INTERNATIONAL STANDARD

ISO/IEC 15938-2

First edition
2002-04-01

Information technology — Multimedia content description interface —

Part 2: Description definition language

Technologies de l'information — Interface de description du contenu multimédia —

Partie 2: Langage de définition de description (DDL)

Reference number
ISO/IEC 15938-2:2002(E)

© ISO/IEC 2002

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2002

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

© ISO/IEC 2002 – All rights reserved

Contents

Page

Foreword iv

Introduction v

1 Scope 1
1.1 Scope of this International Standard 1
1.2 Overview of Description Definition Language 1
1.3 Structure of this document 2

2 Normative references 2

3 Terms and definitions 4
3.1 Conventions 4
3.2 Wrapper of the schema 5
3.3 Terminology 5

4 Symbols and abbreviated terms 7

5 MPEG-7-specific Extensions 8
5.1 MPEG-7 Structural Extensions 8
5.2 MPEG-7 Datatype Extensions 11

6 Unused XML Schema Features 14

Annex A (informative) **Overview of XML Schema Structural Components** 15
A.1 Introduction 15
A.2 Detailed Description 15

Annex B (informative) **Overview of XML Schema Datatypes** 29
B.1 Introduction 29
B.2 Detailed Description 29

Annex C (normative) **XML Schema for MPEG-7 Extensions** 34

Annex D (informative) **Patent statements** 35

Bibliography 37

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 15938-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 15938 consists of the following parts, under the general title *Information technology — Multimedia content description interface*:

— *Part 1: Systems*

— *Part 2: Description definition language*

— *Part 3: Visual*

— *Part 4: Audio*

— *Part 5: Multimedia description schemes*

— *Part 6: Reference software*

— *Part 7: Conformance testing*

— *Part 8: Extraction and use of MPEG-7 descriptions*

Annex C forms a normative part of this part of ISO/IEC 15938. Annexes A, B and D are for information only.

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

This standard, also known as "Multimedia Content Description Interface", provides a standardized set of technologies for describing multimedia content. The standard addresses a broad spectrum of multimedia applications and requirements by providing a metadata system for describing the features of multimedia content.

The following are specified in this standard:

- **Description Schemes (DS)** describe entities or relationships pertaining to multimedia content. Description Schemes specify the structure and semantics of their components, which may be Description Schemes, Descriptors, or datatypes.
- **Descriptors (D)** describe features, attributes, or groups of attributes of multimedia content.
- **Datatypes** are the basic reusable datatypes employed by Description Schemes and Descriptors.
- **Description Definition Language (DDL)** defines Description Schemes, Descriptors, and Datatypes by specifying their syntax, and allows their extension.
- **Systems tools** support delivery of descriptions, multiplexing of descriptions with multimedia content, synchronization, file format, and so forth.

This standard is subdivided into eight parts:

Part 1 – Systems: specifies the tools for preparing descriptions for efficient transport and storage, compressing descriptions, and allowing synchronization between content and descriptions.

Part 2 – Description definition language: specifies the language for defining the standard set of description tools (DSs, Ds, and datatypes) and for defining new description tools.

Part 3 – Visual: specifies the description tools pertaining to visual content.

Part 4 – Audio: specifies the description tools pertaining to audio content.

Part 5 – Multimedia description schemes: specifies the generic description tools pertaining to multimedia including audio and visual content.

Part 6 – Reference software: provides a software implementation of the standard.

Part 7 – Conformance testing: specifies the guidelines and procedures for testing conformance of implementations of the standard.

Part 8 – Extraction and use of MPEG-7 descriptions: provides guidelines and examples of the extraction and use of descriptions.

This document specifies the Description Definition Language (DDL) part of the ISO/IEC 15938 standard. The DDL is the language (and syntax) for constraining the structure and content of MPEG-7 descriptions. This document also provides informative examples demonstrating the use of the DDL to constrain and instantiate MPEG-7 descriptions.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Information technology — Multimedia content description interface —

Part 2: Description definition language

1 Scope

1.1 Scope of this International Standard

This International Standard specifies a metadata system for describing multimedia content. It specifies the Description Definition Language (DDL) that comprises part 2 of the standard (ISO/IEC 15938-2).

The goal of this part of the MPEG-7 International Standard is to specify a language that will enable MPEG-7 users and developers to:

- create valid MPEG-7 description schemes and descriptors;
- develop tools such as editors and parsers for processing descriptions, description schemes and descriptors;
- generate refinements, extensions and modifications to the DDL.

This International Standard describes the features of the DDL. It defines the syntax of the DDL constructs and datatypes and provides optional (informative) examples that illustrate the application of the DDL to the specification and instantiation of MPEG-7 descriptions.

1.2 Overview of Description Definition Language

This International Standard, known as the "Multimedia Content Description Interface", aims at providing standardized core technologies allowing the description of audiovisual data content in multimedia environments. This is a challenging task given the broad spectrum of requirements and targeted multimedia applications, and the broad number of audiovisual features of importance in such a context. In order to achieve this broad goal, the standard specifies:

- Descriptors (D): representations of Features, that define the syntax and the semantics of each feature representation;
- Description Schemes (DS), that specify the structure and semantics of the relationships between their components, which may be both Ds and DSs;
- A Description Definition Language (DDL), to allow the creation of new DSs and, possibly, Ds and to allow the extension and modification of existing DSs;
- System tools, to support multiplexing of description, synchronization issues, transmission mechanisms, file format, etc.

The DDL forms a core part of the MPEG-7 standard. It provides the solid descriptive foundation through which users can create their own Description Schemes and Descriptors. The DDL defines the syntactic rules to express and combine Description Schemes and Descriptors. According to the definition in the MPEG-7 Requirements Document [1] the DDL is

> '...a language that allows the creation of new Description Schemes and, possibly, Descriptors. It also allows the extension and modification of existing Description Schemes.'

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The DDL is not a modelling language such as Unified Modelling Language (UML) but a schema language to represent the results of modelling audiovisual data, i.e., DSs and Ds.

The DDL must satisfy the MPEG-7 DDL requirements. It has to be able to express spatial, temporal, structural, and conceptual relationships between the elements of a DS, and between DSs. It must provide a rich model for links and references between one or more descriptions and the data that it describes. In addition, it must be platform and application independent and human- and machine-readable.

(Non-normative) DDL Parser applications will be required which are capable of validating description schemes (content and structure) and descriptor data types [both primitive (integer, text, date, time) and composite (histograms, enumerated types)], against the DDL. The DDL Parsers must also be capable of validating MPEG-7 descriptions or instantiations, against their corresponding validated MPEG-7 schemas (DSs and Ds).

The DDL design has been informed by numerous proposals and input documents submitted to MPEG-7 since the MPEG-7 Call for Proposals in October 1998 [2]. It has also been heavily influenced by W3C's XML Schema Language [3,4,5] and the Resource Description Framework (RDF) [6].

At the 51st MPEG meeting in Noordwijkerhout in March 2000, it was decided to adopt XML Schema Language [3,4, 5] as the MPEG-7 DDL. However because XML Schema language was not designed specifically for audiovisual content, certain extensions have been necessary in order to satisfy all of the MPEG-7 DDL requirements.

Hence the W3C's XML Schema Language Recommendations [3,4,5] together with the MPEG-7 extensions described in this document, constitute the normative reference for the MPEG-7 DDL. Overviews of the XML Schema Language structural components and datatypes are given in Annexes A and B for informative purposes only. If Annexes A and B contradict the normative references [3,4,5] then the normative references should be used as the definitive specification.

1.3 Structure of this document

The DDL can be broken down into the following logical normative components:

- The XML Schema structural language components;
- The XML Schema datatype language components;
- The MPEG-7 specific extensions.

The complete normative specifications for XML Schema structural and datatype components can be found in [4] and [5] respectively. This International Standard provides the normative specification of the MPEG-7-specific extensions as well as non-normative examples which provide the MPEG-7 community with a clear understanding of the DDL and assist with the development of valid description schemes and descriptors.

The structure of the remainder of this International Standard is as follows:

- Clause 5 describes the normative MPEG-7–specific extensions.
- Clause 6 describes those features of XML Schema which are currently not used by MPEG-7.
- Annex A provides an informative overview of the structural components of XML Schema Language.
- Annex B provides an informative overview of the datatyping mechanisms within XML Schema Language.
- Annex C provides the normative XML Schema which defines the MPEG-7 extensions.
- Annex D contains the Patent Statements.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 15938. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 15938 are encouraged to

--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau maintains a list of currently valid ITU-T Recommendations.

- ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*
- ISO 639 (all parts), *Codes for the representation of names of languages*
- ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*
- ISO 3166-2, *Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code*

NOTE The current list of valid ISO 3166-1 country and ISO 3166-2 region codes is maintained by the ISO 3166 Maintenance agency at the ISO Central Secretariat. Information on the current list of valid region and country codes can be found at http://www.iso.org/mara/iso3166 [11]

- ISO 4217, *Codes for the representation of currencies and funds*

NOTE The current list of valid ISO 4217 currency code is maintained by the official maintenance authority British Standards Institution (http://www.bsi-global.com/iso4217currency) [12]

- XML, *Extensible Markup Language (XML) 1.0,* 6 October 2000 <http://www.w3.org/TR/2000/REC-xml-20001006>
- *Namespaces in XML*, W3C Recommendation, 14 January 1999 <http://www.w3.org/TR/REC-xml-names/>
- *XML Schema, W3C Recommendation*, 2 May 2001 <http://www.w3.org/XML/Schema>
- *XML Schema Part 0: Primer,* W3C Recommendation, 2 May 2001 <http://www.w3.org/TR/xmlschema-0/>
- *XML Schema Part 1: Structures*, W3C Recommendation, 2 May 2001 <http://www.w3.org/TR/xmlschema-1/>
- *XML Schema Part 2: Datatypes*, W3C Recommendation, 2 May 2001 <http://www.w3.org/TR/xmlschema-2/>
- XPath, *XML Path Language, W3C Recommendation*, 16 November 1999 <http://www.w3.org/TR/1999/REC-xpath-19991116>

NOTE These documents are maintained by the W3C (http://www.w3.org/).

- RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*
- RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*
- RFC 2048, *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*
- RFC2045-CHARSETS, *Registered Character set codes of RFC2045*
- RFC2046-MIMETYPES, *Registered Mimetypes of RFC2046*

NOTE The relevant lists can be obtained as follows:

  - MIMETYPES. The current list of registered mimetypes, as defined in RFC2046, RFC2048, is maintained by IANA (Internet Assigned Numbers Authority). It is available from ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types/ [13].
  - CHARSETS. The current list of registered character set codes, as defined in RFC2045 and RFC2048 is maintained by IANA (Internet Assigned Numbers Authority). It is available from http://www.iana.org/assignments/character-sets [14].

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3 Terms and definitions

3.1 Conventions

3.1.1 Naming conventions

The Description Definition Language (DDL) provides constructs such as "element", "attribute", "simpleType" and "complexType" which allow the specification of audio, visual and multimedia descriptors, description schemes and description tools (ISO/IEC 15938-3, ISO/IEC 15938-4, ISO/IEC 15938-5). The names associated with these constructs are created on the basis of the following conventions:

- If the name is composed of multiple words, the first letter of each word is capitalized, with the exception that the capitalization of the first word depends on the type of construct as follows:
- Element naming: the first letter of the first word is capitalized (e.g. *TimePoint* element of *TimeType*).
- Attribute naming: the first letter of the first word is not capitalized (e.g. *timeUnit* attribute of *IncrDurationType*).
- complexType naming: the first letter of the first word is capitalized, and the suffix "Type" is used at the end of the name (e.g. *PersonType*).
- simpleType naming: the first letter of the first word is not capitalized, the suffix "Type" may be used at the end of the name (e.g. *timePointType*).

Note that when referencing a complexType or simpleType in the definition of a description tool, the "Type" suffix is not used. For instance, the text refers to the "*Time datatype*" (instead of "*TimeType datatype*"), to the "*MediaLocator D*" (instead of "*MediaLocatorType D*") and to the "*Person DS*" (instead of "*PersonType DS*").

3.1.2 Documentation convention

Examples illustrating use of the constructs provided by the DDL are presented in this document using a specific font and background as shown in the following example:

```
<complexType name="ExampleType">
  <sequence>
     <element name="Element1" type="string"/>
  </sequence>
  <attribute name="attribute1" type="string" default="attrvalue1"/>
</complexType>
```

Corresponding instantiations of DDL construct examples are presented in this document using a separate font and background as shown in the following example:

```
<Example attribute1="example attribute value">
  <Element1>example element content</Element1>
</Example>
```

Moreover, the schema examples defined in this document follow a type-centric approach. The example description tools are specified by defining the complexType or simpleType for each tool, but not declaring an element of this type at the global scope with the exception of the root element. However, in order to illustrate informative example descriptions in this document, the examples assume that an element of the example type (complexType or simpleType) has been declared, such as a member of another complexType or simpleType. For example, the informative description using the *ExampleType* tool above assumes that the following declaration has been made:

```
<element name="Example" type="mpeg7:ExampleType">
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.2 Wrapper of the schema

The description examples and syntax of description tools specified in this document assume that a schema wrapper is provided which identifies the XML Schema namespace (XML Schema) and MPEG-7 namespace:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  targetNamespace="urn:mpeg:mpeg7:schema:2001"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
<!-- Multimedia Description Schemes ISO/IEC 15938-5 -->
```

The following tag is used to close the schema:

```
</schema>
```

3.3 Terminology

For the purposes of this part of ISO/IEC 15938, the following terms and definitions apply.

3.3.1 Attribute
A property of simple type which is associated with a complex type or an element. Attribute instantiations appear inside the header tag of the element to which it is associated.

3.3.2 Base type
A **type** that serves as the root **type** of a derivation hierarchy for other **types** which are extensions or restrictions of it.

3.3.3 Complex type
A **type** definition which is a set of attribute declarations and a content type applicable to the attributes and children of an element instance respectively.

3.3.4 Conformance
The checking process carried out by a processor in which an instance XML document is checked to ensure that it conforms to the rules specified in a particular schema.

3.3.5 Constraint
A restriction on the structure, value or occurrence of an element or attribute.

3.3.6 Content Model
The structure of the content within an element or complex type. It may be empty (attributes only), simple (text only) or complex. Complex content models may be mixed (child elements and text) or only contain child elements.

3.3.7 Datatype
A primitive reusable **type** employed by **Description Schemes** and **Descriptors**. A ***datatype*** is used to constrain the possible values of an attribute, text-only element or simple type. It consists of a) a set of distinct values, called its value space, b) a set of lexical representations called its lexical space, and c) a set of facets that characterize properties of the value space, individual values or lexical items.

3.3.8 Definition
A ***definition*** creates a new simple or complex type which can be used within declarations or other definitions.

3.3.9 Declaration
A ***declaration*** enables the appearance in a document instance of an element or attribute with a specific name and type.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.3.10 Derived type
A **type** that is defined in terms of extension or restriction of other **types**.

3.3.11 Description
An instantiation of one or more **description tools (description schemes, descriptors or datatypes)**.

3.3.12 Description Scheme
A **description tool** that describes entities or relationships pertaining to multimedia content. **Description Schemes** specify the structure and semantics of their components, which may be **Description Schemes**, **Descriptors**, or **datatypes**.

3.3.13 Description Tool
A **Description Scheme**, **Descriptor**, or **datatype**.

3.3.14 Descriptor
A representation of a feature which defines the syntax or semantics of a feature's representation.

3.3.15 Element
An ***element*** declaration is an association of a name with a simple or complex type definition, an optional (default) value and a set of identity-constraint definitions.

3.3.16 Extension
A complex type definition which allows element or attribute content in addition to that allowed by another specified type definition.

3.3.17 Facet
A constraint which can be applied to a datatype to constrain its set of possible values.

3.3.18 Instance
An ***instance*** or ***instantiation*** is XML data which conforms to the constraints specified in a particular XML schema or DTD.

3.3.19 Namespace
A collection of names, identified by a URI reference [11] which are used in XML documents as element types and attribute names.

3.3.20 Restriction
A type definition whose declarations or facets are in a one-to-one relation with those of another specified type definition, but which restricts the possibilities of the type it corresponds to.

3.3.21 Schema
A set of Schema Components which define a class of XML documents by expressing syntactic, structural and value constraints applicable to document instances.

3.3.22 Schema Component
The generic term for the building blocks that comprise the abstract data model.

3.3.23 Scope
The extent of availability of an element or attribute. Scope can be global or local. Elements and attributes which are declared under *schema* and not as part of a complex type definition, have global scope.

3.3.24 Simple Type
A ***simpleType*** definition is a set of constraints on string values or lists of strings. Simple types cannot have element content and cannot carry attributes. Attributes or text-only elements with no children can be declared to have simple types.

3.3.25 Type
A ***type*** definition is either a simple or complex set of constraints which can be applied to attribute or element instances.

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.3.26 Type Reference
A reference to a predefined simple or complex type.

3.3.27 Validation
The process of parsing an XML document to determine whether it satisfies the constraints embodied in the Schema to which it should conform.

3.3.28 Well-formedness
The term applies to an XML document which meets the set of well-formedness constraints, as outlined in [7], including the constraint that all start tags must have a matching end tag.

4 Symbols and abbreviated terms

For the purposes of this part of ISO/IEC 15938, the symbols and abbreviated terms given in the following apply.

AV: Audio-visual

CIF: Common Intermediate Format

CS: Classification Scheme

D: Descriptor

Ds: Descriptors

DCT: Discrete Cosine Transform

DDL: Description Definition Language

DS: Description Scheme

DSs: Description Schemes

DTD: Document Type Definition

IANA: Internet Assigned Numbers Authority

IETF: Internet Engineering Task Force

IPMP: Intellectual Property Management and Protection

ISO: International Organization for Standardization

JPEG: Joint Photographic Experts Group

MDS: Multimedia Description Scheme

MPEG: Moving Picture Experts Group

MPEG-2: Generic coding of moving pictures and associated audio information [ISO/IEC 13818]

MPEG-4: Coding of audio-visual objects [ISO/IEC 14496]

MPEG-7: Multimedia Content Description Interface Standard [ISO/IEC 15938]

MP3: MPEG-2 layer 3 audio coding

QCIF: Quarter Common Intermediate Format

RDF: Resource Description Framework

SMPTE: Society of Motion Picture and Television Engineers

TZ: Time Zone

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

TZD: Time Zone Difference

URI: Uniform Resource Identifier [RFC 2396]

URL: Uniform Resource Locator [RFC 2396]

W3C: World Wide Web Consortium

XML: Extensible Markup Language

XLink: XML Linking Language

XPath: XML Path Language

XPointer: XML Pointer Language

XSL: XML Stylesheet Language

XSLT: XSL Transformations

5 MPEG-7-specific Extensions

The following features have been added to the XML Schema Language specification in order to satisfy specific MPEG-7 requirements:

- Array and Matrix datatypes;
- Built-in datatypes for basicTimePoint and basicDuration.

