way.

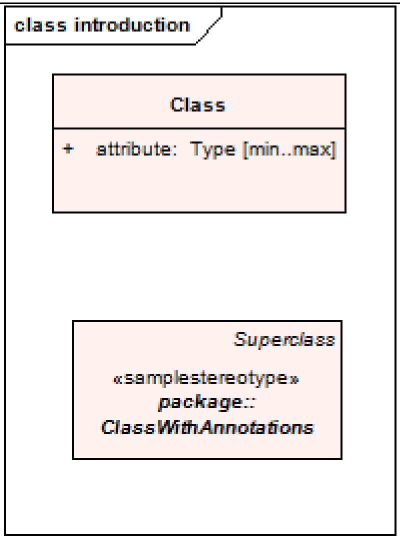
This included a UML profile for DATEX II, where a huge amount of metadata required to maintain and use DATEX II is captured in UML stereotypes and tagged values (i.e. data concepts on the meta-model layer of DATEX II). Further to this, conventions on naming, structuring/packaging and interpreting UML concepts are also required. The D2 project had created an initial input into this in a document called "UML Methodology and Modelling Constraints". This Technical Specification builds on this D2 input and extends it with further information, especially with meta-modelling constructs, conventions and agreements that were achieved when further developing the DATEX II data model and the DATEX II tools.

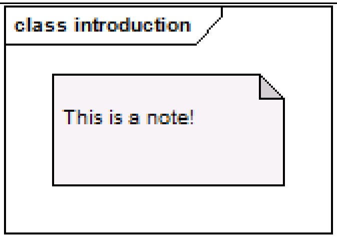
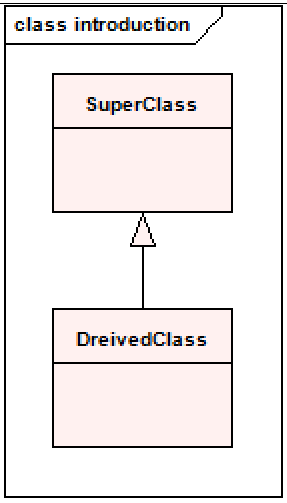
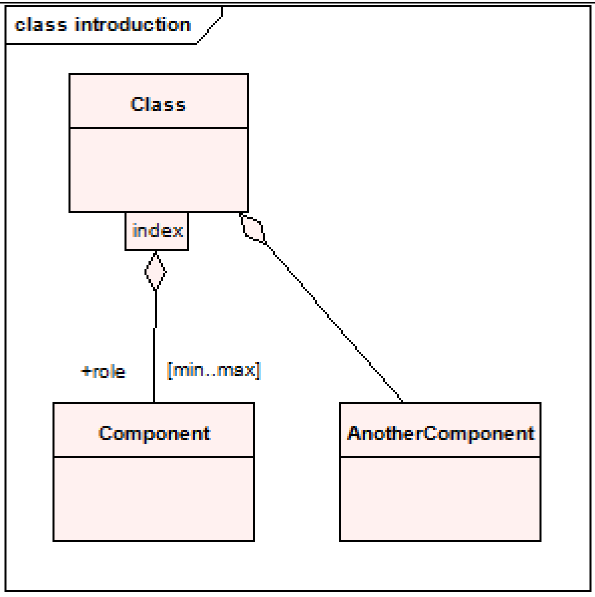
Annex B
(informative)

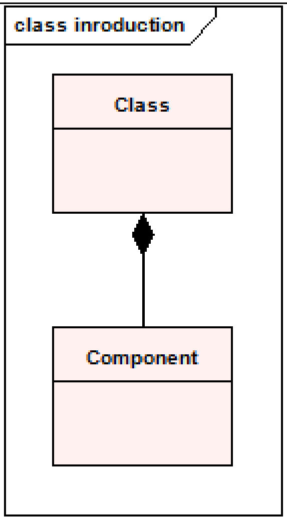
UML graphical notation

This standard makes use of a methodology to express a structural definition of the DATEX II data model called UML. The following table shows a short description of UML diagram elements used to ensure that no misinterpretation may occur caused by ongoing development of UML. UML 1.4 is standardized in ISO 19501. However, a version UML2 is currently prepared for standardisation by the Object Management Group www.omg.org.

Table B.1 — UML notation elements

Element Name	**Element**	**Description**
Class	 class introduction Class + attribute: Type [min..max] Superclass «samplestereotype» package:: ClassWithAnnotations	A class is a template for a given data element which can contain attributes. It is a rectangle divided into two compartments. The upper compartment contains the name of the class and the lower compartment contains a list of attributes owned by that class. In some diagrams, the bottom compartment of Attributes may be omitted for clarity reason. An attribute line has a specifier “+, # or –“ for the visibility (not used in this standard), a name of the attribute and after a colon a data type and in squared brackets the multiplicity (which is described in more detail in aggregation hereunder). The second class in the example depicts a class with additional metadata markup. This includes a <<stereotype>> assigned to the class, a package prefix for a class that is not defined in the package where the diagram is. A quote of a Superclass that the class is specialised from, and it shows that names of abstract classes (i.e. classes that can not be instantiated) are set in italics. Note that class names sometimes have a ∞ in their lower right corner, which is not a feature of UML but of the tools that the classes have been created with.

Element Name	Element	Description
Notes		Notes boxes are used in the diagrams to indicate a normative restriction or condition that cannot be indicated by UML standard notation. A note can apply to one or more classes or relations. Sometimes a Note box is used for extra information e.g. to tell what an "index" is (see section on Aggregations below).
Specialization / Generalization		A Specialization (i.e. Inheritance) defines the relationship of a specialised a general class (derived class) whose properties are inherited from a more general class (super class). In terms of data structures this implies that the derived class has at least the same attributes as the super class and normally will extend the state definition of the class with more attributes. The reason for using inheritance often is the capability of having different specializations from one super class. Generalization is a different name for the same relationship seen in reverse order, i.e. from the more specialised towards the more general class.
Aggregation	 class introduction Class index +role [min..max] Component AnotherComponent	The aggregation describes an owner/component relationship. The class on the side of the diamond "has" an instance of the aggregated class. The name of that instance ("role name") is given on the left side of the connection and starts again with the "+" as a specifier of visibility. On the right side the multiplicity of that instance is given as a range of the allowed count of occurrences. Optionally, the component can be addressed by an index which provides the means for the aggregation to refer to the owned element. Role, index and multiplicity are optional element, as the second depicted aggregation demonstrates.