5.1 MPEG-7 Structural Extensions

5.1.1 Defining Arrays and Matrices

Mechanisms are required which enable the parser to check that the size of arrays and matrices are conformant to:

- either the pre-defined value of a facet in a schema definition or
- to an attribute at the time of instantiation.

Using the *list* datatype, two methods are provided for specifying sizes of (1D) arrays and multi-dimensional matrices. A new facet *mpeg7:dimension,* is provided to enable the specification of the dimensions of arrays or matrices at the time of schema definition. A new attribute *mpeg7:dim*, is provided to enable the size of the array or matrix to be specified in the instance.

When *mpeg7:dimension* is used with positive integers, the size is fixed and the parser will generate the array or matrix from a list of values, accordingly. The *mpeg7:dimension* facet can also take a value "unbounded". In such a case, a derived simpleType can redefine these unbounded dimensions with a fixed value. Furthermore, an instantiated matrix, whose definition contains unbounded dimensions, should be used in conjunction with the *mpeg7:dim* attributes. It removes any ambiguity that occurs when using unbounded dimensions.

For 1D arrays, the *mpeg7:dimension* facet provides the same functionality as the existing *length* facet. If neither the *mpeg7:dimension* nor the *length* facet are specified, the default is a list of arbitrary length. The length will be determined by the number of elements in the list. Preferably only one facet (*length* or *mpeg7:dimension*) is specified. If both are specified then they must be consistent otherwise the parser will flag an error.

In the following example, a 2D matrix of integers with 3 rows and 4 columns is defined. The dimension value for a 3D matrix will define the number of rows, columns and layers.

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example:

```
<simpleType name="IntegerMatrix2D">
   <list itemType="integer">
      <annotation><appinfo>
         <mpeg7:dimension value="unbounded unbounded"/>
      </appinfo></annotation>
   </list>
</simpleType>

<simpleType name="IntegerMatrix3x4">
   <restriction base="mpeg7:IntegerMatrix2D">
      <annotation><appinfo>
         <mpeg7:dimension value="3 4"/>
       </appinfo></annotation>
    </restriction>
</simpleType>

<element name="IntegerMatrix3x4" type="mpeg7:IntegerMatrix3x4"/>
```

Non-normative example of a valid instance:

```
<IntegerMatrix3x4>
   8  9  4  5
   6  7  8  2
   2  1  3  5
</IntegerMatrix3x4>
```

In case of unbounded dimensions, the value of *mpeg7:dimension* should be specified within the description using the following reserved *mpeg7:dim* attribute.

To support parameterized array sizes, one can use the special attribute, *mpeg7:dim*, to specify the dimension of a (complex) list type at the time of instantiation. The *mpeg7:dim* attribute must itself be defined in the mpeg7 namespace as a list of positive integers. It is defined in a separate schema which contains the MPEG-7 extensions (Annex C). It is a reserved attribute and it must not be used for any other purpose.

Non-normative example:

```
<simpleType name= "listOfPositiveIntegerForDim">
      <list itemType="positiveInteger"/>
</simpleType>

<attribute name="dim">
   <simpleType>
      <restriction base="listOfPositiveIntegerForDim">
           <minLength value="1"/>
      </restriction>
   </simpleType>
</attribute>

<complexType name="NDimIntegerArray">
   <simpleContent>
      <extension base="listOfInteger">
      <attribute ref="mpeg7:dim" />
      </extension>
   </simpleContent>
</complexType>

<element name="IntegerMatrix" type="NDimIntegerArray" />
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In the following example, a 2D matrix with 2 rows and 4 columns is specified at the time of instantiation using *mpeg7:dim*. Similarly for a 3D matrix, *mpeg7:dim* will specify the number or rows, columns and layers and the correspondence between the list of values and the matrix is in this order (row major).

Non-normative example of a valid instance:

```
<IntegerMatrix mpeg7:dim="2 4">
  1 2 3 4
  5 6 7 8
</IntegerMatrix>
```

Below are a set of non-normative examples which illustrate how arrays and matrices can be defined using the XML Schema *list* datatype. Note that you cannot create new list types by derivation from existing list types nor from complex types:

```
<simpleType name="listOfInteger">
     <list itemType="integer"/>
</simpleType>

<!-- Definition of "Vector of integers"    -->
<complexType name="VectorI">
   <simpleContent>
      <extension base="listOfInteger">
         <attribute ref="mpeg7:dim" />
      </extension>
   </simpleContent>
</complexType>

<simpleType name="listOfFloat">
     <list itemType="float"/>
</simpleType>

<!-- Definition of "Vector of reals"    -->
<complexType name="VectorR">
   <simpleContent>
      <extension base="listOfFloat">
         <attribute ref="mpeg7:dim" />
      </extension>
   </simpleContent>
</complexType>

<!-- Definition of "Matrix of integers"         -->
<complexType name="MatrixI">
    <simpleContent>
      <extension base="listOfInteger">
         <attribute ref="mpeg7:dim" />
      </extension>
    </simpleContent>
</complexType>

<!-- Definition of "Matrix of reals"         -->
<complexType name="MatrixR">
   <simpleContent>
      <extension base="listOfFloat">
         <attribute ref="mpeg7:dim" />
      </extension>
   </simpleContent>
</complexType>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.2 MPEG-7 Datatype Extensions

The following built-in primitive time data types have been provided within the DDL to satisfy the specific requirements of MPEG-7 implementers:

- **basicTimePoint**,
- **basicDuration.**

These are specified in detail in the XML Schema containing the MPEG-7 extensions, which is shown in Annex C.

5.2.1 BasicTimePoint

The basicTimePoint datatype specifies a time point according to the Gregorian dates, day time and the Time Zone (TZ). The format is based on ISO 8601. To reduce conversion problems only a subset of the ISO 8601 formats is used.

5.2.1.1 basicTimePoint datatype syntax

```
<!-- ################################################ -->
<!--   Definition the basicTimePoint Datatype              -->
<!-- ################################################ -->

<simpleType name="basicTimePointType">
  <restriction base="string">
     <pattern value="(\-?\d+(\-\d{2}(\-\d{2})?)?)?(T\d{2}(:\d{2}(:\d{2}
                  (:\d+(\.\d{2})?)?)?)?)?(F\d+)?((\-|\+)\d{2}:\d{2})?"/>
  </restriction>
</simpleType>
```

5.2.1.2 basicTimePoint datatype semantics

Name	*Definition*
basicTimePointType	A simpleType representing an instance in time using a lexical representation of Gregorian date (YYYY-MM-DD), day time (hh:mm:ss), a fraction specification (nnn.ff) including the specification of the number of fractions of one second (NNN) and the time zone according to UTC (±hh:mm). The lexical representation is given by YYYY-MM-DDThh:mm:ss:nnn.ffFNNN±hh:mm The following lexicals are used for digits of the corresponding date/time elements: • Y: Year, can be a variable number of digits, • M: Month, • D: Day, • h: hour, • m: minute, • s: second, • n/f: decimal numbers of fractions in the format *nnn.ff*, *nnn* can be any number between 0 and *NNN-1* (*NNN* and *nnn* can have an arbitrary number of digits). *ff* are the decimal fractions of a *1/NNN* second and are limited to two digits.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Name	*Definition*
	• N: number of fractions of one second which are counted by *nnn*. Like *nnn*, *NNN* can have an arbitrary number of digits and is not limited to three. • ±hh:mm: the time zone according to the UTC specified by a positive or negative (±) number of hours (h) with two digits and number of minutes (m) also with two digits. If no time zone is specified, a time zone of 00:00 is assumed. Also delimiters for the time specification (T) and the number of fractions of one second are used (F). While the date representation is a subset of the ISO8601 formats, the Time representation is specified according to the SMPTE timestamp formats. The basicTimePointType is derived from the string datatype by restriction.

Beside the specification of Gregorian Date, Day time counting the number (nnn) of fractions (FNNN) of a second is possible to allow a higher precision than one second. If counting fractions is used, the number of fractions making up one second (FNNN) has to be specified along with the counted number of fraction (nnn). Thus the FNNN defines the value range of the counted number of fractions (nnn) value. So the value range of 'nnn' is limited from 0 to FNNN-1. If no time zone is specified a time zone of 00:00 is assumed.

To support the current practice in AV material, the time expressions specify a fraction of a second by counting predefined fractions instead of a decimal expression used within the date and time datatypes of XML Schema. Nevertheless XML Schema-based time specifications can be represented as follows:

For instance a timeInstant of T13:20:01.235 would be expressed using the basicTimePoint datatype by T13:20:01:235F1000. According to the number of used decimal digits the number of fractions of one second are 1000 as specified in the basicTimePoint datatype.

According to this specification of the basicTimePoint, the following rules apply, in addition to the restriction by the pattern facet, when instantiating an element or attribute of this datatype:

- the two digits MM representing the “month” specification must have a value range from 1 to 12.
- the two digits DD representing the “day” specification must have a value range from 1 to 28 if the month value equals 2, 1 to 29 if the month value equals 2 and the year is a leap year, 1 to 30 if the month value equals 4, 6, 9 or 11, and 1 to 31 if the month value equals 1, 3, 5, 7, 8, 10 or 12.
- the two digits hh which follow the time delimiter “T” representing the “hour” specification must have a value range from 0 to 23.
- the two digits mm which follow the time delimiter “T” representing the “minute” specification must have a value range from 0 to 59.
- the two digits ss representing the “seconds” specification must have a value range from 0 to 60. A value of 60 is only applicable in the case of a leap second.
- nnn is an arbitrary number of digits representing a number of fractions specification which has a value range from 0 to NNN-1. NNN is an arbitrary number of digits representing a number of fractions building up one second. If nnn is specified also NNN has to be specified.

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- the two digits hh which follow the sign "±" representing the "hour" of the time zone specification must have a value range from 0 to 15.

- the two digits mm which follow the time delimiter ":" representing the "minute" of the time zone specification must have a value 0, 15, 30 or 45.

If the basicTimePoint datatype is further restricted by a pattern facet, these additional rules also apply to the "corresponding digits" present in the restricted datatype. "Corresponding digits" are digits in the same position with respect to their position relative to the delimiters "-", "T", ":", ".", "F" and "+" in the expressions of two datatypes.

5.2.2 BasicDuration

The basicDuration datatype specifies the duration of a time period according to days and time of day. The format is based on the ISO 8601 norm. To reduce conversion problems only a subset of the ISO 8601 formats is used. Fractions of a second are specified according to the basicTimePoint datatype.

5.2.2.1 basicDuration datatype syntax

```
<!-- ################################################ -->
<!--   Definition the basicDuration Datatype                -->
<!-- ################################################ -->

<simpleType name="basicDurationType">
  <restriction base="string" >
     <pattern value="\-?P(\d+D)?(T(\d+H)?(\d+M)?(\d+S)?(\d+N)?
                  (\d{2}f)?)?(\d+F)?((\-|\+)\d{2}:\d{2}Z)?"/>
  </restriction>
</simpleType>
```

5.2.2.2 basicDuration datatype semantics

Name	*Definition*
basicDurationType	A simpleType representing a duration in time using a lexical representation of days (nD), time duration and a fraction specification (TnHnMnSnNnf) including the specification of the number of fractions of one second (nF) and the difference in the time zone according to UTC (±hh:mm): (-)PnDTnHnMnSnNnfnF±hh:mmZ While the day representation is a subset of the ISO8601 formats the Time duration representation is specified according to the SMPTE timestamp formats. The basicDurationType is derived from the string datatype by restriction.

In the used lexical expression n specifies a non-negative number. The numbers are followed by a separator specifying the semantic of the number: D (days), H (hours), M (minutes), S (seconds), N (number of fractions), f (for a decimal expression of fractions), F (number of fractions of one second). Also a delimiter for the time specification (T) is used.The scope of basicDuration datatype is limited to days to avoid ambiguities depending on the date of the start time point if for instances describing the duration of a segment with "one month".

The duration for instance of a time segment can be specified using a number of days and time. The specification of the number of fractions (nN) is the same as in the case of basicTimePoint. Also in the basicDuration the fractions of one second (nF) has to be specified if the number of fractions (nN) is specified. The scope of basicDuration datatype is limited to days to avoid ambiguities depending on the date of the start time point if for instances

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

describing the duration of a segment with "one month". Note also that the basicDuration datatype can specify a difference in the TZ specification that indicates the difference in the time zone from the starting point to the end point (for instance using a mobile camera). In this case the value of the basicDuration does not differ whether this shift in the time zone is specified or not. The basicDuration always describes the time duration of a time period. The difference in the TZ is only used to map the resulting end time point of the period to the local time.

According to this specification of the basicDuration, the following rules apply in addition to the restriction by the pattern facet when validating an instantiation of an element or attribute of this datatype:

- the two digits hh representing the “hour” of the time zone difference specification must have a value range from 0 to 26.
- the two digits mm representing the “minute” of the time zone difference specification must have a value 0,15,30 or 45.

If the basicDuration datatype is further restricted by a pattern facet, these additional rules also apply to the “corresponding digits” present in the restricted datatype. “Corresponding digits” are digits in the same position with respect to their position relative to the delimiter “Z” in the expressions of two datatypes.

6 Unused XML Schema Features

The following feature of XML Schema language is not currently necessary to satisfy the requirements of the MPEG-7 DDL. However that is not to say that it may not prove useful in the future as MPEG-7 implementations grow in size and complexity:

- Notation declarations.

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(informative)

Overview of XML Schema Structural Components

A.1 Introduction

This Annex describes the language constructs provided by XML Schema which are used to define the structures and attributes of MPEG-7 Description Schemes and Descriptors.

According to the XML Schema specification, an XML Schema consists of a set of schema components which can be divided into three groups. The primary components are:

- The Schema – the wrapper around the definitions and declarations (Section A.2.1);
- Simple type definitions (Section A.2.2.1);
- Complex type definitions (Section A.2.2.2);
- Attribute declarations (Section A.2.3.6);
- Element declarations (Section A.2.3.1).

The secondary components are:

- Attribute group definitions (Section A.2.4.1);
- Identity-constraint definitions (Section A.2.9);
- Named group definitions (Section A.2.4.3);
- Notation declarations (Not used in MPEG-7 DDL).

The third group are the “helper” components which contribute to the other components and cannot stand alone:

- Substitution groups (Section A.2.3.4);
- Wildcards (Section A.2.5);
- Annotations (Section A.2.6).

A.2 Detailed Description

A.2.1 Schema Wrapper – Preamble

The preamble consists of an XML element “schema” which contains the following attributes:

- A URI reference to the XML Schema namespace: xmnls="http://www.w3.org/2001/XMLSchema"
- *targetNamespace*: the current schema identity i.e. the URI reference by which the current schema is to be identified;
- *version*: a schema version specification;
- a reference (URI) to the MPEG7 DDL to be used for validation:

  xmlns:mpeg7=urn:mpeg:mpeg7:schema:2001
- *xmlns*: URI references to other imported schemas and abbreviations for referring to definitions in these external schemas. The xmlns keyword can be followed by ‘:’ and a prefix. This identifies the origins of the referenced constructs;
- *elementFormDefault:* indicates whether locally declared elements must be qualified or unqualified;
- *attributeFormDefault*: indicates whether locally declared attributes must be qualified or unqualified.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example of an XML schema preamble:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
      targetNamespace="urn:mpeg:mpeg7:schema:2001"
      elementFormDefault="qualified"
      attributeFormDefault="unqualified">
……
……
</schema>
```

XML namespaces [8] provide a simple method for qualifying names of descriptors and description schemes used in MPEG-7 DDL descriptions by associating them with namespaces identified by URI references [9]. Every schema definition must begin with this preamble in order to identify the namespace of itself. This also provides the ability to generate descriptions which combine schemas from multiple different namespaces.

A.2.2 Type Definitions

Type definitions define internal schema components which can be used in other schema components such as element or attribute declarations or other type definitions. XML Schema provides two kinds of type definition component:

- simple types – which are simple data types (built-in or derived) which cannot have any child elements or attributes;
- complex types – which may carry attributes and have children elements or be derived from other simple or complex types.

A.2.2.1 Simple Type Definitions

New simple types can be defined by derivation from existing simple types (built-ins or derived) through a restriction of the base simple type. Elements and attributes can then be declared which have these simple types. Simple types cannot have element content and cannot carry attributes.

The available built-in simple types and their corresponding facets are in Appendix B of [4]. The examples below illustrate how to define new simple types through the application of restrictions using facets on the base simple types.

Non-normative examples of simpleType definitions:

```
<simpleType name="unsigned6">
   <restriction base="nonNegativeInteger">
      <minInclusive value="0"/>
      <maxInclusive value="63"/>
   </restriction>
</simpleType>
```

```
<simpleType name="directionType">
  <restriction base="string">
      <enumeration value="uniDirectional"/>
      <enumeration value="biDirectional"/>
  </restriction>
</simpleType>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In addition to atomic (indivisible) simple types, XML Schema provides two aggregate simple types: List types and Union types.

List types are composed of sequences of atomic types, separated by whitespace which acts as the delimiter. XML Schema has three built-in list types, NMTOKENS, IDREFS and ENTITIES. In addition, you can create new list types by derivation from existing atomic types. You cannot create list types from existing list types or from complex types but you can apply facets (length, minLength, maxLength and enumeration) to derive new list types.

Non-normative example of list type definitions:

```
<simpleType name="listOfString">
  <list itemType="string"/>
</simpleType>

<simpleType name="listOf6Strings">
  <restriction base="listOfString">
     <length value="6"/>
  </restriction>
</simpleType>
```

Union types enable element or attribute values to be one or more instances of one type drawn from the union of multiple atomic and list types. In the example below, a new type "Unsigned6OrDirection" is defined which is the union of two existing types: "unsigned6" and "directionType".

Non-normative example of a union type definition:

```
<element name="Unsigned6OrDirection">
  <simpleType>
     <union memberTypes="unsigned6 directionType"/>
  </simpleType>
</element>
```

Non-normative example of a valid instance:

```
<Unsigned6OrDirection>45</Unsigned6OrDirection>
<Unsigned6OrDirection>uniDirectional</Unsigned6OrDirection>
```

A.2.2.2 Complex Type Definitions

A complex type definition is a set of attribute declarations and a content type, applicable to the attributes and children of an element declared to be of this complex type.