Element Name	**Element**	**Description**
Composition		The composition strengthens the type of aggregation in a way that the lifetime of the composed element is the same as the composing class, i.e. the structure can be seen as a "composition". In data structures composition is normally seen as an embedded data element. Such a component cannot live outside the composite.

Annex C
(informative)

DATEX II Metamodel

C.1 Introduction

Defining a formal, well defined data model first of all requires a formal, well defined language to be used for the definition itself. An attempt to provide any degree of formal rigor within a specification using an informal, ambiguous language is likely to fail. In the special case of data modelling, this well known base principle of software engineering has led to a best practice where often the specification language is used "recursively". UML for example is itself defined in UML. The challenge of course is to avoid infinite recursion.

In the case of UML, the OMG has taken a deliberate decision to use four layers for specification. The bottom layer – called M0 (the letter M denotes the metalayer) – is data itself. Information about this data and its structure – i.e. the actual data model – is on layer one M1. The language to specify such a data model is the UML, which forms layer M2. The vast majority of users will never see anything further than that. Those few that have to deal with modelling principles – e.g. those intending to develop tools working with models – will have to understand the formal structure of UML itself, as it is specified in UML ISO/IEC 19501:2005. This description uses a small UML subset that forms a meta-metamodel used to define UML (and other OMG M2 standards) on layer M3.

When work on the DATEX II data model started, the engineers also had to decide on which 'language' (i.e. which metamodel) to use. As a consequence of lessons learned from the weaker, mainly text based model used for DATEX in the past, the decision was taken that the DATEX II data model should be created using UML and a UML tool. After a tool had been selected, modellers started immediately to transform the DATEX model into UML, based on a body of experience gained through projects like TRIDENT in the late nineties and OTAP in 2002/2003. It seemed as if the decision on the layer M2 choice had been taken.

What became apparent during the months of modelling work that followed was that the "use UML" principle as such actually did not answer all questions. Engineers started to develop and agree on conventions to be respected when working on the model. This included first of all the choice, which UML constructs to use. The discussion about this choice of UML constructs also fostered a better common understanding of what these constructs mean and what they should be used for, i.e. the concrete semantics of the constructs for DATEX II.

It also became quickly apparent that the standard UML constructs and features would capture many but not all aspects of the DATEX II data model. Fortunately, UML has built-in extension mechanisms that enabled the DATEX II team to extend the model with additional metadata by creating a UML profile with additional stereotypes and tagged values.

When trying to reflect on this process in order to start work on this Technical Specification – and namely part one – it became apparent that the DATEX II work had probably created its own metamodel by defining these conventions, rules and this UML profile, which then had just been expressed using UML. Obviously this M2 layer model had to be fully understood and expressed to be able to derive the provisions for this part one of this set of Technical Specifications. But how should this be done? The DATEX II team itself had never exposed this foundation of their work explicitly. Thus it was decided that the best way to go forward was to define an explicit DATEX II metamodel. Of course this would need to be expressed in UML as well, but using the same layout for exposing a M2 or higher level metamodel could be confusing to all users except for those very few familiar with metamodelling. Therefore the final decision was taken to depict this explicit M2 layer in a UML based graphical notation that would have a different look & feel than the UML tool output used to depict the data model on level M1, which would provide for a clear, visual separation of both levels.

The use of this formal M2 metamodel promised to be manifold. Firstly, this formal structure would provide a sound basis for developing software that could be used to verify compliance of models with the provisions of this Technical Specification and to map the platform independent model to concrete transfer syntax implementations, in particular to an XML Schema Definition. Secondly, the process of creating this software in itself was useful to validate the consistency and the completeness of the requirements as a feedback. The basic assumption is that most provisions for the UML model would eventually come as a requirement out of this process.

C.2 Meta-Metamodel – how to describe a metamodel?

The main question then was to decide how to express the M2 metamodel, i.e. which M3 model to use. The quoted preference for a distinct visual appearance suggested a graphical approach that would have a clear mapping to UML.

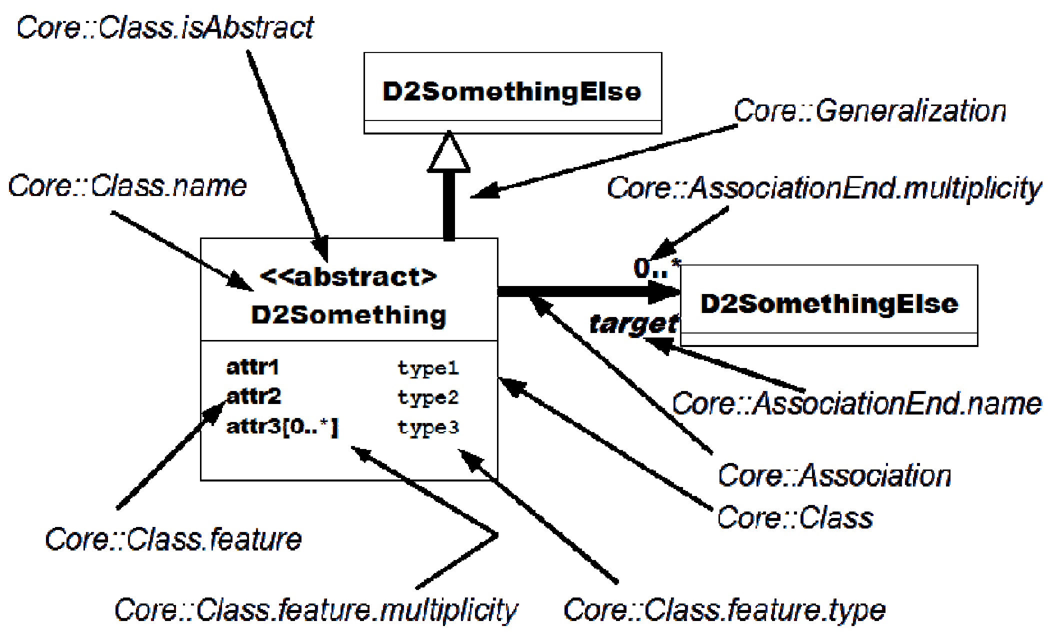


Figure C.1 – DATEX II meta-metamodel

This figure depicts the underlying M3 model as well as its mapping to UML constructs from the UML "Core" package. Essentially, this meta-metamodel consists of metaclasses that may or may not be abstract, and that may be a specialization of other metaclasses. Metaclasses may be in a directed association to other metaclasses – depicted by an arrow – which may have a multiplicity and a role name on the arrow head end. Metaclasses may have metaattributes that may have a multiplicity and that shall have a type.

C.3 The DATEX II metamodel

Using this M3 level metamodel, the DATEX II metamodel for the platform independent model on level M2 can be depicted as in the following diagram.

The DATEX II metamodel provides mainly two types of metaclasses: a component class ("D2Component") and an identifiable class ("D2Identifiable"), where the identifiable class is a direct specialization of the component class. The identifiable class is used to denote entities in DATEX II, i.e. objects that have their own identity and lifecycle, like for example one congestion, one measurement site, etc. Components are simple data structures that are only used to aggregate related pieces of information. In that sense components do not form entities or domain objects, they are simply a mechanism that can be used to structure the content model of identifiable objects and encapsulate aspects that can potentially be reused in other parts of the model. Both concepts are mapped to UML Classes.

Note that the two classes have the same value space on the level of the metamodel, the difference becomes visible on the data model level, where "D2Identifiable" is mapped to classes with either an <<identifiable>> or a <<versionedIdentifiable>> Stereotype, which again controls the generation of implementations, e.g. adds an "id" attribute in XML Schema and – for versioned classes – also a “version” attribute. The “versioned” variant is used where object retain their id during their lifecycle but have – potentially multiple – updates of their object states over time, so called “versions”.

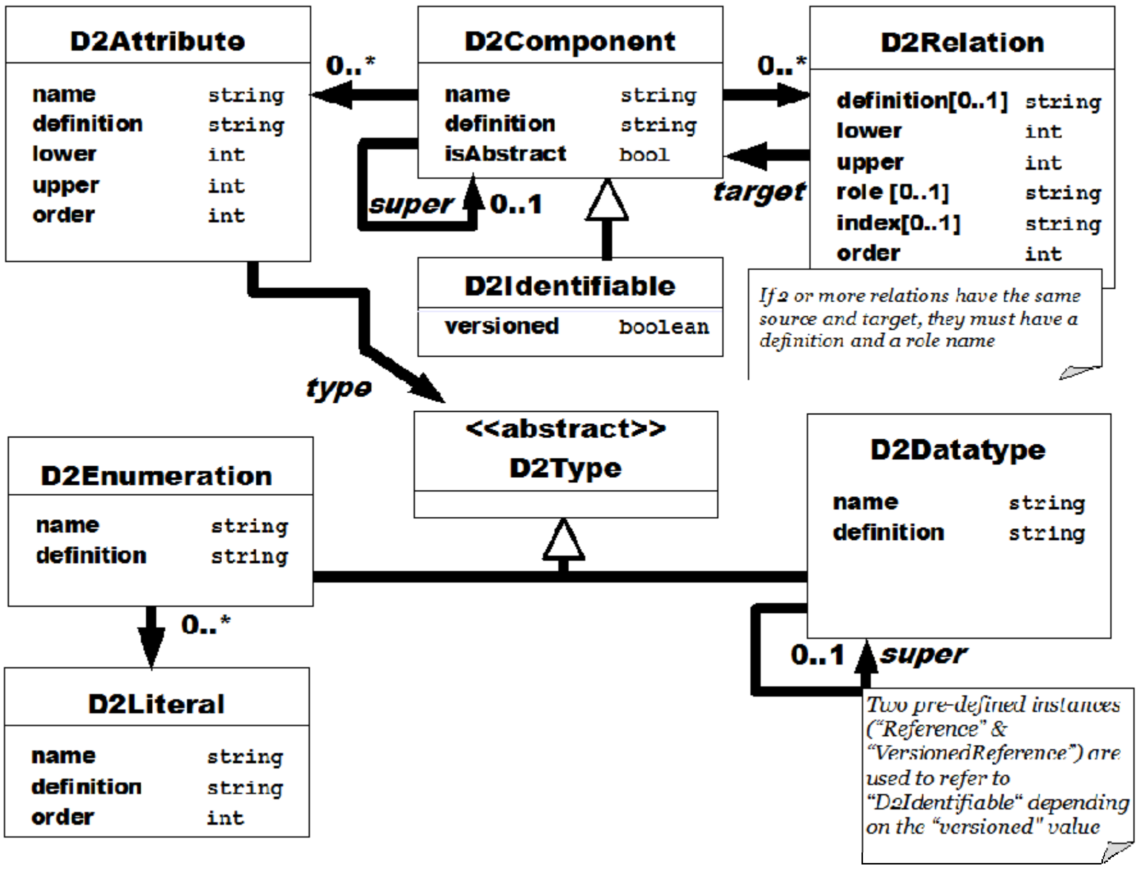


Figure C.2 - DATEX II metamodel

The relations themselves ("D2Relation") have additional metadata assigned to them. It is mandatory to provide a range for multiplicity ("lower", "upper") and an "order". Note that this does not mean that the UML specification must have explicit multiplicity. In cases where no UML multiplicity is explicitly specified, a default value of "1" is used. The "order" relates to multiple associations connected to the same 'whole' metaclass. In this case, the "order" attribute values (distinct non-negative integers) govern the sequence order in which the different 'part' data structures appear in the serialized 'whole' object state. Relations may furthermore have three optional attributes: a "definition", a "role" name and an "index". Note that in some cases these attributes become mandatory, especially in cases where default values (if neither "role" nor "qualifier" is provided, a default value derived from the target class name by turning the first letter to lower case is assumed) would be ambiguous.

Both, the "D2Component" and the "D2Identifiable" metaclass may also have attributes – modelled as a metaclass called "D2Attribute" – which themselves have to have a set of metaattributes. These consist of a name and a definition, a multiplicity stated as lower and upper bound and an order. Except for the name metaattribute these metaattributes have the same semantics as in the "D2Relation" metaclass, just that all are mandatory. This implies especially that for attributes, a definition has to be provided in all cases. As in the case of "D2Relation", the (mandatory) multiplicity of attributes is either determined from the (optional) UML multiplicity, or it is set to 1..1 by default.