Complex type definitions provide the following:

- Constraints on the appearance and content of attributes;
- Constraints on children elements - to be empty (simpleContent), or to conform to a element-only (complexContent) or mixed content model (mixed="true");
- Derivation of complex types from other simple or complex types through extension or restriction.

New complex types are defined using the complexType element and such definitions typically contain a set of element declarations, element references, and attribute declarations. The declarations are not themselves types, but rather an association between a name and constraints which govern the appearance of that name in documents governed by the associated schema. Elements are declared using the *element* element, and attributes are declared using the *attribute* element. In the example below, Organization is defined as a complex type, and within the definition of Organization we see three element declarations and one attribute declaration.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example of a complexType definition:

```
<complexType name="OrganizationType">
  <sequence>
     <element name="OrganizationName" type="string"/>
     <element name="ContactPerson" type="mpeg7:IndividualType" minOccurs="0"
            maxOccurs="unbounded"/>
     <element name="Address" type="mpeg7:PlaceType" minOccurs="0"/>
  </sequence>
  <attribute name="id" type="ID" use="required"/>
</complexType>
```

The consequence of this definition is that any element appearing in an instance whose type is declared to be *Organization* must consist of three elements and one attribute. These elements must be called *OrganizationName*, *ContactPerson* and *Address*. The first of these elements will contain a string, the second will contain the complexType *Individual* and the third will contain the complexType *Place*. Any element whose type is declared to be *Organization* must appear with an attribute called *id* which must contain an *ID*.

A.2.2.3 Element Content Constraints

It is possible to constrain the content model of a complexType to the following:

- empty – no child elements only attributes;
- mixed – character data appears between elements and their children. The *mixed* attribute must be set to "true";
- complexContent – the default content type which consists of elements and attributes;
- simpleContent – use when deriving a complexType from a simpleType. This indicates that the new type contains only character data and no elements.

Non-normative example of the *empty* content model:

```
<element name="InternationalPrice">
   <complexType>
      <attribute name="currency" type="mpeg7:ISO4217CurrencyCode"/>
      <attribute name="value" type="decimal"/>
   </complexType>
</element>
```

Non-normative example of a valid instance:

```
<InternationalPrice currency="EU" value="423.46"/>
```

Non-normative example of the *mixed* content model:

```
<element name="Salutation">
    <complexType mixed="true">
       <sequence>
          <element name="Name" type="string"/>
       </sequence>
    </complexType>
</element>
```

Non-normative example of a valid instance:

```
<Salutation>Dear Mr.<Name>Robert Smith</Name>,</Salutation>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In addition, XML Schema provides an *anyType* type which does not constrain element content in any way. This is the base type from which all simple and complex types are derived. Elements declared using this type can contain elements, character data or a mixture of characters and elements. This is the default type if none is specified.

```
<element name="anything" type="anyType"/>
```

A.2.2.4 Derived Complex Types

It is possible to derive new complex Types by:

- Extension of a simple base type definition (*simpleContent* content model);
- Restriction of a simple base type definition (*simpleContent* content model);
- Extension of a complex base type definition (*complexContent* content model);
- Restriction of a complex base type definition (*complexContent* content model).

A complex type which extends another does so by having additional content model particles at the end of the other definition's content model, or by having additional attribute declarations, or both.

Below is an example of the derivation of a complexType by extension of a simpleType. The simpleType is extended through the addition of attributes. To indicate that the content model of the new type contains only character data and no elements, the *simpleContent* element is used.

Non-normative example of derivation of a complexType by extension of a simpleType:

```
<complexType name="ControlledTermType">
  <simpleContent>
     <extension base="string">
        <attribute name="CSName" type="string" use="optional"/>
        <attribute name="CSTermId" type="string" use="optional"/>
        <attribute name="CSLocation" type="uriReference" use="optional"/>
     </extension>
  </simpleContent>
</complexType>
```

Below is an example of a new complex type which is derived through the extension of an existing complex type. The *Person* type is extended through the addition of a new *role* element, to create the *Creator* type. The *complexContent* element is required to indicate that we intend to restrict or extend the content model of a complex type and that the new type contains only elements and no character data.

Non-normative example of derivation of a new complexType by extension of an existing complexType:

```
<complexType name="Creator">
  <complexContent>
     <extension base="mpeg7:PersonType">
        <sequence>
           <element name="role" type="mpeg7:ControlledTermType"/>
        </sequence>
     </extension>
  </complexContent>
</complexType>
```

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A type definition whose declarations or facets are in a one-to-one relation with those of another specified type definition, with each in turn restricting the possibilities of the one it corresponds to, is said to be a restriction. The specific restrictions might include narrowed ranges or reduced alternatives. Members of a type, A, whose definition is a restriction of the definition of another type, B, are always members of type B as well.

In the example below the *simpleName* type is derived through restriction of the *personName* type. The restriction is on the occurrence constraint of the *forename* element which is reduced from unbounded to 1.

Non-normative example of derivation by restriction:

```
<complexType name="PersonNameType">
  <sequence>
     <element name="Title" minOccurs="0"/>
     <element name="Forename" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="SimpleName">
  <complexContent>
     <restriction base="mpeg7:PersonNameType">
     <sequence>
        <element name="Title" maxOccurs="0"/>
        <element name="Forename" minOccurs="1" maxOccurs="1"/>
     </sequence>
     </restriction>
  </complexContent>
</complexType>
```

A.2.2.5 Restricting Permitted Derivations

XML Schema also provides a number of mechanisms for controlling permitted derivations of particular types. The *final* attribute can be set to *restriction*|*extension*|*#all* to prevent derivations by restriction, extension or any derivations at all, respectively.

Non-normative example of a "final" derivation restriction:

```
<complexType name="PersonNameType" final="restriction">
  <sequence>
     <element name="Title" minOccurs="0"/>
     <element name="Forename" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

Moreover, it is possible to specify an optional *finalDefault* attribute on the *schema* element which is equivalent to specifying a final attribute on every type definition and element declaration in the schema.

It is also possible to control which facets can be applied in the derivation of new simple types through use of the *fixed* attribute. If fixed is true for a particular facet then that facet cannot be changed in any new derived types.

Non-normative example of fixed facets:

```
<simpleType name="unsigned6">
  <restriction base="nonNegativeInteger">
     <minInclusive value="0" fixed="true"/>
     <maxInclusive value="63" fixed="true"/>
  </restriction>
</simpleType>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The *block* attribute can also be applied to a type definition to prevent derivations of particular types from replacing the content model of the base type in instantiations. The block attribute can be set to *restriction*|*extension*|*#all* to prevent replacements by derivations by restriction, derivations by extension or any derivations at all, respectively. There is also an optional *blockDefault* attribute provided for the *schema* element which can be used to specify a *block* attribute on every type definition and element declaration in the schema. *blockDefault* can also be set to *substitution* in order to disallow substitutions.

A.2.3 Declaration Components

A.2.3.1 Element Declarations

An element declaration specifies a type definition for an element either by reference or explicitly, and may provide occurrence (*minOccurs* and *maxOccurs* attributes) and default information (*default* attribute). The default values for minOccurs and maxOccurs are:

minOccurs =1;

maxOccurs =

1. unbounded, if the maxOccurs [attribute] equals unbounded,
2. otherwise the number corresponding to the normalized value of the maxOccurs [attribute], if present,
3. otherwise 1.

Non-normative examples of element declarations:

```
<element name="MyElement" type="string" default="abc"/>

<element name="Et0" type="mpeg7:MyComplexType" minOccurs="0" maxOccurs="unbounded"/>
```

A.2.3.2 Anonymous Type Definitions

Schemas can be constructed by defining named types and then declaring elements that reference the types using the *element name=.. type=..* construction. This style of schema construction is straightforward but it can be unwieldy, especially if you define many types that are referenced only once and contain very few constraints. In these cases, a type can be more succinctly defined as an anonymous type which saves the overhead of having to be named and explicitly referenced.

Non-normative example of an anonymous complexType definition:

```
<element name="Et1">
  <complexType>
     <sequence>
        <element ref="mpeg7:Et0"/>
                . . .
     </sequence>
     <attribute>. . .</attribute>
  </complexType>
</element>
```

A.2.3.3 Element References

The element declarations we have described so far have each associated a name with an existing type definition. Sometimes it is preferable to reference an existing element rather than declare a new element.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example of an element reference:

```
<element name="Comment" type="string" />
<element ref="mpeg7:Comment" minOccurs="0" />
```

This declaration references an existing element, *comment*, that was declared elsewhere in the schema. In general, the value of the *ref* attribute must reference a global element, i.e. one that has been declared under *schema* rather than as part of a complex type definition. The consequence of this declaration is that an element called *comment* may appear in an instance document, and its content must be consistent with that element's type, in this case, *string*.

A.2.3.4 Substitution Groups

Substitution groups allow elements to be substituted for other elements. Using the *substitutionGroup* attribute, elements can be declared to be substitutable for a particular element, called the head element. The head element must be a global element. For example if we declare two elements *seriesTitle* and *programTitle* and assign them to a substitution group whose head element is the *Title* element, then *seriesTitle* and *programTitle* can be used anywhere that *Title* has been used. Elements in a substitution group must be the same type as the head element or they can have a type that has been derived from the head element's type.

Non-normative examples of substitutionGroups:

```
<element name="Title" type="string"/>
<element name="SeriesTitle" type="string" substitutionGroup="mpeg7:Title"/>
<element name="ProgramTitle" type="string" substitutionGroup="mpeg7:Title"/>
```

A.2.3.5 Abstract Elements and Types

Elements and Types may be declared to be *abstract*. Abstract elements cannot appear in instantiations. When an element is declared to be abstract, a member of that element's substitutable class must appear in the instance document.

Non-normative example of an abstract element definition:

```
<element name= "AbstractElement" type="string" abstract="true"/>
```

When an element's corresponding type definition is declared as abstract, all instances of that element must use *xsi:type* to indicate a derived type that is not abstract.

Non-normative example of an abstract type definition:

```
<complexType name="VehicleType" abstract="true"/>

<complexType name="CarType">
  <complexContent>
     <extension base="mpeg7:VehicleType">
  </complexContent>
</complexType>

<complexType name="PlaneType">
  <complexContent>
     <extension base="mpeg7:VehicleType">
  </complexContent>
</complexType>

<element name="Transport" type="mpeg7:VehicleType"/>
```


--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example of a valid instance:

```
<Transport xsi:type="CarType"/>
```

A.2.3.6 Attribute Declarations

Attribute declarations associate a name with constraints on the presence and value of the attribute by referring to a simple datatype.

Attribute definitions enable the following constraints to be associated with an attribute name:

- *type* - constrains the attribute value to this simple datatype;
- *use* - a value which indicates whether the attribute is *required* | *optional* | *prohibited*; by default the value is "optional";
- *fixed* - indicates a fixed value for the attribute;
- *default* – indicates a default value for the attribute.

Attributes may appear once or not at all (the default). A *use* attribute is provided in an attribute declaration to indicate whether the attribute is required, optional or prohibited. If optional, the attribute's value can be set to either a fixed value or a default value using the *fixed* and *default* attributes respectively. To illustrate, the example below contains a declaration for the *Income* attribute, which is declared with *use =optional* and *fixed =42*. This declaration means that the appearance of an *Income* attribute is optional (the default) and if it does appear, its value must be 42, and if it does not appear, a schema processor will create an Income attribute with this value.

Non-normative example of attribute declarations:

```
<attribute name="Income" type="integer" use="optional" fixed="42"/>
<complexType name="MyDS2">
   <sequence>
      <element ref="mpeg7:MyDS1"/>
   </sequence>
   <attribute ref="mpeg7:Income"/>
</complexType>
```

A.2.3.7 Global Elements and Attributes

Global elements and attributes, are created by declarations that appear as the children of the *schema* element. Once declared, a global element or a global attribute can be referenced in one or more declarations using the *ref* attribute as described above. A declaration that references a global element enables the referenced element to appear in the instance document in the context of the referencing declaration. The declaration of a global element also enables the element to appear at the top-level of an instance document.

There are a number of caveats concerning the use of global elements and attributes. One caveat is that global declarations cannot contain references; global declarations must identify simple and complex types directly. Put concretely, global declarations cannot contain the ref attribute, they must use the type attribute (or, as we describe shortly, be followed by an anonymous type definition). A second caveat is that cardinality constraints cannot be placed on global declarations, although they can be placed on local declarations that reference global declarations. In other words, global declarations cannot contain the attributes minOccurs, maxOccurs, or use.

A.2.4 Group Definitions

A.2.4.1 Attribute Group Definitions

Attribute Group definitions provide a mechanism to include a group of attributes by name within a complexType definition.

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example of attributeGroup definitions:

```
<attributeGroup name="idHrefGroup">
    <attribute name="id" type="ID" use="required"/>
    <attribute name="href" type="uriReference" use="required"/>
</attributeGroup>

<complexType name="MyDSType">
    <sequence>
       <element name="Segment" type="mpeg7:SegmentType"/>
    </sequence>
    <attributeGroup ref="mpeg7:idHrefGroup"/>
</complexType>
```

A.2.4.2 Unnamed Groups of Elements

Three compositors are provided to construct unnamed groups of elements:

- sequence – constrains the elements in the group to appear in the same order in which they are declared;
- choice – only one of the elements in the group may appear in an instance;
- all – all of the elements in the group may appear once or not at all and in any order.

Non-normative example of unnamed groups of elements:

```
<complexType name="SegmentNodeType">
  <sequence maxOccurs="unbounded">
     <choice minOccurs="0" maxOccurs="unbounded">
        <element name="ReferenceToSegment"
                 type="mpeg7:ReferenceToSegmentType" />
        <element name="SegmentNode" type="mpeg7:SegmentNodeType"/>
     </choice>
     <element name="SegmentRelationshipNode"
              type="mpeg7:SegmentRelationshipNodeType"
              minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="id" type="ID" use="optional"/>
</complexType>
```

A.2.4.3 Named Groups

XML Schema enables a group of elements to be defined and named so that the elements can be used to build up the content models of complex types. A group definition associates a name with a group of elements. By reference to the name, the entire group can be incorporated by reference into a complexType or element. The group name does not appear in description instantiations. This construct serves a similar purpose to the *attributeGroup* mechanism for attributes but is applied to content models.

Possible compositors within a group definition are: *all* | *choice* | *sequence*.

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Non-normative example of named groups:

```
<group name="MyGrp">
      <choice>
            <element ref="mpeg7:MyDS1"/>
            <element ref="mpeg7:MyDS2"/>
      </choice>
</group>
<complexType name="AnotherDSType">
       <group ref="MyGrp"/>
       <attribute ……/>
</complexType>
```

A.2.5 Nil Values

XML Schema provides a mechanism by which elements can be explicitly given a "null" value, rather than simply being absent. XML Schema's nil mechanism involves an "out of band" nil signal. There is no actual nil value that appears as element content, instead there is an attribute to indicate that the element content is nil. For example, the publicationDate element declaration below indicates that publicationDate can be nil:

```
<xsd:element name="publicationDate" type="date" nillable="true"/>
```

And to explicitly represent that *publicationDate* has a nil value in the instance document, we set the nil attribute (from the XML Schema namespace for instances) to true:

```
<publicationDate xsi:nil="true"></publicationDate>
```

Note that the nil mechanism applies only to element values, and not to attribute values. An element with xsi:nil="true" may not have any element content but it may still carry attributes.

A.2.6 Wildcards

The *any* element can be used to specify that any well-formed XML is permissible in an element type's content model. The *namespace* attribute associated with the *any* element can be used to restrict the namespaces from which the element content can be taken. Possible namespace values are:

- ##any – Any well-formed XML from any namespace;
- ##local – Any well-formed XML that is nor qualified i.e. not declared to be in a namespace;
- ##other – Any well-formed XML in a namespace which is different to the target namespace of the type being defined;
- "http://www.w3.org/1999/xhtml##targetNamespace" – Any well-formed XML belonging to any namespace in the (whitespace separated) list; ##targetNamespace is the target namespace of the type being defined;

A *processContents* attribute can be set of *skip*|*lax*|*strict* depending on whether the content which appears in the *any* element instantiation needs to be validated or not.

Non-normative example of the *any* wildcard:

```
<element name="OpenElement">
      <complexType>
          <sequence>
            <any namespace="##other" processContents="skip"/>
          </sequence>
      </complexType>
</element>
```

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Similarly, the *anyAttribute* element allows any attribute to be associated with elements. A *namespace* attribute (as for the *any* element) can be associated with *anyAttribute* to specify the possible namespaces from which the attribute content can be taken.

Non-normative example of *anyAttribute*:

```
<element name="OpenElement">
  <complexType>
     <sequence>
           <any namespace="##other"/>
     </sequence>
     <anyAttribute namespace="##local"/>
  </complexType>
</element>
```

The above declaration permits the following example of an instantiation:

```
<OpenElement size="10000">
      <comment>This could be anything.</comment>
</OpenElement>
```

A.2.7 Annotations

Schema annotations can be inserted using the *annotation* element and either of its two sub-elements, *documentation* or *appinfo*. Annotations can appear at the beginning of most schema constructs i.e. inside elements and complex type definitions. The *documentation* element can be used to insert comments and the *appInfo* element can be used to provide information for tools, stylesheets and other applications.

Non-normative example of documentation annotation:

```
<complexType name="MatrixR">
  <annotation>
     <documentation>
          Matrix of Reals with Size1 rows and Size2 columns
     </documentation>
  </annotation>
  <simpleContent>
     <extension base="mpeg7:listOfFloat">
        <attribute name="size1" type="nonNegativeInteger" use="required" />
        <attribute name="size2" type="nonNegativeInteger" use="required" />
     </extension>
  </simpleContent>
</complexType>
```

Non-normative example of appinfo annotation:

```
<simpleType name="IntegerMatrix3x4">
  <restriction base="IntegerMatrix2D">
     <annotation><appinfo>
        <mpeg7:dimension value="3 4"/>
     </appinfo></annotation>
    </restriction>
</simpleType>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.8 Including and Importing From External Schema Documents

XML Schema provides two mechanisms which enable schema components to be spread over multiple schema documents and reused or shared amongst schemas: the *include* and *import* mechanisms.