Types in the DATEX II metamodel are represented by two metaclasses: "D2Enumeration" and "D2Datatype". Data types in DATEX II have only a definition and a name in the metamodel, i.e. the DATEX II metamodel does not actually model the value space of the data type. The rationale behind this design is that platform mappings to concrete implementation platforms should be free to use the most appropriate representation of the type in the particular target platform, and not be constrained by structural regulations from the platform independent model. In the case of enumerations, the intended mapping to string literals seems to be sufficiently stable and valid across all platforms that the metamodelling of this very important data type can be explicit, based on defining the metaclass "D2Literal" with attributes "name", "definition" and "order" which have the same semantics as in the other metaclasses of the metamodel.

The described metaclasses and their metaattributes and relations are mapped to UML in the following way:

— The DATEX II metaclasses "D2Component", "D2Identifiable" and "D2Datatype" are mapped to the UML metaclass "Class" in the "Foundation:Core" package. Their "name" metaattribute is mapped to the "name" metaattribute of "Class". Their "definition" is mapped to an instance of the UML "TaggedValue" metaclass from the "Foundation:Extension Mechanism" package with "TaggedValue.name" being fixed to "definition" and "TaggedValue.dataValue" containing the definition of the corresponding metaclass, as contained in their "definition" metaattribute.

— The "isAbstract" metaattribute of "D2Component" and "D2Idenifiable" is mapped to the metaattribute of the same name in UML Class.

— The distinction between "D2Component" and "D2idenitfiable" is mapped to an instance of the UML "Stereotype" metaclass from the "Foundation:Extension Mechanism" package with "name" equal to either "identifiable" or “versionedIdentifiable”. Note that "D2Component" classes are mapped to UML classes without any of the stereotypes known to and governed by this specification.

— The "D2Relation" metaclass is mapped to UML "Association" in "Foundation:Core". The following table summarises the mapping of the metaattributes of this metaclass:

Table C.1 — Mapping of “D2Relation” attributes

Attribute	**Mapping**
definition	"Foundation:Extension Mechanism:TaggedValue.dataValue" (with "Foundation:Extension Mechanism:TaggedValue.name" set to "definition")
lower	"Foundation:Core:AssociationEnd.multiplicity.range.lower"
upper	"Foundation:Core:AssociationEnd.multiplicity.range.upper"
role	"Foundation:Core:AssociationEnd.name"
index	"Foundation:Core:AssociationEnd.qualifier.name"
order	"Foundation:Extension Mechanism:TaggedValue.dataValue" (with "Foundation:Extension Mechanism:TaggedValue.name" set to "order")

— The "D2Attribute" metaclass is mapped to UML "Attribute" in "Foundation:Core". The following table summarises the mapping of the metaattributes of this metaclass:

Table C.2 — Mapping of "D2Attribute" attributes

Attribute	**Mapping**
name	"Foundation:Core:Attribute.name"
definition	"Foundation:Extension Mechanism:TaggedValue.dataValue" (with "Foundation:Extension Mechanism:TaggedValue.name" set to "definition")
lower	"Foundation:Core:Attribute.multiplicity.range.lower"
upper	"Foundation:Core:Attribute.multiplicity.range.upper"
order	"Foundation:Extension Mechanism:TaggedValue.dataValue" (with "Foundation:Extension Mechanism:TaggedValue.name" set to "order")

— The "D2Enumeration" metaclass is mapped to UML "Enumeration" in "Foundation:Core". Its "name" metaattribute is mapped to the "name" metaattribute of "Enumeration". Its "definition" is mapped to an instance of the UML "TaggedValue" metaclass from the "Foundation:Extension Mechanism" package with "TaggedValue.name" being fixed to "definition" and "TaggedValue.dataValue" containing the definition of the corresponding metaclass, as contained in their "definition" metaattribute.

— The "D2Literal" metaclass is mapped to UML "EnumerationLiteral" in "Foundation:Core". Its "name" metaattribute is mapped to the "name" metaattribute of "EnumerationLiteral". Its "definition" is mapped to an instance of the UML "TaggedValue" metaclass from the "Foundation:Extension Mechanism" package with "TaggedValue.name" being fixed to "definition" and "TaggedValue.dataValue" containing the definition of the corresponding metaclass, as contained in their "definition" metaattribute. Its "order" is mapped to another instance of the UML "TaggedValue" metaclass from the "Foundation:Extension Mechanism" package with "TaggedValue.name" being fixed to "order" and "TaggedValue.dataValue" containing the order value of the corresponding metaclass, as contained in their "order" metaattribute.

— The following table lists the mappings of the various metaclass relationships in the DATEXI II metamodel. Note that only the metamodel associations are listed here. Generalizations on the metamodel level are seen as having semantics on the layer of the meta-metamodel, and thus are not mapped to UML constructs on this layer.

Table C.3 — Mapping of relationships

Relation	**Mapping**
"D2Component"-"D2Component"	"Foundation:Core:Generalization"
"D2Datatype"-"D2Datatype"	"Foundation:Core:Generalization"
"D2Enumeration"-"D2Literal"	"Foundation:Core:Enumeration.literal"
"D2Component"-"D2Attribute"	"Foundation:Core:Class.feature"
"D2Relation"-"D2Component"	"Foundation:Core:AssociationEnd.participant"
"D2Component"-"D2Relation"	"Foundation:Core:Class.association"
"D2Attribute"-"D2Type"	"Foundation:Core:Attribute.type"

Annex D
(normative)

XML Schema Definition Mapping

D.1 General

The previous annex provided background information on how the DATEX II data model has been created based on an explicit presentation of the underlying metamodel. The constructs of this metamodel – amended by additional metadata for a platform specific model – govern the mapping of models to XML Schema Definitions (XSD). This normative annex describes how the normative schemas used in all parts of this Technical Specification are actually created from a UML model conform to Clauses 6 and 7 of this specification.

The XSD mapping is basically covered by a small set of general rules, which are then amended / extended by further detailed mechanisms that handle specific cases. These principle rules are described in the rest of this normative annex.

D.2 Platform specific model rules for XML with XML schema definition

D.2.1 General

A model that needs to be mapped to XML Schema Definition must fulfil additional requirements on top of those specified in Clauses 6 and 7. These additional requirements are captured in this annex. Conformant models fulfilling these requirements can be mapped to a corresponding XML Schema Definition according to the mapping described in subsequent sections.

D.2.2 Numbered list of requirements

a) UML Classes with a “datatype” UML Stereotype assigned may have a “schemaType” UML TaggedValue or they may have a “schemaTypeInclude” UML TaggedValue. They may also have none of these two.

b) UML Classes with a “datatype” UML Stereotype assigned may not have a “schemaType” UML TaggedValue and a “schemaTypeInclude” UML TaggedValue at the same time.

c) The value of a “schemaType” UML TaggedValue shall be an XML Schema Definition built-in simple type, i.e. it shall follow the following production rule:

```
schema-type ::= "duration" , "dateTime" , "time" , "date" , "gYearMonth" ,
"gYear" , "gMonthDay" , "gDay" , "gMonth" , "boolean" , "base64Binary" ,
"hexBinary" , "float" , "double" , "anyURI" , "QName" , "NOTATION" , "string" ,
"decimal" , "normalizedString" , "token" , "language" , "Name" , "NMTOKEN" ,
"NCName" , "NMTOKENS" , "ID" , "IDREF" , "IDREFS" , "ENTITY" , "ENTITIES" ,
"integer" , "nonPositiveInteger" , "long" , "nonNegativeInteger" ,
"negativeInteger" , "int" , "unsignedLong" , "positiveInteger" , "short" ,
"unsignedInt" , "byte" , "unsignedShort" , "unsignedByte”
```

d) UML Classes with a “datatype” UML Stereotype that have neither a “schemaType” nor a “schemaTypeInclude” UML TaggedValue themselves shall have an ancestor class (i.e. a direct UML Generalization, UML Generalization of its UML Generalization, etc.) that has either a “schemaType” or a “schemaTypeInclude” UML TaggedValue.

e) The “dataValue” of a “schemaTypeInclude” UML TaggedValue shall be a URI that uniquely denotes an XML schema type definition for a type of the same name as the UML class that the “schemaTypeInclude” UML TaggedValue is assigned to.

   Note that classes whose “schemaTypeInclude” UML TaggedValue points to a definition of a complex type for the name of the UML Class shall not have specializations.

f) UML Classes with a “datatype” UML Stereotype assigned may have a “facets” TaggedValue, but only if they are mapped to an XML schema simple type. The “dataValue” of this UML TaggedValue shall be a valid content for the restriction element of an XML schema simple type definition of a type restriction of the XML schema type the UML Class is mapped to.

   Note that the mapping type defines which facets are allowed according to the XML Schema specifications.

g) UML Classes that do not have a “datatype” UML Stereotype assigned may have a “rootElement’ UML TaggedValue.

   Note that UML Enumerations and UML Classes that do have a “datatype” UML Stereotype assigned shall not have a “rootElement” UML TaggedValue.

h) Attributes of UML Classes that do not have a “datatype” UML Stereotype assigned may have an “attribute” UML TaggedValue, which may have a “dataValue” of either “yes” or “no”, but only if their type is mapped to an XML schema simple type and if their multiplicity is either "0..1" or "1..1".

i) Attributes of UML Classes that do not have a “datatype” UML Stereotype assigned may have a “schemaName” UML TaggedValue.

j) Extending subclause 7.2 m)) for all UML Classes that do not have a “datatype” UML Stereotype assigned, the following UML constructs shall be unique:

   — UML Attribute “names”

   — “names” of remote UML AssociationEnds of UML Associations connected to the class

   — “qualifiers” of UML AssociationEnds directly connected to the UML Class

   — names of UML Classes connected via the source side of an UML Association without an UML AssociationEnd name on the source end – with their UML Class name’s first letter turned to lower case – plus

   — “dataValues” of “schemaName” UML TaggedValues of attributes of this UML Class

   This means that none of the listed names, qualifiers or “schemaName” values shall be lexically equal.

D.3 Mapping the PSM to XML Schema Definition

D.3.1 Mapping of "D2Datatype"

“D2Datatype” classes are in principle mapped to XSD type definitions of the same name as the class. Exceptions – i.e. cases where a “D2Datatype” class is mapped to something else – do exist and are described below. The value of the class' "definition" metaattribute is mapped to an XML Schema “annotation” element, i.e. the principle structure looks like this:

```
<xs:simpleType name="ClassNameFromUMLModel">
  <xs:annotation>
    <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
    </xs:annotation>
    <xs:restriction base="RestrictionBase" />
  </xs:simpleType>
```

The value of “RestrictionBase” is determined according to the following algorithm:

1) If the UML Class has got a “schemaType” (ref. D.2.2 a)) tagged value, this value shall be an XML Schema pre-defined simple type from the list provided in (ref. D.2.2 c)), the value of “RestrictionBase” is this XML Schema pre-defined type name.

2) If the UML Class has neither got a “schemaType” nor a “schemaTypeInclude” tagged value, the value of “RestrictionBase” is the name of the class' superclass. Note that D.2.2 d) ensures that such a class does have a superclass.

The XML Schema type definitions generated by this rule may be modified by providing a “facets” tagged value. In this case, the “xs:restriction” element will be expanded and the content of the 'facets' tagged value will be inserted between the opening and the closing element:

```
    <xs:restriction base="RestrictionBase" >
      ContentOfFacetsTaggedValueHere
    </xs:restriction>
```

Note that according to D.2.2 f) the content of this tagged value shall be a valid content model for the “xs:restriction” element.

This principle mapping rule for “D2Datatype” classes may be overridden by providing a user supplied XML Schema type mapping via a sub-schema provided via the “schemaTypeInclude” tagged value – which is helpful in cases where intended type definitions do not fit into the scheme described above. The content of the user supplied type definition is not captured in the tagged value itself but at a different place that is uniquely denoted by a URI contained in the tagged values. This may be either by treating the tagged value as a URL – trying to load the content from there – or by another means that allows providing external XML Schema type definitions (e.g. as file from local disk).

D.3.2 Mapping of "D2Enumeration" and "D2Literal"

“D2Enumeration” classes – and their corresponding “D2Literal” classes – are always mapped to an XML Schema simple type definition with the following structure:

```
  <xs:simpleType name="EnumerationNameFromUMLModel">
    <xs:annotation>
      <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="EnumerationLiteralNameFromUMLModel">
        <xs:annotation>
          <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      ...
    </xs:restriction>
  </xs:simpleType>
```

D.3.3 Mapping of "D2Component"

There are two different ways of mapping “D2Component” classes to an XML Schema complex type definition.

D.3.3.1 "D2Component" classes without superclass

First, those “D2Component” classes that do not have a superclass are mapped in principle to the following structure:

```
<xs:complexType name="ClassNameFromUMLModel">
  <xs:annotation>
    <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="NameFromUMLModel" type="TypeFromUMLModel" minOccurs="LowerBound"
                 maxOccurs="UpperBound">
      <xs:annotation>
        <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
      </xs:annotation>
    </xs:element>
    …
  </xs:sequence>
</xs:complexType>
```

The name of the generated XML Schema type is taken from the “name” metaattribute of the UML Class, the content of the “xs:documentation” element is taken from the (mandatory – see 7.2 a)) “definition” tagged value. The “xs:element” entries in the sequence are generated from:

1) The UML Attributes specified in the UML Class and appearing in the order of their “order” tagged values.

2) The UML Associations that are connected to the UML Class, again appearing in the order of the “order” tagged values of the association ends connected to the “D2Component” class.

3) A single extension element at the very end.

Instances of the "D2Attribute" (UML Attributes) and "D2Relation" (UML Associations) classes are mapped to the following structure:

```
<xs:element name="NameFromUMLModel" type="D2LogicalModel:TypeFromUMLModel"
            minOccurs="LowerBound" maxOccurs="UpperBound">
  <xs:annotation>
    <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
  </xs:annotation>
</xs:element>
```

If lower or upper bounds are not provided, a default of “1” is assumed. Note that “xs:annotation” elements may be omitted in cases where they are not required by rules and not provided by the model (e.g. in case of a single association between two “D2Component” classes).

The “NameFromModel” is created for “D2Attributes” according to the following hierarchy:

4) The value of the “schemaName” tagged value, if present.

5) The value of the “name” metaattribute of the “D2Attribute” class.

The “NameFromModel” is created for “D2Relations” according to the following hierarchy:

6) The “name” of the remote UML AssociationEnd connected to the class, if present.

7) The “name” of the UML Class connected on the other side of a UML Association, with its UML Class name turned to lower case (by turning the first character or any prefixing acronym to lower case).

“TypeFromUMLModel” is the “name” of either the associated “D2Enumeration” or the associated “D2DataType” for “D2Attributes”. For “D2Relations” it is by default the “name” of the associated class, with a single deviation from this rule in case the UML AssociationEnd metaclass directly connected to the UML Class has a “qualifier” UML Attribute. Note that in this particular case “LowerBound” is hardcoded to “0” and “UpperBound” is hardcoded to “unbounded”.

In this case, the type name is created as “_<name>” where <name> is determined according to the following hierarchy:

8) The “name” of the remote UML AssociationEnd connected to the class – turned to UCC – if present.

9) A concatenation of the “name” of the UML Class plus the “name” of the “qualifier” UML Attribute plus the “name” of the UML Class connected on the other side of the UML Association.

A definition for this type is then also added to the schema as:

```
<xs:complexType name="_<name>">
  <xs:sequence>
    <xs:element name="<name1>" type="<name2>" minOccurs="1" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="<nameFromUMLQualifier>" type="xs:int" use="required" />
</xs:complexType>
```

where <name2> is the the “name” of the UML Class connected on the other side of the UML Association, whereas <name1> is the same name turned to LCC.

The structure of the final extension element is:

```
    <xs:element name="ClassNameToLCC+'Extension'" type="D2LogicalModel:_ExtensionType"
                minOccurs="0" />
```

Again, the name of this element is generated by taking the class' name and turning it to lower camel case (by turning the first character to lower case), and then appending the fixed string “Extension”.

“D2Attribute” instances may alternatively be mapped to an XML Attribute by setting an “attribute” tagged value to “yes”. In this case, the structure is extended by adding “xs:attribute” elements to the content model.

```
<xs:complexType name="ClassNameFromUMLModel">
  <xs:annotation>
    ...
  </xs:annotation>
  <xs:sequence>
    ...
  </xs:sequence>
  <xs:attribute name=”AttributeNameFromUMLModel” type=”TypeFromUMLModel”
              use=”required”/>
    ...
</xs:complexType>
```

The 'use=”required”' is set if multiplicity is 1..1 and omitted in case of 0..1. Note that other multiplicities are not allowed and that the attribute's mapped type must be an XML Schema simple type according to D.2.2 h).

D.3.3.2 "D2Component" classes with superclass

Those “D2Component” classes with a superclass are mapped in a similar way, just that they are mapped to extensions of their superclass.

```
<xs:complexType name="ClassNameFromUMLModel">
  <xs:annotation>
    <xs:documentation>ContentOfDefinitionTaggedValueFromUMLModel</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="D2LogicalModel:NameOfSuperclassFromUMLModel">
      <xs:sequence>
      ...
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

D.3.4 Mapping of "D2Identifiable" classes

“D2Identifiable” classes are mapped in the same way as “D2Component” classes, just that they have the following additional attribute definition.

```
<xs:attribute name="id" type="xs:string" use="required" />
```

In case of “versioned” being “true”, a second attribute is created for the version number.

```
<xs:attribute name="version" type="xs:string" use="required" />
```

Besides the type itself, an Identity-constraint definition will be created that ensures uniqueness of instances of the type, depending on “id” or “id”+”version”, respectively. This constraint definition has the following structure (example for the versioned case including a “version” attribute)

```
<xs:unique name="_"{targetClass}"Constraint">
    <xs:selector xpath=".//"{targetClass} />
    <xs:field xpath="@id" />
    <xs:field xpath="@version" />
</xs:unique>
```

Furthermore, corresponding (typed) referencing types are created, that allow reference to elements of these types, based on common untyped reference types (“Reference” & “VersionedReference”).

```
<xs:complexType name="Reference">
    <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
```

and

```
<xs:complexType name="VersionedReference">
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="version" type="xs:string" use="required"/>
</xs:complexType>
```

Depending on whether the target type for a class {targetClass} is versioned or not, the corresponding reference type looks like this

```
<xs:complexType name=”_”{targetClass}"Reference">
   <xs:complexContent>
      <xs:extension base="D2LogicalModel:Reference">
         <xs:attribute name="targetClass" use="required" fixed={targetClass}/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

or like this

```
<xs:complexType name="_"{targetClass}"VersionedReference">
   <xs:complexContent>
      <xs:extension base="D2LogicalModel:VersionedReference">
         <xs:attribute name="targetClass" use="required" fixed={targetClass}/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The {targetClass} is determined by setting a “targetClass” tagged value on the attribute that implements the reference.

D.3.5 XML elements

For all “D2Component” or “D2Identifiable” classes that have a “rootElement” tagged value, a top level XML element declaration is generated that has the following structure:

```
<xs:element name="d2LogicalModel" type="D2LogicalModel:D2LogicalModel" />
```

The example shows the “d2LogicalModel” element which is mandatory according to 8.2 b), but other top level elements may be declared in the same way, i.e. the “rootElement” content is used for the “name” attribute and the class name itself is used for the “type” attribute.

For classes that have a “modelBaseVersion” tagged value (required for “D2LogicalModel” according to 8.2 c)) an attribute of the same name is created with the following structure:

```
<xs:attribute name="modelBaseVersion" use="required" fixed="xxx" />
```

where the content of the “fixed” XML attribute (“xxx”) is determined from the tagged value.

D.3.6 Extension mapping

There is a special deviation from the mapping presented so far for mapping a level B extended model to XML Schema. The only difference in the mapping is for specializations that cross the extension border, i.e. specializations where the superclass is in the core model and the subclass is in the extension. Clause 9 requires that this situation is determined by a superclass / subclass pair where the subclass has an “extension” tagged value being set to “levelb” and the superclass has not.

In these cases, the specialization is not mapped to an extension, but the extension element of the superclass is replaced by an XML element of the same name that is of an internal complex type which consists of a sequence of XML elements that represent the – potentially – multiple classes that extend this superclass in the extensions.

This means that

```
      <xs:element name="someClassExtension" type="D2LogicalModel:_ExtensionType"
                  minOccurs="0" />
```

is replaced by

```
      <xs:element name="someClassExtension type="D2LogicalModel:_SomeClassExtensionType"
minOccurs="0" />

Where "SomeClassExtensionType" is defined as:
        <xs:complexType name="_SomeClassExtensionType">
          <xs:sequence>
            <xs:element name="classA" type="D2LogicalModel:ClassA" minOccurs="0" />
            <xs:element name="classB" type="D2LogicalModel:ClassB" minOccurs="0" />
            ...
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>

          </xs:sequence>
        </xs:complexType>
```

D.3.7 Overall document structure and namespaces

The whole set of type and element definitions provided so far in this section is finally wrapped into the following XML structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
           xmlns:D2LogicalModel="http://datex2.eu/schema/2/2_0"
           targetNamespace="http://datex2.eu/schema/2/2_0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">
  …
</xs:schema>
```

The target namespace for this version of DATEX II is `http://datex2.eu/schema/2/2_0`. The first version number indicates the version of the model, the second number the version of the schema generation tool that was used to generate the schema. The first version number is taken from the "modelBaseVersion" tagged value of the top level element, the seconded is fixed for a particular version of the mapping (e.g. hard coded in a software tool).

If a model contains extension classes with their "extension" tagged value set to "levelb", the namespacve is retained, i.e. Level B extension have the same namespace as the Level A model.

If a model contains extension classes with their "extension" tagged value set to "levelc", the generated schema is not allowed to use the same namespace than Level A. The definition of the new namespace is not goverened by this specification, e.g. it can be an input field in a schema generation tool's configuration screen.

Note that model elements reused from Level A in Level C extensions will therefore have a new namespace and will become different implementation classes after data binding. Modellers that want to reuse code from Level A implementations in an extended model/schema are therefore encouraged to consider a Level B extension.

Annex E
(normative)

General structure element

E.1 Package "D2LogicalModel"

Table E.1 defines those packages, classes and their attributes – as well as any tagged values assigned to these – that are mandatory in the (single top-level) package "D2LogicalModel".

Table E.1 — Mandatory metadata content of the package "D2LogicalModel"

Name	Designation (Type)	Definition	Datatype	Multiplicity
D2LogicalModel	DATEX II logical model (Class)	The DATEX II logical model comprising exchange, content payload and management sub-models.	—	—
rootElement	Root element (Tagged Value of Class)	**UML TaggedValue** with "dataValue" equal to "d2LogicalModel"; this is the standard entry point into an XML coded instance of the DATEX II model.	String (implicit)	—
modelBaseVersion	Model Base version (Tagged Value of Class)	**UML TaggedValue** with "dataValue" that corresponds to the DATEX II model version identifier.	String (implicit)	—
extensionName	Extension name (Tagged Value of Class)	**UML TaggedValue** with "dataValue" providing the name of the extension(s) contained in the model.	String (implicit)	—
extensionVersion	Extension version (Tagged Value of Class)	**UML TaggedValue** with "dataValue" providing the corresponding version identifier.	String (implicit)	—
General	— (Package)	Package which contains the actual DATEX II domain model, i.e. reusable classes, enumerations and datatypes.	—	—
PayloadPublication	— (Package)	This package contains the different publications that can be exchanged via a DATEX II interface.	—	—

E.2 Package "PayloadPublication"

Table E.2 defines those packages, classes and their attributes – as well as any tagged values assigned to these – that are mandatory in the package "PayloadPublication".

Table E.2 — Mandatory metadata content of the package “PayloadPublication”

Name	Designation (Type)	Definition	Datatype	Multiplicity
PayloadPublication	Payload publication (Class)	A payload publication of traffic related information or associated management information created at a specific point in time that can be exchanged via a DATEX II interface.	—	—
defaultLanguage	Default language (Attribute)	The default language used throughout the payload publications, specified by an ISO 639-2 3-alpha code.	Language	1
feedDescription	Feed description (Attribute)	A description of the information which is to be found in the publications originating from the particular feed (URL).	MultilingualString	0..1
feedType	Feed type (Attribute)	A classification of the information which is to be found in the publications originating from the particular feed (URL). Different URLs from one source may be used to filter the information which is made available to clients (e.g. by type or location).	String	0..1

publicationTime	Publication time (Attribute)	Date/time at which the payload publication was created.	DateTime	1
publicationCreator	Creator of the publication (Association)	This association describes the entity that has created the publication by providing the country and an identifier that is supposed to be unique within this country.	—	1
InternationalIdentifier	International identifier (Class)	An identifier/name whose range is specific to the particular country.	—	—
country	Country ID	ISO 3166-1 two character country code.	CountryEnum Defined Literals: be; bg; ch; cs; cy; cz; de; dk; ee; es; fi; fo; fr; gb; gg; gi; gr; hr; hu; ie; im; is; it; je; li; lt; lu; lv; ma; mc; mk; mt; nl; no; pl; pt; ro; se; si; sk; sm; tr; va; other;	1
nationalIdentifier	National Identifier	Identifier or name unique within the specified country.	String	1

E.3 Package "General"

Table E.3 defines the different attributes and roles defined in the package "General".

Table E.3 — Classes, attributes and roles of the package General

Name	Designation	Definition	Type	Multiplicity
DataTypes	— (Package)	A collection of information describing data types that are reused elsewhere in the DATEX II model.	—	—

E.4 Package "DataTypes"

Table E.4 defines the different attributes and roles defined in the package "DataTypes"

Table E.4 — Classes, attributes and roles of the package DataTypes

Name	Designation	Definition	XSD mapping	Multiplicity
Generic	— (Package)	A collection of generic data type descriptions. These are, in general, mathematical concepts or widely used computer science concepts, without specific specified units.	—	—

E.5 Package "Generic"

Table E.5 defines the different attributes and roles defined in the package "Generic".

Table E.5 — Classes, attributes and roles of the package Generic

Name	Designation	Definition	XSD mapping
Boolean	— (Datatype)	Boolean has the value space required to support the mathematical concept of binary-valued logic: {true, false}.	xs:boolean
Date	— (Datatype)	A combination of year, month and day integer-valued properties plus an optional timezone property. It represents an interval of exactly one day, beginning on the first moment of the day in the timezone, i.e. '00:00:00' up to but not including '24:00:00'.	xs:date
DateTime	— (Datatype)	A combination of integer-valued year, month, day, hour, minute properties, a decimal-valued second property and a timezone property from which it is possible to determine the local time, the equivalent UTC time and the timezone offset from UTC.	xs:dateTime
Float	— (Datatype)	A floating point number whose value space consists of the values m × 2^e, where m is an integer whose absolute value is less than 2^24, and e is an integer between -149 and 104, inclusive.	xs:float
Integer	— (Datatype)	An integer number whose value space is the set {-2147483648, -2147483647, -2147483646, ..., -2, -1, 0, 1, 2, ..., 2147483645, 2147483646, 2147483647}.	xs:integer
Language	— (Datatype)	A language datatype, identifies a specified language.	xs:language
MultilingualString	— (Datatype)	A multilingual string, whereby the same text may be expressed in more than one language.	see MultilingualString.xsd
NonNegativeInteger	— (Datatype)	An integer number whose value space is the set {0, 1, 2, ..., 2147483645, 2147483646, 2147483647}.	xs:nonNegativeInteger
Reference	— (Datatype)	A reference to an identifiable managed object where the identifier is unique. It comprises an identifier (e.g. GUID) and a string identifying the class of the	see Reference.xsd

Name	Designation	Definition	XSD mapping
		referenced object.	
String	— (Datatype)	A character string whose value space is the set of finite-length sequences of characters. Every character has a corresponding Universal Character Set code point (as defined in ISO/IEC 10646), which is an integer.	xs:string
Time	— (Datatype)	An instant of time that recurs every day. The value space of time is the space of time of day values as defined in § 5.3 of [ISO 8601]. Specifically, it is a set of zero-duration daily time instances.	xs:time
Url	— (Datatype)	A Universal Resource Locator (URL) address comprising a compact string of characters for a resource available on the Internet.	xs:anyURI
VersionedReference	— (Datatype)	A reference to an identifiable version managed object where the combination of the identifier and version is unique. It comprises an identifier (e.g. GUID), a version (NonNegativeInteger) and a string identifying the class of the referenced object.	see VersionedReference.xsd

Content of file MultilingualString.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:D2LogicalModel="http://datex2.eu/schema/2/2_0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://datex2.eu/schema/2/2_0"
           elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:complexType name="MultilingualStringValue">
      <xs:simpleContent>
         <xs:extension base="D2LogicalModel:MultilingualStringValueType">
            <xs:attribute name="lang" type="xs:language"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="MultilingualString">
      <xs:sequence>
         <xs:element  name="value"  type="D2LogicalModel:MultilingualStringValue"
maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="MultilingualStringValueType">
      <xs:restriction base="xs:string">
         <xs:maxLength value="1024"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

Content of file Reference.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                        xmlns:D2LogicalModel="http://datex2.eu/schema/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://datex2.eu/schema/"         elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:complexType name="Reference">
      <xs:attribute name="id" type="xs:string" use="required"/>
   </xs:complexType>
</xs:schema>
```

Content of file VersionedReference.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                        xmlns:D2LogicalModel="http://datex2.eu/schema/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://datex2.eu/schema/"         elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:complexType name="VersionedReference">
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="version" type="xs:string" use="required"/>
   </xs:complexType>
                                             </xs:schema>
```

Bibliography

[1] Extensible Markup Language (XML) 1.0 (Fifth Edition); W3C Recommendation 26 November 2008 (see www.w3.org/TR/2008/REC-xml-20081126)

[2] XML Schema Part 1: Structures Second Edition; W3C Recommendation 28 October 2004 (see www.w3.org/TR/xmlschema-1)

[3] XML Schema Part 2: Datatypes Second Edition; W3C Recommendation 28 October 2004 (see www.w3.org/TR/xmlschema-2)

[4] Uniform Resource Identifier (URI): Generic Syntax; IETF Standards Track, January 2005 (http://tools.ietf.org/html/rfc3986)

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™