Using the *include* mechanism it is possible to specify another schema file in the same target namespace of the including schema and thus bring in the definitions and declarations contained in the included schema. For example consider the header of the schema file below, Audio.xsd which lies in the MPEG-7 namespace, urn:mpeg:mpeg7:schema:2001. This *includes* the schema declarations and definitions in the schema file *MDS.xsd* which lies in the same namespace but is located at urn:mpeg:mpeg7:schema:2001/MDS.xsd

Non-normative example of schema inclusion:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
      targetNamespace="urn:mpeg:mpeg7:schema:2001">
<include schemaLocation="urn:mpeg:mpeg7:schema:2001/MDS.xsd"/>
```

Using the *import* mechanism, it is possible to bring in definitions and declarations from schemas which lie in multiple different namespaces. In the example below, the schema file *dcmes.xsd* is imported from the Dublin Core namespace.

Non-normative example of schema importation:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
      targetNamespace="urn:mpeg:mpeg7:schema:2001"
      xmlns:dc="http://dublincore.org/documents/2001">
<import namespace="http://dublincore.org/documents/2001"
       schemaLocation="http://dublincore.org/documents/2001/dcmes.xsd"/>
```

A.2.9 Redefining Types and Groups

Using the *redefine* mechanism it is possible to redefine simple and complex types, groups and attribute groups that are obtained from external schema files which lie in the same target namespace as the redefining schema.

The *redefine* mechanism includes all of the definitions and declarations in the file specified in the *schemaLocation* attribute. It is then possible to redefine types by deriving new types based on included types, with the same name.

Non-normative example of redefinition:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
      targetNamespace="urn:mpeg:mpeg7:schema:2001">

<redefine schemaLocation="urn:mpeg:mpeg7:schema:2001/MDS.xsd">

<!-- redefinition of PersonNameType -->
  <complexType name="mpeg7:PersonNameType">
      <complexContent>
         <extension base="mpeg7:PersonNameType">
            <sequence>
                <element name="Affiliation" type="string"/>
             </sequence>
         </extension>
      </complexContent>
   </complexType>

</schema>
```

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A.2.10 Identity Constraint Definitions

A.2.10.1 Specifying Uniqueness

Using the *unique* element, it is possible within XML Schema to indicate that a particular attribute or element must be unique within a certain scope. You must first define the scope by selecting a set of elements using the *selector* tag and an XPath expression [10]. Then you must identify the attributes or child elements within the selected scope which are to be unique. This is done using the *field* tag and XPath expressions relative to the selected scope. By setting multiple field tags within the same scope, it is possible to specify that combinations of values must be unique. In the example below, we have specified that the combination of the values of the *Title* element and *CatalogueId* attribute for each *NewsDoc* will provide a unique id to each news item.

Non-normative example of unique specification:

```
<unique name="NewsId">
   <selector xpath="VideoCatalogue/NewsCatalogue/NewsDoc"/>
   <field xpath="Title"/>
   <field xpath="@catalogueId"/>
</unique>
```

A.2.10.2 Defining Keys and their References

It is also possible within XML Schema to define elements or attributes or combinations of elements and attributes as keys – i.e., the key's value must be unique and non nullable. The name associated with a key can then be referenced from elsewhere in the schema to ensure that matching, unique and non-null values must exist for other elements, attributes or combinations elsewhere in the schema.

In the example below we use the *key* element to specify that the combination of the *Title* element and the *Category* attribute in the selector path must be unique within the scope in which it is declared.

Non-normative example of key definition:

```
<key name="NewsKey">
    <selector xpath="VideoCatalogue/NewsCatalogue/NewsDoc"/>
    <field xpath="Title"/>
    <field xpath="@category"/>
</key>
```

Using the *keyref* tag, it is possible to specify links between corresponding unique, non-nullable combinations in different locations in a schema. In the example below we specify the constraint that for each value of newsRef (pair of titleRef and categoryRef attribute values) there must be a newsKey with the same value. The values in the two attributes, titleRef and categoryRef, combined, are a reference, newsRef, to a key defined by newsKey (which is the combination of Title and Category as shown above).

Non-normative example of *keyref:*

```
<keyref name="NewsRef" refer="NewsKey">
     <selector xpath="Library/CheckoutRegister/NewsItem"/>
     <field xpath="@titleRef"/>
     <field xpath="@categoryRef/>
<keyref>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)

Overview of XML Schema Datatypes

B.1 Introduction

This Annex describes the facilities for defining datatypes within the DDL. MPEG-7 data typing facilities are based on XML Schema: Datatypes [5], but with extensions to satisfy MPEG-7-specific DDL requirements.

Section B.2 describes the data type facilities provided by XML Schema: Datatypes. Section B.2.1 describes the built-in data types. Section B.2.2 describes the built-in derived data types. Section B.2.3 describes the mechanisms for defining customized derived data types.

Subclause 5.2 describes the datatyping extensions required to satisfy MPEG-7 requirements. These consist of providing the following two built-in datatypes, derived by applying the *pattern* facet to the primitive *string* datatype:

- BasicTimePoint;
- BasicDuration.

B.2 Detailed Description

B.2.1 Built-in Primitive Datatypes

The following built-in, primitive, atomic data types are provided within XML Schema:Datatypes [6]:

- **string** – character strings;
- **boolean** – {true|false};
- **decimal** – arbitrary precision decimal numbers;
- **float** – single-precision 32-bit floating point type;
- **double** – double-precision 64-bit floating point type;
- **duration** – duration of time, represented using the [ISO 8601] extended format PnYn MnDTnH nMnS, where nY represents the number of years, nM the number of months, nD the number of days, 'T' is the date/time separator, nH the number of hours, nM the number of minutes and nS the number of seconds. The number of seconds can include decimal digits to arbitrary precision;
- **dateTime** – a specific instant of time, represented using the [ISO 8601] extended format CCYY-MM-DDThh:mm:ss where "CC" represents the century, "YY" the year, "MM" the month and "DD" the day, "T" is the date/time separator and "hh", "mm", "ss" represent hour, minute and second respectively;
- **time** – an instant of time that recurs every day represented using hh:mm:ss.sss with optional following time zone indicator;
- **date** – calendar date represented using CCYY-MM-DD with an optional following time zone qualifier;
- **gYearMonth** – a specific gregorian month in a specific gregorian year represented using CCYY-MM;
- **gYear** – a gregorian calendar year represented using CCYY;
- **gMonthDay** – a gregorian date that recurs, specifically a day of the year such as the third of May. It is represented using –MM-DD;
- **gDay** – a gregorian day that recurs, specifically a day of the month such as the 5th of the month. It is represented using ---DD;

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- **gMonth** – a gregorian month that recurs every year. It is represented using --MM--;
- **hexBinary** – arbitrary hex-encoded binary data. For example, "0FB7" is a hex encoding for the 16-bit integer 4023 (whose binary representation is 111110110111);
- **base64Binary** – Base64-encoded arbitrary binary data;
- **anyURI** – a Uniform Resource Identifier Reference (URI) [9];
- **Qname** – XML qualified names;
- **NOTATION** – the NOTATION attribute type from [XML 1.0 (Second Edition)] [7].

B.2.2 Built-in Derived Datatypes

The following built-in atomic datatypes, which have been derived from the primitive data types (base types), are also provided by XML Schema:Datatypes [6]:

- **normalizedString** – white space normalized strings;
- **token** – tokenised strings;
- **language** – natural language identifiers as defined by [RFC 1766] [15];
- **NMTOKEN** – the NMTOKEN attribute type from [XML 1.0 (Second Edition)] [7];
- **NMTOKENS** – the NMTOKENs attribute type from [XML 1.0 (Second Edition)] [7];
- **Name** – XML Names as specified in [XML 1.0 (Second Edition)] [7];
- **NCName** – XML "non-colonized" Names;
- **ID** – the ID attribute type from [XML 1.0 (Second Edition)] [7];
- **IDREF** – the IDREF attribute type from [XML 1.0 (Second Edition)] [7];
- **IDREFS** – the IDREFS attribute type from [XML 1.0 (Second Edition)] [7];
- **ENTITY** – the ENTITY attribute type from [XML 1.0 (Second Edition)] [7];
- **ENTITIES** – the ENTITIES attribute type from [XML 1.0 (Second Edition)] [7];
- **integer** – derived from decimal by fixing the value of fractionDigits to be 0. This results in the standard mathematical concept of the integer numbers i.e., the infinite set {...,-2,-1,0,1,2,...};
- **nonPositiveInteger** – derived from integer by setting the value of maxInclusive to be 0. This results in the standard mathematical concept of the non-positive integers i.e., the infinite set {...,-2,-1,0};
- **negativeInteger** – derived from nonPositiveInteger by setting the value of maxInclusive to be -1. This results in the standard mathematical concept of the negative integers i.e., the infinite set {...,-2,-1};
- **long** – derived from integer by setting the value of maxInclusive to 9223372036854775807 and minInclusive to –9223372036854775808;
- **int** – derived from long by setting the value of maxInclusive to 2147483647 and minInclusive to –2147483648;
- **short** – derived from int by setting maxInclusive to be 32767 and minInclusive to –32768;
- **byte** – derived from short by setting the value of maxInclusive to 127 and minInclusive to –128;
- **nonNegativeInteger** – derived from integer by setting the value of minInclusive to 0. This results in the standard mathematical concept of the non-negative integers i.e., the infinite set {0,1,2,...};
- **unsignedLong** – derived from nonNegativeInteger by setting the value of maxInclusive to 18446744073709551615;
- **unsignedInt** – derived from unsignedLong by setting the value of maxInclusive to be 4294967295;
- **unsignedShort** – derived from unsignedInt by setting the value of maxInclusive to 65535;

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- **unsignedByte** – derived from unsignedShort by setting the value of maxInclusive to 255;
- **positiveInteger** – derived from nonNegativeInteger by setting the value of minInclusive to 1. This results in the standard mathematical concept of the positive integer numbers i.e., the infinite set {1,2,...}.

B.2.3 The List Datatype

A *list* datatype is derived from an atomic datatype and has values which consist of a finite-length (possibly empty) sequence of whitespace separated values of the base type. The following constraining facets may be applied to a list datatype:

- length
- maxLength
- minLength
- enumeration

Non-normative example:

```
<simpleType name="listOfFloat">
  <list itemType="float"/>
</simpleType>

<simpleType name="listOf5Floats">
  <restriction base="mpeg7:listOfFloat">
     <length value="5"/>
  </restriction>
</simpleType>
```

B.2.4 The Union Datatype

Union types enable element or attribute values to be one or more instances of one type drawn from the union of multiple atomic and list types. The *pattern* and *enumeration* facets are applicable to union types.

Non-normative example:

```
<element name="Unsigned6OrDirection">
  <simpleType>
     <union memberTypes="unsigned6 directionType"/>
  </simpleType>
</element>
```

Non-normative example of a valid instance:

```
<Unsigned6OrDirection>45</Unsigned6OrDirection>
<Unsigned6OrDirection>uniDirectional</Unsigned6OrDirection>
```

B.2.5 Defining Derived Datatypes

A derived datatype can be defined from a primitive datatype or another derived datatype by adding constraining facets. Below are descriptions of the facets which are provided to generate customized datatypes.

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.2.5.1 Bounds facets

Bounds facets include *minInclusive, minExclusive* (lower bound), *maxInclusive, maxExclusive* (upper bound). They can be used to derive new data types by constraining the bounds of the value space of "ordered" base types.

Non-normative example:

```
<simpleType name="heightInInches">
   <restriction base ="float">
        <minInclusive value="0.0"/>
        <maxInclusive value="120.0"/>
   </restriction>
</simpleType>
```

B.2.5.2 Numeric facets

The numeric facets are *totalDigits* and *fractionDigits. totalDigits* is the maximum number of digits in values of datatypes derived from *decimal*. The value of *totalDigits* must be a positiveInteger. fractionDigits is the maximum number of digits in the fractional part of values of datatypes derived from decimal. The value of fractionDigits must be a nonNegativeInteger.

```
<simpleType name="mySizeType">
  <restriction base="decimal">
      <totalDigits value="8"/>
      <fractionDigits value="2" fixed="true"/>
  </restriction>
</simpleType>

<attribute name="size" type="mpeg7:mySizeType"/>
```

B.2.5.3 Pattern Facet

pattern - a regular expression used to constrain the format of a string;

Non-normative example:

```
<simpleType name="myPhoneNum">
  <restriction base="string">
      <pattern value="\d{3}-\d{4}"/>
  </restriction>
</simpleType>
```

B.2.5.4 Enumeration Facet

enumeration - constrains the value space to a specified set of values;

Non-normative example:

```
<simpleType name="modelTypes">
   <restriction base ="string">
        <enumeration value="Analytic"/>
        <enumeration value="Synthetic"/>
        <enumeration value="Correspondence"/>
   </restriction>
</simpleType>
```


© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.2.5.5 Length Facets

length, minlength, maxlength – applies to certain datatypes to constrain length, minimum length or maximum length. They can only have values which are nonNegativeIntegers;

Non-normative example:

```
<simpleType name="myString">
   <restriction base ="string">
       <maxLength value="24"/>
   </restriction>
</simpleType>
```

B.2.5.6 Whitespace Facet

whitespace – enables users to specify how whitespace should be handled in the value space. Possible values are *{preserve, replace* or *collapse}.*

Non-normative example:

```
<simpleType name="token">
  <restriction base="CDATA">
     <whiteSpace value="collapse"/>
  </restriction>
</simpleType>
```

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C
(normative)

XML Schema for MPEG-7 Extensions

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ################################################################## -->
<!--  ISO/IEC 15938 Information Technology - Multimedia Content Description
Interface  -->
<!-- Part 2: Description Definition Language (ISO/IEC 15938-2)              -->
<!-- ################################################################## -->

<schema targetNamespace="urn:mpeg:mpeg7:schema:2001"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <annotation>
     <documentation>
        This document contains tools defined as MPEG-7 specific extension of
        XML Schema in ISO/IEC 15938-2
     </documentation>
  </annotation>
  <!-- ######################################### -->
  <!-- Definition of 'mpeg7:dim" for Matrix Datatype   -->
  <!-- ######################################### -->

  <simpleType name="listOfPositiveIntegerForDim">
     <list itemType="positiveInteger"/>
  </simpleType>

  <attribute name="dim">
     <simpleType>
        <restriction base="mpeg7:listOfPositiveIntegerForDim">
           <minLength value="1"/>
        </restriction>
     </simpleType>
  </attribute>

  <!-- ##################################### -->
  <!-- Definition of MPEG-7 Datatype Extensions  -->
  <!-- ##################################### -->
  <simpleType name="basicTimePointType">
     <restriction base="string">
        <pattern value=="(\-?\d+(\-\d{2}(\-\d{2})?)?)?(T\d{2}(:\d{2}(:\d{2}
                  (:\d+(\.\d{2})?)?)?)?)?(F\d+)?((\-|\+)\d{2}:\d{2})?"/>
     </restriction>
  </simpleType>

  <simpleType name="basicDurationType">
     <restriction base="string">
        <pattern value="\-?P(\d+D)?(T(\d+H)?(\d+M)?(\d+S)?(\d+N)?
                  (\d{2}f)?)?(\d+F)?((\-|\+)\d{2}:\d{2}Z)?"/>
      </restriction>
  </simpleType>
</schema>
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
(informative)

Patent statements

The International Organization for Standardization and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 15938 may involve the use of patents.

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from the companies listed below.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 15938 may be the subject of patent rights other than those identified in this annex. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Company	MPEG-7 Patent statements					
	Systems	DDL	Visual	Audio	MDS	Reference software
Bosch	X	X	X	X	X	X
Canon				X		
CIE	X	X	X	X	X	X
Denso	X				X	
Ericsson	X	X	X	X	X	X
ETRI	X	X	X	X	X	X
Expway	X	X	X	X	X	X
FhG				X		
Geocast	X	X	X	X	X	
HHI			X			X
Hitachi	X	X	X	X	X	X
Hyundai			X		X	X
IBM	X	X	X	X	X	X
JVC	X	X	X	X	X	X
KDDI			X	X	X	X
LG Electronics	X	X	X	X	X	X
Matsushita	X	X	X	X	X	X
Mitsubishi Electric	X	X	X	X	X	X
NEC						

© ISO/IEC 2002 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NHK	X	X	X	X	X	X
Philips	X	X	X	X	X	X
Ricoh	X	X	X	X	X	X
Samsung	X	X	X	X	X	X
Sharp	X		X	X	X	X
Siemens	X	X	X	X	X	X
Sony	X	X	X	X	X	X
Toshiba	X	X	X	X	X	X
Vivcom					X	X

© ISO/IEC 2002 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Bibliography

[1] ISO/IEC JTC1/SC29/WG11N4317, *MPEG-7 Requirements Document V.15*, F. Pereira ed., Sydney, July 2001

[2] ISO/IEC JTC1/SC29/WG11N2730, *Results of MPEG-7 Technology Proposal Evaluations and Recommendations,* MPEG7 Requirements Group, Seoul, March 1999

[3] *XML Schema Part 0: Primer*, W3C Recommendation, 2 May 2001 http://www.w3.org/TR/xmlschema-0/

[4] *XML Schema Part 1: Structures*, W3C Recommendation, 2 May 2001 http://www.w3.org/TR/xmlschema-1/

[5] *XML Schema Part 2: Datatypes*, W3C Recommendation, 2 May 2001 http://www.w3.org/TR/xmlschema-2/

[6] *Resource Description Framework (RDF) Schema Specification 1.0*, W3C Candidate Recommendation 27 March 2001 http://www.w3.org/TR/rdf-schema

[7] *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C recommendation, 6 October 2000 http://www.w3.org/TR/REC-xml

[8] Namespaces in XML, W3C Recommendation, 14 January 1999 http://www.w3.org/TR/REC-xml-names/

[9] IETF (Internet Engineering Task Force) RFC 2396: *Uniform Resource Identifiers (URI): Generic Syntax*, eds. T. Berners-Lee, R. Fielding, L. Masinter. August 1998 http://www.ietf.org/rfc/rfc2396.txt

[10] *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999 http://www.w3.org/TR/xpath

[11] ISO 3166 *Country Codes* http://www.iso.org/mara/iso3166

[12] ISO 4217 *Codes for the representation of currencies and funds*, http://www.bsi-global.com/iso4217currency

[13] *IANA List of Mime Types* ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types

[14] *IANA List of Character sets* http://www.iana.org/assignments/character-sets

[15] RFC 1766: *Tags for the Identification of Languages*. H. Alvestrand, ed. 1995. http://www.ietf.org/rfc/rfc1766.txt

© ISO/IEC 2002 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.040

Price based on 37 pages

© ISO/IEC 2002 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale