

# IEEE P1484.12.3, Draft 8

## Draft Standard for Learning Technology— Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata

Sponsor

**Learning Technology Standards Committee**  
of the  
**IEEE Computer Society**

**Abstract:** This Standard defines a World Wide Web Consortium (W3C) Extensible Markup Language (XML) Schema definition language binding of the learning object metadata (LOM) data model defined in IEEE 1484.12.1–2002 *Standard for Learning Object Metadata*. The purpose of this Standard is to allow the creation of LOM instances in XML. This allows for interoperability and the exchange of LOM XML instances between various systems. This Standard uses the W3C XML Schema definition language to define the syntax and semantics of the XML encodings.

**Keywords:** 1484.12.1–2002, Extensible Markup Language, learning object metadata, LOM, LOM XML instance, LOM XML Schema binding, metadata, W3C XML Schema definition language, XML, XML Schema definition, XSD.

---

The Institute of Electrical and Electronics Engineers, Inc.  
Three Park Avenue, New York, NY 10016-5997, USA

Copyright © 2005 by the Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved. Published **date to be supplied**. Printed in the United States of America.

This document is an unapproved draft of a proposed IEEE Standard. As such, this document is subject to change. USE AT YOUR OWN RISK! Because this is an unapproved draft, this document must not be utilized for any conformance/compliance purposes. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only. Prior to submitting this document to another standards development organization for stan-

standardization activities, permission must first be obtained from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department. Other entities seeking permission to reproduce this document, in whole or in part, must obtain permission from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department.

IEEE Standards Department  
Standards Licensing and Contracts  
445 Hoes Lane, P.O. Box 1331  
Piscataway, NJ 08855-1331, USA

## Introduction

(This introduction is not part of P1484.12.3, *Draft Standard for Learning Technology—Extensible Markup Language (XML) Schema Binding for Learning Object Metadata*.)

This Standard defines World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent learning object metadata (LOM) instances as defined in IEEE 1484.12.1–2002, *Standard for Learning Object Metadata*. This Standard defines the structure and constraints of the XML 1.1 documents in W3C XML Schema definition language.

The purpose of this Standard is to allow the creation of interoperable LOM instances in XML. This Standard uses the W3C XML Schema definition language as the encoding. This allows for interoperability and the exchange of LOM XML instances between various systems.

## Participants

At the time this Standard was completed, the working group had the following membership:

Wayne Hodgins, <i>Chair</i> Erik Duval and Scott Lewis, <i>Technical Editors</i>		
Mitchell Bonnett	Peter Greene	Mikael Nilsson
Debbie Brown	Thomas Herrmann	Lassi Nirhamo
Tsz Chan	Cord Hockemeyer	Claude Ostyn
Michael Collett	Wayne Hodgins	Miroslav Pavlovic
Ingo Dahn	Robert Bruce Kelsey	Klaus Rapf

Geoffrey Darnton	Mark Knight	Daniel Rehak
Maulik Dave	David Leciston	Tyde Richards
Marco De Vos	Gregory Luri	Robby Robson
Guru Dutt Dhingra	Ryan Madron	Thomas Starai
Sam Dooley	Faramarz Maghsoodlou	Gerald Stueve
Erik Duval	Jon Mason	Brian Taliesin
Kameshwar Erankik	William Melton	Schawn Thropp
Frank Farance	George Miao	Mark Tillinghast
David Fore	Rajesh Moorkath	Li Zhang
Ernesto Garcia	Brandon Muramatsu	Philomena Zimmerman
Matthew Gream	Boyd Nielsen	

The following members of the balloting committee voted on this Standard. Balloters may have voted for approval, disapproval, or abstention. [To be supplied by the IEEE editor.](#)

Also included are the following nonvoting IEEE-SA Standards Board liaisons: [To be supplied by the IEEE Editor.](#)

## Contents

1. Overview .....	1
1.1 Scope .....	1
1.2 Purpose .....	1
2. Normative references .....	1
3. Definitions and acronyms .....	2
3.1 Definitions .....	2
3.2 Acronyms and abbreviations .....	4
4. Conformance .....	4
4.1 Strictly conforming LOM XML instances .....	5
4.2 Conforming LOM XML instances .....	5
5. LOM XML Schema binding definition .....	5
5.1 General information .....	5
5.1.1 Smallest permitted maximums .....	6
5.1.2 LOM XML instance conformance constraints .....	6
5.1.3 Extension support .....	6
5.2 LOM namespaces .....	7
5.3 Table format and organization .....	8
5.4 LOM .....	9
5.4.1 General .....	10
5.4.2 Life Cycle .....	12
5.4.3 Meta-Metadata .....	14
5.4.4 Technical .....	16
5.4.5 Educational .....	19
5.4.6 Rights .....	22
5.4.7 Relation .....	23
5.4.8 Annotation .....	25
5.4.9 Classification .....	25
5.5 Common data types and elements .....	27
5.5.1 CharacterString .....	27
5.5.2 DateTime .....	27
5.5.3 Duration .....	28
5.5.4 LangString .....	30
5.5.5 Vocabulary .....	32
Annex A (informative) Bibliography .....	33
Annex B (informative) Internet availability of XSD files .....	34
Annex C (informative) XSD file descriptions .....	35
C.1 Composite XSDs .....	36
C.1.1 lomStrict.xsd .....	37
C.1.2 lomCustom.xsd .....	37
C.1.3 lomLoose.xsd .....	37
C.2 Component XSDs .....	38
Annex D (informative) Enabling extended data elements and attributes .....	41

D.1 Enabling extended data elements .....	41
D.2 Enabling extended attributes .....	42
Annex E (informative) XSD implementation choices .....	44
E.1 Data types .....	44
E.2 Elements .....	45
E.3 Aggregates .....	46
E.3.1 Using the XML Schema element sequence .....	46
E.3.2 Using the XML Schema element all .....	47
E.3.3 Using the XML Schema element choice .....	48
E.3.4 Using the XML Schema element choice with the XML Schema constraint unique .....	49
E.3.5 Summary .....	49
E.4 Vocabularies .....	50
E.4.1 vocab/loose.xsd .....	51
E.4.2 vocab/strict.xsd .....	51
E.4.3 vocab/custom.xsd .....	52
E.5 Additional notes .....	53

# Draft Standard for Learning Technology— Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata

## 1. Overview

The scope and purpose of this Standard are discussed in 1.1 and 1.2.

### 1.1 Scope

This Standard defines World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent learning object metadata (LOM) instances as defined in IEEE 1484.12.1–2002, *Standard for Learning Object Metadata*.<sup>1</sup> This Standard defines the structure and constraints of the XML 1.1 documents in W3C XML Schema definition language. An implementation that conforms to this Standard shall conform to IEEE 1484.12.1–2002.

### 1.2 Purpose

The purpose of this Standard is to allow the creation of interoperable LOM instances in XML. This Standard uses the W3C XML Schema definition language as the encoding. This allows for interoperability and the exchange of LOM XML instances between various systems.

## 2. Normative references

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

IEEE 1484.12.1–2002, Standard for Learning Object Metadata.

---

<sup>1</sup> For information on normative references, see Clause 2.

IETF RFC 2048:1996, Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures.

IETF RFC 2426:1998, vCard MIME Directory Profile.

ISO 639–1, Code for the representation of names of languages – Part 1: Alpha-2 code.

ISO 639–2, Codes for the representation of names of languages – Part 2: Alpha-3 code.

ISO 3166–1, Codes for the representation of names of countries and their subdivisions – Part 1: Country codes.

ISO/IEC 10646–1, Information technology—Universal multiple-octet coded character set – Part 1: Architecture and basic multilingual plane.

W3C Recommendation (28 October 2004), XML Schema Part 1: Structures Second Edition.

W3C Recommendation (28 October 2004), XML Schema Part 2: Datatypes Second Edition.

W3C Recommendation (4 February 2004), Namespaces in XML 1.1.

### 3. Definitions and acronyms

Definitions and acronyms are defined in 3.1 and 3.2, respectively.

#### 3.1 Definitions

For purposes of this Standard, the following terms and definitions apply. IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [B1]<sup>1</sup>, should be referenced for terms not defined in this subclause.

**aggregate element:** A LOM data element that contains other LOM data elements called subelements. *See also:* **subelement**.

**component Extensible Markup Language Schema definition (component XSD):** An Extensible Markup Language Schema definition that defines a constituent of a composite schema. *See also:* **composite Extensible Markup Language Schema definition**.

**composite Extensible Markup Language Schema definition (composite XSD):** An Extensible Markup Language Schema definition that is a structure made up of distinct component XML Schema definitions. *See also:* **component Extensible Markup Language Schema definition**.

---

<sup>1</sup> The numbers in brackets correspond to those of the bibliography in Annex A.

**content model:** A framework that identifies the makeup (i.e., data types, multiplicity constraints, ordering) of a specific model.

**data type:** A property of distinct values, indicating common features of those values and operations on those values.

**extended data element:** A data element that is not defined in the LOMv1.0 base schema. *See also:* **LOMv1.0 base schema**.

**extended vocabulary:** A value space that is not defined in the LOMv1.0 base schema for a LOM data element of type Vocabulary. *See also:* **LOMv1.0 base schema**.

**Extensible Markup Language Information Set (XML Infoset):** An abstract data set that provides a consistent set of definitions for use in other specifications that need to refer to the information in a well-formed XML document (W3C, XML Information Set). *See also:* **post-schema-validation infoset**.

**Extensible Markup Language Schema binding (XML Schema binding):** A textual representation of the behaviors, attributes, and value space of a data-model element in W3C XML Schema definition language.

**learning object:** In this Standard, any entity, digital or non-digital, that may be used for learning, education, or training (IEEE 1484.12.1–2002).

**learning object metadata data element (LOM data element):** A data element for which the name, explanation, size, ordering, value space, and data type are defined in IEEE 1484.12.1–2002. *See also:* **LOMv1.0 base schema**.

**learning object metadata Extensible Markup Language instance (LOM XML instance):** A collection of metadata for a learning object that conforms to IEEE 1484.12.1–2002, that is represented in XML, and that adheres to the requirements and constraints of the XML Schema binding defined in this Standard.

**LOMv1.0 base schema:** A structured collection of standard data items, including their data types, multiplicities, and container/component relationships, as defined in IEEE 1484.12.1–2002, Clause 6.

**mixed content:** An element type has mixed content when elements of that type may contain character data, optionally interspersed with child elements (W3C, Extensible Markup Language [XML] 1.1).

**Multipurpose Internet Mail Extensions type (MIME type):** A standard way of classifying content types on the Internet.

**post-schema-validation infoset:** A transformed version of the XML infoset of a document produced by a conforming W3C XML Schema processor. W3C XML Schema defines the data elements added to the XML infoset of the original document during validation to produce

the post-schema-validation info set. *See also:* **Extensible Markup Language information set**.

**subelement:** A LOM data element that is contained within another LOM data element called an aggregate element. A subelement may contain other subelements, in which case it is also an aggregate element. *See also:* **aggregate element**.

**token:** A character string. The value space of a token is the set of strings that do not contain the line feed (#xA) or tab (#x9) characters, that have no leading or trailing spaces (#x20), and that have no internal sequences of two or more spaces. The lexical space of a token is the set of strings that do not contain the line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20), and that have no internal sequences of two or more spaces. The base type of a token is `normalizedString`. For purposes of this Standard, tokens are case sensitive. (Adapted from W3C, XML Schema Part 2.) *See also:* **token set**.

**token set:** A set of tokens in which each token is unique. *See also:* **token**.

**uniqueness constraint:** A restriction placed on a LOM data element that enforces that the LOM data element is unique within a LOM XML instance. If a LOM data element has a uniqueness constraint, it appears in a LOM XML instance zero or one time. *See also:* **learning object metadata data element**.

**value space:** The set of values for a given data type (ISO/IEC 11404:1996).

NOTE—In IEEE 1484.12.1–2002, a value space is typically enumerated outright or defined by reference to another standard or specification.

## 3.2 Acronyms and abbreviations

IANA: Internet Assigned Numbers Authority  
LOM: Learning Object Metadata  
MIME: Multipurpose Internet Mail Extensions  
PSVI: post-schema-validation info set  
SPM: smallest permitted maximum  
UTC: coordinated universal time  
W3C: World Wide Web Consortium  
XML: Extensible Markup Language  
XSD: XML Schema definition

## 4. Conformance

Conformance to this Standard is discussed in 4.1 and 4.2. In 4.1 and 4.2, *strictly conforming LOM XML instance* and *conforming LOM XML instance* refer to the metadata represented in the LOM XML instance prior to any processing of the LOM XML instance.

In this Standard, “shall” is to be interpreted as a requirement on an implementation; “shall not” is to be interpreted as a prohibition.

## 4.1 Strictly conforming LOM XML instances

A *strictly conforming* LOM XML instance

- Shall be a strictly conforming LOM instance as defined in IEEE 1484.12.1–2002;
- Shall conform to the requirements of Clause 5 of this Standard;
- Shall not include vocabulary values that are not defined in Clause 5 of this Standard;
- Shall not include XML elements or attributes that are not defined in Clause 5 of this Standard; and
- Shall not include mixed content.

## 4.2 Conforming LOM XML instances

A *conforming* LOM XML instance

- Shall be a conforming LOM instance as defined in IEEE 1484.12.1–2002;
- Shall conform to the requirements of Clause 5 of this Standard;
- May include vocabulary values that are not defined in Clause 5 of this Standard;
- May include XML elements and attributes that are not defined in Clause 5 of this Standard by using the extension mechanism described in 5.1.3; and
- May include mixed content.

## 5. LOM XML Schema binding definition

The LOM XML Schema binding is defined in 5.1 – 5.5. Subclause 5.4 lists the LOM data elements defined in the LOMv1.0 base schema. For each LOM data element, the corresponding XML element for the XML Schema binding is described.

### 5.1 General information

As defined in IEEE 1484.12.1–2002, all LOM data elements are optional. An XML Schema definition (XSD) for the LOMv1.0 base schema shall not require any LOM data to be present in a LOM XML instance.

The LOMv1.0 base schema defines an aggregation relationship between data items. An XSD for LOM defines an aggregation relationship between subelements that maintains the relationship defined in the LOMv1.0 base schema. In a LOM XML instance, a subelement shall appear only within the aggregate element of the aggregation relationship. For example, in 5.4.3.1, the Identifier subelement appears by definition as a component of the Meta-Metadata

element described in 5.4.3. The presence of the subelement automatically implies the presence of the aggregate element to which the subelement belongs.

The LOMv1.0 base schema does not define any sequencing of LOM data elements, except for their aggregation relationships. Therefore, an XML Schema binding of the LOMv1.0 base schema shall not define any sequence of the elements in a LOM XML instance, except for their aggregation relationships.

NOTE—XML names used in this Standard are derived from LOM data element names by concatenating the LOM data element names and using *lowerCamelCase* capitalization.

### 5.1.1 Smallest permitted maximums

The W3C XML Schema definition language does not support the concept of smallest permitted maximums (SPMs) as defined in IEEE 1484.12.1–2002. The W3C XML Schema definition language does provide a means for restricting maximum lengths (i.e., `maxLength` for character strings) and maximum numbers of occurrences (i.e., `maxOccurs` for aggregate elements); however, these restrictions are not compatible with the definition of SPM.

If a LOM XML instance contains more than the SPM number of occurrences of a LOM data element, then an application shall process at least the SPM number of occurrences of the LOM data element. If a LOM XML instance contains more than the SPM number of characters in a character string, then an application shall process at least the SPM number of characters in the character string. (SPMs for LOM data elements and character strings are defined in IEEE 1484.12.1–2002.)

NOTE—To encourage interoperability, creators of XSDs intended to validate LOM XML instances against IEEE 1484.12.1–2002 should not use `maxOccurs` or `maxLength` restrictions or should set the restricted limits to `unbounded`.

### 5.1.2 LOM XML instance conformance constraints

Constraints on the validity of a *conforming* or *strictly conforming* LOM XML instance with respect to IEEE 1484.12.1–2002 are defined throughout 5.4 and 5.5. All *conforming* and *strictly conforming* LOM XML instances shall satisfy these constraints.

NOTE—Some applications may use an XSD to validate a LOM XML instance. If an application uses an XSD to validate a LOM XML instance, the post-schema-validation infoset (PSVI) of a LOM XML instance is insufficient to determine the validity of the LOM XML instance with respect to IEEE 1484.12.1–2002. Use of an XSD to validate a LOM XML instance does not relieve an application from any requirements on enforcing the constraints that appear throughout 5.4 and 5.5.

### 5.1.3 Extension support

Extensions to the LOMv1.0 base schema are permitted by this Standard. As defined in IEEE 1484.12.1–2002, extensions to the LOMv1.0 base schema shall retain the data types and value

spaces of LOM data elements from the LOMv1.0 base schema and shall not define data types or value spaces for aggregate elements in the LOMv1.0 base schema.

This Standard permits the following types of extensions for *conforming* but not for *strictly conforming* LOM XML instances:

- *LOMv1.0 base schema XML element extension:* LOMv1.0 base schema data elements may be extended. These extensions are new XML elements defined in a namespace other than the namespaces defined in 5.2. Extended data elements may be added to any LOM data element that is defined as an aggregate element. Extended data elements shall not be added to LOM data elements that are not aggregate elements (see 5.4 and 5.5).
- *LOMv1.0 base schema XML attribute extension:* LOM data elements may be extended by defining attributes for the LOM data elements. These extensions are XML attributes defined in a namespace other than the namespaces defined in 5.2. These attributes may be added to any LOM data element (see 5.4 and 5.5).
- *LOMv1.0 base schema vocabulary data type extension:* LOMv1.0 base schema vocabularies may be extended. These extensions are additional tokens for LOM data elements of type Vocabulary (see 5.4). If vocabularies are extended, the source of the additional vocabulary tokens should be identified and shall not be LOMv1.0.

## 5.2 LOM namespaces

All of the LOM data elements defined in 5.4 and 5.5 shall be defined in the namespace `http://ltsc.ieee.org/xsd/LOM`. This namespace is reserved by this Standard and shall not be used to represent any other data element (see W3C Recommendation, Namespaces in XML 1.1.). This namespace shall not be used to define extensions to a LOM XML instance.

The following namespaces are reserved by this Standard and shall not be used to represent any other data element. The namespaces are used in the XSDs provided with this Standard (see Annex B).

- `http://ltsc.ieee.org/xsd/LOM/custom`
- `http://ltsc.ieee.org/xsd/LOM/unique`
- `http://ltsc.ieee.org/xsd/LOM/vocab`
- `http://ltsc.ieee.org/xsd/LOM/extend`

If an organization provides extensions to a LOM XML instance, the organization should define the namespace for these extensions.

## 5.3 Table format and organization

This Standard uses tables to describe the requirements for a LOM XML instance. The tables express requirements for each of the LOM data elements in the LOMv1.0 base schema. These requirements are

- *LOM data element*: The name of the LOM data element in the LOMv1.0 base schema.
- *XML name*: The name used in a LOM XML instance that conforms to this Standard for the corresponding LOM data element. A dash ("-") in the tables in 5.5, which describe common LOM data types and LOM data elements, indicates that no XML name exists because the LOM name refers to a data type instead of a LOM data element.
- *Subelements*: A listing of subelements of the LOM data element. An entry of "None" indicates that the LOM data element does not have subelements. If present, the subelements listed shall be contained by their associated aggregate element. This ensures that the aggregation relationships of the LOMv1.0 base schema components are enforced. The order of appearance of subelements shall not be significant.
- *Min*: If a LOM data element is not the top-level LOM element, the requirement on the minimum number of times the LOM data element may appear in a LOM XML instance in the context of the LOM data element's aggregate element. For the LOM element, the requirement on the minimum number of times the LOM data element may appear in a LOM XML instance.
- *Max*: If a LOM data element is not the top-level LOM element, the requirement on the maximum number of times the LOM data element may appear in a LOM XML instance in the context of the LOM data element's aggregate element. For the LOM element, the requirement on the maximum number of times the LOM element may appear in a LOM XML instance. An infinity symbol ("∞") indicates that the maximum number of times is unbounded. Values in parenthesis are SPMs.
- *Order*: Indicates whether the order of the values is significant. This Standard uses three designators for the order: "Ordered", "Unordered", and "Unspecified". "Ordered" indicates that the ordering of the values is significant. "Unordered" indicates that the ordering of the values is not significant. For a LOM data element that has a multiplicity of zero or one, the concept of order has no meaning, which is indicated by "Unspecified". *Note*: The significance of the order of a list of ordered values is determined by the implementation.
- *LOM data type*: Indicates whether the LOM data type is LangString, DateTime, Duration, Vocabulary, or CharacterString, or is Unspecified. If the LOM data element is an aggregate element, it has no data type and, therefore, is described as "Unspecified". If the data type is Vocabulary, the permissible values from the LOMv1.0 base schema are listed.

Each table that describes an aggregate element includes the direct-descendant subelements of that aggregate element. For example, Table 1 in 5.4 describes the LOM data element and includes the LOM data element's direct-descendant subelements, such as General and Life Cycle. However, the table does not include the subelements of the General element, such as Identifier and Title. If a subelement is also an aggregate element, a separate table describes the subelement with its direct-descendant subelements. If information beyond that provided by IEEE 1484.12.1–2002 is needed to define subelements in a LOM XML instance, that information is provided in subclauses of the aggregate element description.

With the exception of the order for Description, Educational, Identifier, and Intended End User Role, in the case of any discrepancy in the tables in 5.4 and 5.5 and IEEE 1484.12.1–2002, the values from IEEE 1484.12.1–2002 shall be used.

Although the Max column may indicate that a subelement may appear in an aggregate element an unbounded maximum number of times, an application that processes a LOM XML instance shall process at least the SPM number of subelements (see 5.1.1).

## 5.4 LOM

Table 1 describes the LOM element and its direct-descendant subelements.

**Table 1—The LOM element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
LOM	lom	General Life Cycle Meta-Metadata Technical Educational Rights Relation Annotation Classification	1	1	Unspecified	Unspecified
General	general	See 5.4.1	0	1	Unspecified	Unspecified
Life Cycle	lifeCycle	See 5.4.2	0	1	Unspecified	Unspecified
Meta-Metadata	metaMetadata	See 5.4.3	0	1	Unspecified	Unspecified
Technical	technical	See 5.4.4	0	1	Unspecified	Unspecified
Educational	educational	See 5.4.5	0	$\infty$ (100)	Unordered	Unspecified
Rights	rights	See 5.4.6	0	1	Unspecified	Unspecified
Relation	relation	See 5.4.7	0	$\infty$ (100)	Unordered	Unspecified

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Annotation	annotation	See 5.4.8	0	$\infty$ (30)	Unordered	Unspecified
Classification	classification	See 5.4.9	0	$\infty$ (40)	Unordered	Unspecified

### Namespace declaration

The LOM XML instance shall include a namespace declaration that declares the LOM namespace for the LOM element and its components. The LOM namespace shall be "http://ltsc.ieee.org/xsd/LOM" as defined in 5.2.

The namespace declaration used is in conformance with "W3C Recommendation, Namespaces in XML 1.1." Examples are shown in Figures 1 and 2, respectively.

```
<lom xmlns="http://ltsc.ieee.org/xsd/LOM">
. . .
</lom>
```

**Figure 1—An example default namespace declaration**

```
<lom:lom xmlns:lom="http://ltsc.ieee.org/xsd/LOM">
. . .
</lom:lom>
```

**Figure 2—An example prefix-specific namespace declaration**

NOTE—A namespace declaration allows an application to recognize a LOM XML instance as one that contains LOM data elements described in this Standard.

### 5.4.1 General

Table 2 describes the General element and its direct-descendant subelements.

**Table 2—The General element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
General	general	Identifier Title Language Description Keyword Coverage Structure Aggregation Level	0	1	Unspecified	Unspecified
Identifier	identifier	See 5.4.1.1	0	$\infty$ (10)	Unordered	Unspecified
Title	title	See 5.5.4	0	1	Unspecified	LangString
Language	language	None	0	$\infty$ (10)	Unordered	CharacterString
Description	description	See 5.5.4	0	$\infty$ (10)	Unordered	LangString
Keyword	keyword	See 5.5.4	0	$\infty$ (10)	Unordered	LangString
Coverage	coverage	See 5.5.4	0	$\infty$ (10)	Unordered	LangString
Structure	structure	See 5.5.5	0	1	Unspecified	Vocabulary
Aggregation Level	aggregationLevel	See 5.5.5	0	1	Unspecified	Vocabulary

**5.4.1.1 Identifier**

Table 3 describes the Identifier element and its direct-descendant subelements.

**Table 3—The Identifier element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Identifier	identifier	Catalog Entry	0	$\infty$ (10)	Unordered	Unspecified
Catalog	catalog	None	0	1	Unspecified	CharacterString
Entry	entry	None	0	1	Unspecified	CharacterString

### 5.4.1.2 Language

The Language element shall be a character string where the value of the character string shall be one of the following:

- The format and values described in 5.5.4.1; or
- The token none.

### 5.4.1.3 Structure

If the value space for the Source subelement of the Structure element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Structure element shall come from the following list of tokens:

- atomic
- collection
- networked
- hierarchical
- linear

### 5.4.1.4 Aggregation Level

If the value space for the Source subelement of the Aggregation Level element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement element of the Aggregation Level elements shall come from the following list of tokens:

- 1
- 2
- 3
- 4

## 5.4.2 Life Cycle

Table 4 describes the Life Cycle element and its direct-descendant subelements.

**Table 4—The Life Cycle element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Life Cycle	lifeCycle	Version Status Contribute	0	1	Unspecified	Unspecified

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Version	version	See 5.5.4	0	1	Unspecified	LangString
Status	status	See 5.5.5	0	1	Unspecified	Vocabulary
Contribute	contribute	See 5.4.2.2	0	$\infty$ (30)	Ordered	Unspecified

### 5.4.2.1 Status

If the value space for the Source subelement of the Status element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Status element shall come from the following list of tokens:

- draft
- final
- revised
- unavailable

### 5.4.2.2 Contribute

Table 5 describes the Contribute element and its direct-descendant subelements.

**Table 5—The Contribute element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Contribute	contribute	Role Entity Date	0	$\infty$ (30)	Ordered	Unspecified
Role	role	See 5.5.5	0	1	Unspecified	Vocabulary
Entity	entity	None	0	$\infty$ (40)	Ordered	CharacterString
Date	date	See 5.5.5	0	1	Unspecified	DateTime

#### 5.4.2.2.1 Role

If the value space for the Source subelement of the Role element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Role element shall come from the following list of tokens:

- author
- publisher
- unknown

- initiator
- terminator
- validator
- editor
- graphical designer
- technical implementer
- content provider
- technical validator
- educational validator
- script writer
- instructional designer
- subject matter expert

#### 5.4.2.2 Entity

The value held by the Entity element shall be a character string literal that is the canonical lexical representation of a valid vCard as defined in IETF RFC 2426:1998.

NOTE—IETF RFC 2426:1998 does not rely on XML syntax to express the internal structure of a valid vCard. At the time of publication of this Standard, no *de facto* standard W3C XML Schema definition language binding for the vCard specification existed.

#### 5.4.3 Meta-Metadata

Table 6 describes the Meta-Metadata element and its direct-descendant subelements.

**Table 6—The Meta-Metadata element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Meta-Metadata	metaMetadata	Identifier Contribute Metadata Schema Language	0	1	Unspecified	Unspecified
Identifier	identifier	See 5.4.3.1	0	$\infty$ (10)	Unordered	Unspecified
Contribute	contribute	See 5.4.3.2	0	$\infty$ (10)	Ordered	Unspecified
Metadata Schema	metadataSchema	None	0	$\infty$ (10)	Unordered	CharacterString
Language	language	None	0	1	Unspecified	CharacterString

### 5.4.3.1 Identifier

Table 7 describes the Identifier element and its direct-descendant subelements.

**Table 7—The Identifier element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Identifier	identifier	Catalog Entry	0	$\infty$ (10)	Unordered	Unspecified
Catalog	catalog	None	0	1	Unspecified	CharacterString
Entry	entry	None	0	1	Unspecified	CharacterString

### 5.4.3.2 Contribute

Table 8 describes the Contribute element and its direct-descendant subelements.

**Table 8—The Contribute element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Contribute	contribute	Role Entity Date	0	$\infty$ (10)	Ordered	Unspecified
Role	role	See 5.5.5	0	1	Unspecified	Vocabulary
Entity	entity	None	0	$\infty$ (10)	Ordered	CharacterString
Date	date	See 5.5.5	0	1	Unspecified	DateTime

#### 5.4.3.2.1 Role

If the value space for the Source subelement of the Role element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Role element shall come from the following list of tokens:

- creator
- validator

#### 5.4.3.2.2 Entity

The value held by the Entity element shall be a character string literal that is the canonical lexical representation of a valid vCard as defined in IETF RFC 2426:1998.

NOTE—IETF RFC 2426:1998 does not rely on XML syntax to express the internal structure of a valid vCard. At the time of publication of this Standard, no *de facto* standard W3C XML Schema definition language binding for the vCard specification existed.

### 5.4.3.3 Metadata Schema

For those LOM XML instances that are *strictly conforming* to this Standard, if the Metadata Schema element (`metadataSchema`) is present in a LOM XML instance, the element shall contain the value `LOMv1.0`, (i.e., `<metadataSchema>LOMv1.0</metadataSchema>`).

For those LOM XML instances that are *conforming* but not *strictly conforming* (i.e., contain extended data-model elements), if the Metadata Schema element (`metadataSchema`) is present in a LOM XML instance, the element shall contain the value `LOMv1.0`, and additional occurrences of `metadataSchema` should list all other metadata schemas or authoritative specifications used to create the LOM XML instance.

### 5.4.3.4 Language

The Language element shall be a character string where the value of the character string shall have the format and value space described in 5.5.4.1.

## 5.4.4 Technical

Table 9 describes the Technical element and its direct-descendant subelements.

**Table 9—The Technical element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Technical	<code>technical</code>	Format Size Location Requirement Installation Remarks Other Platform Requirements Duration	0	1	Unspecified	Unspecified
Format	<code>format</code>	None	0	$\infty$ (40)	Unordered	CharacterString
Size	<code>size</code>	None	0	1	Unspecified	CharacterString
Location	<code>location</code>	None	0	$\infty$ (10)	Ordered	CharacterString

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Requirement	requirement	See 5.4.4.3	0	$\infty$ (40)	Unordered	Unspecified
Installation Remarks	installationRemarks	See 5.5.4	0	1	Unspecified	LangString
Other Platform Requirements	otherPlatformRequirements	See 5.5.4	0	1	Unspecified	LangString
Duration	duration	See 5.5.3	0	1	Unspecified	Duration

#### 5.4.4.1 Format

The value held by the Format element shall be a character string where the value of the character string shall be one of the following:

- A literal that is the canonical lexical representation of a Multipurpose Internet Mail Extension (MIME) type value from RFC 2048; or
- The token `non-digital`.

#### 5.4.4.2 Size

The value held by the Size element shall be a combination of the digits "0" . . . "9". The Size value in a LOM XML instance shall be a valid non-negative integer as defined by the XML Schema derived data type `nonNegativeInteger` (see XML Schema, Part 2).

#### 5.4.4.3 Requirement

Table 10 describes the Requirement element and its direct-descendant subelement.

**Table 10—The Requirement element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Requirement	requirement	OrComposite	0	$\infty$ (40)	Unordered	Unspecified
OrComposite	orComposite	See 5.4.4.3.1	0	$\infty$ (40)	Unordered	Unspecified

##### 5.4.4.3.1 OrComposite

Table 11 describes the OrComposite element and its direct-descendant subelements.

**Table 11—The OrComposite element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
OrComposite	orComposite	Type Name Minimum Version Maximum Version	0	$\infty$ (40)	Unordered	Unspecified
Type	type	See 5.5.5	0	1	Unspecified	Vocabulary
Name	name	See 5.5.5	0	1	Unspecified	Vocabulary
Minimum Version	minimumVersion	None	0	1	Unspecified	CharacterString
Maximum Version	maximumVersion	None	0	1	Unspecified	CharacterString

**5.4.4.3.1.1 Type**

The Type and Name elements shall be present as a pair. If one element is present in a LOM XML instance, the other shall be present, also.

If the value space for the Source subelement of the Type element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Type element shall come from the following list of tokens:

- operating system
- browser

**5.4.4.3.1.2 Name**

The Type and Name elements shall be present as a pair. If one element is present in a LOM XML instance, the other shall be present, also.

If the value space for Source subelement of the Name element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), and the value of the Value subelement of the Type element is the token `operating system`, then the valid values for the Value subelement of the Name element shall come from the following list of tokens:

- pc-dos
- ms-windows
- macos
- unix
- multi-os
- none

If the value space for the Source subelement of the Name element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), and the value of the Value subelement of the

Type element is the token `browser`, then the valid values for the Value subelement of the Name element shall come from the following list of tokens:

- any
- netscape communicator
- ms-internet explorer
- opera
- amaya

## 5.4.5 Educational

Table 12 describes the Educational element and its direct-descendant subelements.

**Table 12—The Educational element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Educational	<code>educational</code>	Interactivity Type Learning Resource Type Interactivity Level Semantic Density Intended End User Role Context Typical Age Range Difficulty Typical Learning Time Description Language	0	$\infty$ (100)	Unordered	Unspecified
Interactivity Type	<code>interactivityType</code>	See 5.5.5	0	1	Unspecified	Vocabulary
Learning Resource Type	<code>learningResourceType</code>	See 5.5.5	0	$\infty$ (10)	Ordered	Vocabulary
Interactivity Level	<code>interactivityLevel</code>	See 5.5.5	0	1	Unspecified	Vocabulary
Semantic Density	<code>semanticDensity</code>	See 5.5.5	0	1	Unspecified	Vocabulary
Intended End User Role	<code>intendedEndUserRole</code>	See 5.5.5	0	$\infty$ (10)	Ordered	Vocabulary
Context	<code>context</code>	See 5.5.5	0	$\infty$ (10)	Unordered	Vocabulary

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Typical Age Range	typicalAgeRange	See 5.5.4	0	$\infty$ (5)	Unordered	LangString
Difficulty	difficulty	See 5.5.5	0	1	Unspecified	Vocabulary
Typical Learning Time	typicalLearningTime	See 5.5.3	0	1	Unspecified	Duration
Description	description	See 5.5.4	0	$\infty$ (10)	Unordered	LangString
Language	language	None	0	$\infty$ (10)	Unordered	CharacterString

#### 5.4.5.1 Interactivity Type

If the value space for the Source subelement of the Interactivity Type element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Interactivity Type element shall come from the following list of tokens:

- active
- expositive
- mixed

#### 5.4.5.2 Learning Resource Type

If the value space for the Source subelement of the Learning Resource Type element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Learning Resource Type element shall come from the following list of tokens:

- exercise
- simulation
- questionnaire
- diagram
- figure
- graph
- index
- slide
- table
- narrative text
- exam
- experiment

- problem statement
- self assessment
- lecture

#### **5.4.5.3 Interactivity Level**

If the value space for the Source subelement of the Interactivity Level element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Interactivity Level element shall come from the following list of tokens:

- very low
- low
- medium
- high
- very high

#### **5.4.5.4 Semantic Density**

If the value space for the Source subelement of the Semantic Density element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Semantic Density element shall come from the following list of tokens:

- very low
- low
- medium
- high
- very high

#### **5.4.5.5 Intended End User Role**

If the value space for the Source subelement of the Intended End User Role element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Intended End User Role element shall come from the following list of tokens:

- teacher
- author
- learner
- manager

### 5.4.5.6 Context

If the value space for the Source subelement of the Context element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Context element shall come from the following list of tokens:

- school
- higher education
- training
- other

### 5.4.5.7 Difficulty

If the value space for the Source subelement of the Difficulty element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Difficulty element shall come from the following list of tokens:

- very easy
- easy
- medium
- difficult
- very difficult

### 5.4.5.8 Language

The Language element shall be a character string where the value of the character string shall have the format and value space described in 5.5.4.1.

## 5.4.6 Rights

Table 13 describes the Rights element and its direct-descendant subelements.

**Table 13—The Rights element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Rights	rights	Cost Copyright And Other Restrictions Description	0	1	Unspecified	Unspecified
Cost	cost	See 5.5.5	0	1	Unspecified	Vocabulary

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Copyright And Other Restrictions	copyrightAndOtherRestrictions	See 5.5.5	0	1	Unspecified	Vocabulary
Description	description	See 5.5.4	0	1	Unspecified	LangString

#### 5.4.6.1 Cost

If the value space for the Source subelement of the Cost element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Cost element shall come from the following list of tokens:

- yes
- no

#### 5.4.6.2 Copyright And Other Restrictions

If the value space for the Source subelement of the Copyright And Other Restrictions element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Copyright And Other Restrictions element shall come from the following list of tokens:

- yes
- no

#### 5.4.7 Relation

Table 14 describes the Relation element and its direct-descendant subelements.

**Table 14—The Relation element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Relation	relation	Kind Resource	0	$\infty$ (100)	Unordered	Unspecified
Kind	kind	See 5.5.5	0	1	Unspecified	Vocabulary
Resource	resource	See 5.4.7.2	0	1	Unspecified	Unspecified

#### 5.4.7.1 Kind

If the value space for the Source subelement of the Kind element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Kind element shall come from the following list of tokens:

- ispartof
- haspart
- isversionof
- hasversion
- isformatof
- hasformat
- references
- isreferencedby
- isbasedon
- isbasisfor
- requires
- isrequiredby

### 5.4.7.2 Resource

Table 15 describes the Resource element and its direct-descendant subelements.

**Table 15—The Resource element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Resource	resource	Identifier Description	0	1	Unspecified	Unspecified
Identifier	identifier	See 5.4.7.2.1	0	$\infty$ (10)	Unordered	Unspecified
Description	description	See 5.5.4	0	$\infty$ (10)	Unordered	LangString

#### 5.4.7.2.1 Identifier

Table 16 describes the Identifier element and its direct-descendant subelements.

**Table 16—The Identifier element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Identifier	identifier	Catalog Entry	0	$\infty$ (10)	Unordered	Unspecified
Catalog	catalog	None	0	1	Unspecified	CharacterString
Entry	entry	None	0	1	Unspecified	CharacterString

## 5.4.8 Annotation

Table 17 describes the Annotation element and its direct-descendant subelements.

**Table 17—The Annotation element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Annotation	annotation	Entity Date Description	0	$\infty$ (30)	Unordered	Unspecified
Entity	entity	None	0	1	Unspecified	CharacterString
Date	date	See 5.5.2	0	1	Unspecified	DateTime
Description	description	See 5.5.4	0	1	Unspecified	LangString

### 5.4.8.1 Entity

The value held by the Entity element shall be a character string literal that is the canonical lexical representation of a valid vCard as defined in IETF RFC 2426:1998.

NOTE—IETF RFC 2426:1998 does not rely on XML syntax to express the internal structure of a valid vCard. At the time of publication of this Standard, no *de facto* standard W3C XML Schema definition language binding for the vCard specification existed.

## 5.4.9 Classification

Table 18 describes the Classification element and its direct-descendant subelements.

**Table 18—The Classification element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Classification	classification	Purpose Taxon Path Description Keyword	0	$\infty$ (40)	Unordered	Unspecified
Purpose	purpose	None	0	1	Unspecified	Vocabulary
Taxon Path	taxonPath	See 5.4.9.2	0	$\infty$ (15)	Unordered	Unspecified
Description	description	See 5.5.4	0	1	Unspecified	LangString
Keyword	keyword	See 5.5.4	0	$\infty$ (40)	Ordered	LangString

### 5.4.9.1 Purpose

If the value space for the Source subelement of the Purpose element is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the valid values for the Value subelement of the Purpose element shall come from the following list of tokens:

- discipline
- idea
- prerequisite
- educational objective
- accessibility restrictions
- educational level
- skill level
- security level
- competency

### 5.4.9.2 Taxon Path

Table 19 describes the Taxon Path element and its direct-descendant subelements.

**Table 19—The Taxon Path element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Taxon Path	taxonPath	Source Taxon	0	$\infty$ (15)	Unordered	Unspecified
Source	source	See 5.5.4	0	1	Unspecified	LangString
Taxon	taxon	See 5.4.9.2.1	0	$\infty$ (15)	Ordered	Unspecified

#### 5.4.9.2.1 Taxon

Table 20 describes the Taxon element and its direct-descendant subelements.

**Table 20—The Taxon element**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Taxon	taxon	Id Entry	0	$\infty$ (15)	Ordered	Unspecified
Id	id	None	0	1	Unspecified	CharacterString
Entry	entry	See 5.5.4	0	1	Unspecified	LangString

## 5.5 Common data types and elements

Data types that are used in the LOM XML Schema binding are defined in 5.5.1 – 5.5.5.

### 5.5.1 CharacterString

For those elements defined in this Standard that have a LOM data type of CharacterString, the W3C XML Schema definition language binding shall have an XML Schema primitive data type of `string`. The `string` data type supports the repertoire of ISO/IEC 10646–1. The `string` data type ensures the support of multiple languages including multibyte languages.

NOTE—The XML Schema definition language primitive data type `string` is defined in "XML Schema Part 2."

### 5.5.2 DateTime

The DateTime data type is represented as a set of XML elements. LOM data elements with a data type of DateTime may contain the following subelements:

- DateTime
- Description

If present, the subelements listed above shall be contained by the element for which the DateTime data type is defined. The order of appearance of subelements shall not be significant.

Table 21 describes the DateTime data type and its direct-descendant subelements.

**Table 21—The DateTime data type**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
DateTime	–	DateTime Description	0	1	Unspecified	Unspecified
DateTime	<code>dateTime</code>	None	0	1	Unspecified	CharacterString
Description	<code>description</code>	See 5.5.4	0	1	Unspecified	LangString

NOTE—The DateTime data type is made up of two elements, DateTime and Description, that are direct descendents of all aggregate elements that are defined to be of type DateTime. For example, DateTime and Description are the direct descendents of the aggregate element Date (see 5.4.2.2).

#### 5.5.2.1 DateTime

The DateTime element is a pattern that shall be defined according to the following constraints.

YYYY[-MM[-DD[Thh[:mm[:ss[.s[TZD]]]]]]] where:

- YYYY: The 4-digit year ( $\geq 0001$ )
- MM: The 2-digit month (01 through 12 where 01=January, etc.)
- DD: The 2-digit day of month (01 through 31, depending on the values of month and year)
- hh: Two digits of hour (00 through 23) (am and pm are NOT allowed)
- mm: Two digits of minute (00 through 59)
- ss: Two digits of second (00 through 59)
- s: One or more digits representing a decimal fraction of a second
- TZD: The time zone designator ("Z" for coordinated universal time [UTC] or +hh, -hh, +hh:mm, or -hh:mm, where hh is two digits of hour and mm is two digits of minute)

If the DateTime element contains a value, at least the four-digit year shall be present. If additional parts of the date and time are included, the character literals "-", "T", ":", and "." are part of the character lexical representation for the DateTime element.

If the time portion is present, but the time zone designator is not present, the time zone is interpreted as being UTC.

NOTES:

1—Because of restrictions placed on the value space, the W3C XML Schema definition language primitive data type `dateTime` could not be used to create conforming LOM XML instances. Therefore, a regular expression was created to represent all valid dates and times supported by IEEE 1484.12.1–2002.

2—The date portion represents dates in the Common Era (CE), only. The date portion follows the Gregorian calendar for dates from October 15, 1582, forward, and the Julian calendar for dates prior to October 15, 1582, independent of locale. Dates Before Common Era (BCE) and other cases should be represented using the Description element (see 5.5.2).

3—The square bracket meta characters ("[" and "]") indicate optional elements that may appear zero or one time in the character lexical representation of the DateTime element. These meta characters do not appear in the result; only the associated values described appear (e.g., DD is replaced by the corresponding 2-digit value for day of month).

4—The value space is based on ISO 8601:2000 [B3].

### 5.5.3 Duration

The Duration data type is represented as a set of XML elements. LOM data elements with a data type of Duration may contain the following subelements:

- Duration
- Description

If present, the subelements listed above shall be contained by the element for which the Duration data type is defined. The order of appearance of subelements shall not be significant.

Table 22 describes the Duration data type and its direct-descendant subelements.

**Table 22—The Duration data type**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Duration	–	Duration Description	0	1	Unspecified	Unspecified
Duration	duration	None	0	1	Unspecified	CharacterString
Description	description	See 5.5.4	0	1	Unspecified	LangString

NOTE—The Duration data type is made up of two elements, Duration and Description, that are direct descendants of all aggregate elements that are defined to be of type Duration. For example, Duration and Description are the direct descendants of the aggregate element Duration (see 5.4.4).

### 5.5.3.1 Duration

The value space for the Duration element is a pattern that shall be defined according to the following constraints:

$P[yY][mM][dD][T[hH][nM][s[.s]S]]$  where:

- *y*: The number of years (integer,  $\geq 0$ )
- *m*: The number of months (integer,  $\geq 0$ , not restricted, e.g.,  $> 12$  is acceptable)
- *d*: The number of days (integer,  $\geq 0$ , not restricted, e.g.,  $> 31$  is acceptable)
- *h*: The number of hours (integer,  $\geq 0$ , not restricted, e.g.,  $> 23$  is acceptable)
- *n*: The number of minutes (integer,  $\geq 0$ , not restricted, e.g.,  $> 59$  is acceptable)
- *s*: The number of seconds or fraction of seconds (integer,  $\geq 0$ , not restricted, e.g.,  $> 59$  is acceptable)

The character literal designators "P", "Y", "M", "D", "T", "H", "M", and "S" shall appear if the corresponding nonzero value is present.

If the Duration element contains a value, the designator "P" shall be present. If the value of years, months, days, hours, minutes or seconds is zero, the value and corresponding designation (e.g., "M") may be omitted, but at least one designator with a positive integer value shall

be present in addition to the designator "P". The designator "T" shall be omitted if all of the time (hours, minutes, and seconds) is zero.

Negative durations are not supported.

NOTES:

1—Because of the restrictions placed on the value space, the W3C XML Schema definition language primitive data type `dateTime` could not be used to create conforming LOM XML instances. A regular expression was built to represent all valid date/times supported by IEEE 1484.12.1–2002.

2—The value is designated in the Gregorian calendar.

3—The ordering of durations may be indeterminate (e.g., 1 month may be 28, 29, 30, or 31 days).

4—The square bracket meta characters ("[" and "]") indicate optional elements that may appear zero or one time in the character lexical representation of the Duration. These meta characters do not appear in the result; only the associated values described appear (e.g., `dD` is replaced by the corresponding value for the number of days in the duration and is followed by the designator "D").

5—The value space is based on ISO 8601:2000 [B3].

### 5.5.3.2 Description

The Description data type is represented as a `LangString` data type (see 5.5.4).

### 5.5.4 LangString

The `LangString` data type is represented as a set of XML elements. LOM data elements with a data type of `LangString` may contain the following subelements:

- String
- Language

If present, the subelements listed above shall be contained by the element for which the `LangString` data type is defined. The order of appearance of subelements shall not be significant. Within a LOM XML instance, the LOM data element `Language` shall be specified as an attribute of LOM data elements that are of type `LangString`.

Table 23 describes the `LangString` data type and its direct-descendant subelements.

**Table 23—The LangString data type**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
LangString	–	String Language	0	$\infty$ (10)	Unordered	Unspecified
String	<code>string</code>	None	0	1	Unspecified	CharacterString
Language	<code>language</code>	None	0	1	Unspecified	CharacterString

NOTE—The LangString data type is made up of two elements, String and Language, that are direct descendents of all aggregate elements that are defined to be of type LangString. For example, String and Language are the direct descendents of the aggregate element Title (see 5.4.1).

### 5.5.4.1 Language

The Language element shall be a character string consisting of a required language code followed by multiple, optional, hyphen-prefixed subcodes.

The following constraints apply to the language code part of the character string:

- 2-letter codes are defined in ISO 639–1.
- 3-letter codes are defined in ISO 639–2.
- The 1-letter code "i" is reserved and uses as a prefix for registrations defined by the Internet Assigned Numbers Authority (IANA).
- The 1-letter code "x" is reserved and used as a prefix for private use.

The following constraints apply to the first subcode part of the character string:

- 2-letter subcodes are ISO 3166–1 alpha-2 country codes.
- Subcodes from 3 to 8 characters in length are registered with IANA.

Constraints for additional subcodes are unspecified.

The value held by the character string shall be a valid language code as defined by the XML Schema derived data type `language` (see XML Schema, Part 2).

ISO 639–2 specifies two code sets, one for bibliographic applications (ISO 639–2/B) and one for terminology applications (ISO 639–2/T). Either code set may be used.

NOTES:

1—The language code is normally given in lower case and the subcodes (if any) in upper case. However, the values are case insensitive.

2—The XML Schema derived data type `language` does not enforce all the constraints on the language code that are described above.

## Examples

```
"en-GB"
"de"
"fr-CA"
"it"
"i-bnn" (IANA Bunun)
```

### 5.5.5 Vocabulary

The Vocabulary data type is represented as a set of XML elements. LOM data elements with a data type of Vocabulary may contain the following subelements:

- Source
- Value

If present, the subelements listed above shall be contained by the element for which the Vocabulary data type is defined. The order of appearance of subelements shall not be significant.

Table 24 describes the Vocabulary data type and its direct-descendant subelements.

**Table 24—The Vocabulary data type**

LOM data element	XML name	Subelements	Min	Max	Order	LOM data type
Vocabulary	–	Source Value	Unspecified	Unspecified	Unspecified	Unspecified
Source	source	None	0	1	Unspecified	CharacterString
Value	value	None	0	1	Unspecified	CharacterString

NOTE—The Vocabulary data type is made up of two elements, Source and Value, that are direct descendents of all aggregate elements that are defined to be of type Vocabulary. For example, Source and Value are the direct descendents of the aggregate element Aggregation Level (see 5.4.1.4).

#### 5.5.5.1 Value

The value held by the character string shall be a valid token as defined by the XML Schema derived data type `token` (see XML Schema, Part 2). Valid values for the tokens are defined for those LOM data elements that are of type Vocabulary.

NOTE—The XML Schema derived data type `token` supports the repertoire of ISO/IEC 10646–1. The `token` data type ensures the support of multiple languages including multibyte languages. Multibyte languages are not used by the LOMv1.0 base schema vocabularies but may be used if the source of a vocabulary is not LOMv1.0.

## Annex A

(informative)

### Bibliography

[B1] IEEE 100™, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition.

[B2] IETF RFC 2425:1998, MIME Content-Type for Directory Information.

[B3] ISO 8601:2000, Data elements and interchange formats – Information interchange – Representation of dates and times.

[B4] ISO/IEC 11404:1996, Information Technology—Programming Languages, their Environments and System Software Interfaces—Language-Independent Datatypes.

[B5] W3C Recommendation (04 February 2004), Extensible Markup Language (XML) 1.1.

[B6] W3C Recommendation (04 February 2004), XML Information Set (Second Edition).

## **Annex B**

(informative)

### **Internet availability of XSD files**

This Standard includes XSD files that are available on the World Wide Web but are not included in the printed version of this Standard. These XSD files are available for direct inclusion in applications and for use as examples for those who prefer to develop their own XSDs.

The XSD files are available at the following URL:

- <http://standards.ieee.org/reading/ieee/downloads/LOM/lomv1.0/>

NOTE—This Standard does not require the use of an XSD to validate a LOM XML instance.

## Annex C

(informative)

### XSD file descriptions

The LOMv1.0 base schema describes a structured collection of Standard data items, including their data types, multiplicities, and container/component relationships, but does not provide a syntax for encoding LOM data that conforms to the LOMv1.0 base schema.

The LOM XML binding described in this Standard defines the XML syntax for encoding LOM data. The binding is a collection of rules describing how to express LOM data in XML syntax. These binding rules are defined in detail in Clause 5 of this Standard.

When building XSDs, implementers can decide when to enforce LOMv1.0 base schema constraints. These constraints can be defined in an XSD with the intent that schema processors will perform validation of the constraints or that validation will be performed by processing the PSVI. Clause 5 defines the criteria to create valid LOM instances; XSD implementers must decide how to develop XSD schema files according to these constraints. Regardless of the implementation, a *conforming* or *strictly conforming* LOM instance has to conform to the rules defined in Clause 5 of this Standard.

This Standard includes informative W3C XML XSDs for implementers to satisfy many of the requirements of IEEE 1484.12.1–2002 (see Annex B) and the practices of its existing adopters. The XSDs are intended to be used by XML processing tools that support the W3C XML Schema definition language (see XML Schema, Parts 1 and 2). The LOMv1.0 base schema places requirements on a LOM XML instance, some of which cannot be expressed in W3C XML Schema definition language. These requirements are defined in Clause 5 of this Standard and in the LOMv1.0 base schema.

The goal is to provide alternative XSDs, each of which supports the LOM XML Schema binding by making certain choices about which binding constraints to describe in the XSDs. Constraints defined in an XSD should be enforced by tools that support XML Schema validation. Constraints not expressed in an XSD are enforced by other means, such as processing the PSVI. No matter where constraint enforcement is performed, a *conforming* or *strictly conforming* LOM XML instance has to conform to all constraints defined in Clause 5 of this Standard and in the LOMv1.0 base schema.

A predefined, composite XSD (see C.1.1) is provided that can be tailored in multiple ways for particular preferences about LOM data element ordering, vocabulary values checking, and extension use. Three additional predefined composite XSDs are provided that enforce different validation constraints (see C.1.2 – C.1.3). The four composite XSDs draw from several sets of component XSDs (see C.2).

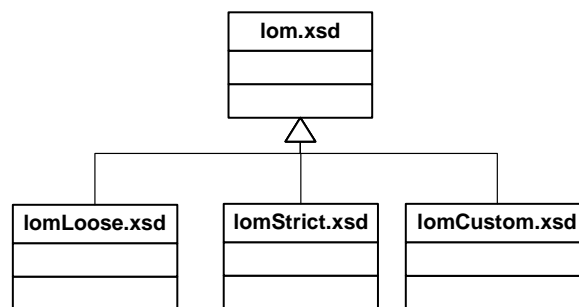
The need for alternative XSDs was motivated by the variety of ways in which application developers need to use the XSDs. The approach of providing alternative XSDs addresses the needs of the various adopting communities to produce *conforming* or *strictly conforming* LOM XML instances consistent with the communities' processing and usage requirements.

The presence or absence of the enforcement of any part of the LOM XML Schema binding by a particular XSD in no way relieves a particular LOM XML instance from satisfying all of the LOM XML Schema binding constraints in order to be considered a *conforming* or *strictly conforming* LOM XML instance. The presence of alternative XSDs merely provides developers choices in determining how to enforce the LOM XML Schema binding constraints described in this Standard.

## C.1 Composite XSDs

The `lom.xsd` composite XSD acts as a template after which the other composite XSDs (`lomLoose.xsd`, `lomStrict.xsd`, and `lomCustom.xsd`) have been modeled. Each of these XSDs is structured from component XSDs described in C.2. By default the `lom.xsd` composite XSD encodes a particular set of binding constraints in the W3C XML Schema definition language such that a LOM XML instance will *strictly conform* to IEEE 1484–12.1–2002.

As shown in Figure C.1, the `lom.xsd` composite XSD is an abstract composite XSD from which three additional composite XSDs (`lomLoose.xsd`, `lomStrict.xsd`, and `lomCustom.xsd`) have been built. These composite XSDs model certain validation approaches based on the selection of a combination of multiple, component XSDs. (See C.1.1 – C.1.3 for more information about these composite XSDs.)



**Figure C.1—The relationship of the composite XSDs.**

NOTE—The names of XSDs in C.1.1 – C.1.3, `lomLoose.xsd`, `lomStrict.xsd`, and `lomCustom.xsd`, refer to files that are accessed through URLs (see Annex B).

### C.1.1 lomStrict.xsd

The `lomStrict.xsd` composite XSD encodes a particular set of binding constraints in the W3C XML Schema definition language such that a LOM XML instance will *strictly conform* to IEEE 1484–12.1–2002. The `lomStrict.xsd` composite XSD

- Limits vocabularies (source/value pairs) to those defined in Clause 5 of this Standard;
- Expresses the uniqueness constraints defined in Clause 5 of this Standard; and
- Does not permit extensions to the LOMv1.0 base schema (see 5.1.3).

### C.1.2 lomCustom.xsd

The `lomCustom.xsd` composite XSD encodes a particular set of binding constraints in the W3C XML Schema definition language such that a LOM XML instance will *conform* to IEEE 1484–12.1–2002. The `lomCustom.xsd` composite XSD

- Defines a mechanism to describe vocabularies (source/value pairs) in addition to those defined in Clause 5 of this Standard. An XML schema processor will check that vocabulary values in a LOM XML instance conform to either the vocabularies defined in Clause 5 or those defined in an extended vocabulary;
- Expresses the uniqueness constraints defined in Clause 5 of this Standard; and
- Defines a mechanism to describe extensions to the LOMv1.0 base schema (see 5.1.3). Extensions are defined in a set of supporting XSDs. An XML schema processor will check that the extensions used in a LOM XML instance conform to constraints defined in the supporting XSDs that include extensions.

### C.1.3 lomLoose.xsd

The `lomLoose.xsd` composite XSD encodes a particular set of binding constraints in the W3C XML Schema definition language such that a LOM XML instance may not either *conform* or *strictly conform* to IEEE 1484–12.1–2002. The `lomLoose.xsd` composite XSD

- Does not provide validation to any defined vocabulary values (i.e., values defined in Clause 5 of this Standard or values defined in an extended vocabulary). An XML schema processor will verify that the vocabulary values are valid tokens, only;
- Does not express the uniqueness constraints defined in Clause 5 of this Standard; and
- Defines a mechanism to describe extensions to the LOMv1.0 base schema (see 5.1.3). Extensions are defined in a set of supporting XSDs. An XML schema processor will check that the extensions used in a LOM XML instance conform to constraints defined in the supporting XSDs that include extensions.

NOTE—The absence of the validation of vocabulary values does not relieve a particular LOM XML instance from satisfying the vocabulary requirements defined in Clause 5 of this Standard. Applications that use the `vocab/loose.xsd` component XSD should enforce those vocabulary requirements by other means.

## C.2 Component XSDs

A component XSD represents a constituent of a composite XSD. The following component XSDs are used by the composite XSDs defined in C.1:

- `common/anyElement.xsd`: Defines the base XML Schema model groups for elements and attributes used for extended data elements and XML attributes.
- `common/dataTypes.xsd`: Defines global data type declarations for data types defined in Clause 5 of this Standard.
- `common/elementNames.xsd`: Defines global element declarations for each of the data elements defined in Clause 5 of this Standard. This component XSD is used to check for the uniqueness of data elements declared to be unique within their aggregate elements by the presence of the `uniqueElementName` attribute. The XML Schema constraint `unique` is used to enforce uniqueness constraints.
- `common/elementTypes.xsd`: Defines global data type declarations for data elements defined in Clause 5 of this Standard. This component XSD defines the aggregation relationships among the LOM data elements. These aggregation relationships enforce the LOMv1.0 base schema requirement that elements can be present in a LOM XML instance as elements of the aggregate element to which they belong, only.
- `common/rootElement.xsd`: Defines the element name declaration for the element that contains all other LOM data elements for a LOM XML instance.
- `common/vocabTypes.xsd`: Defines global data type declarations for those LOM data elements that have values taken from a vocabulary data type defined in Clause 5 of this Standard.
- `common/vocabValues.xsd`: Defines the standard vocabulary value declarations defined in Clause 5 of this Standard. These declarations are used in conjunction with both `vocab/custom.xsd` and `vocab/strict.xsd`.
- `extend/custom.xsd`: Defines the XML Schema model groups `customElements` and `customAttributes` to support validation of extended data elements and attributes. The model groups in this component XSD reference the model groups defined in `common/anyElement.xsd`. This component XSD should be used if extensions are to be supported in LOM XML instances (see 5.1.3). *Note:* Assuming proper processing of the PSVI, by using the `extend/custom.xsd` component XSD LOM XML instances that use extensions will *conform* but not *strictly conform* to this Standard.

- `extend/strict.xsd`: Defines the XML Schema model groups `customElements` and `customAttributes`. The model groups are defined as empty model groups and are used with other component XSDs to support the validation of *strictly conforming* LOM XML instances. This component XSD should be used if extensions are not to be supported in LOM XML instances. *Note*: Assuming proper processing of the PSVI, by using the `extend/strict.xsd` component XSD and, therefore, not supporting extensions, LOM XML instances will *strictly conform* to this Standard.
- `unique/loose.xsd`: Defines XML Schema model group declarations for LOM data elements to support the schema-based validation of uniqueness constraints within a LOM XML instance where the exact set of attributes associated with each element has to be as specified by the LOM XML Schema binding (i.e., when extra attributes to enforce uniqueness have to be avoided). This component XSD is used to relax the enforcement of uniqueness constraints to avoid the introduction of the XML attribute `uniqueElementName` (see E.3.3). *Note*: The absence of the enforcement of uniqueness constraints does not relieve a LOM XML instance from satisfying the uniqueness constraints defined in Clause 5 of this Standard. Applications that use of the `unique/loose.xsd` component XSD have to enforce those uniqueness constraints by other means.
- `unique/strict.xsd`: Defines XML Schema model group declarations for LOM data elements defined in Clause 5 of this Standard to support schema-based validation of the uniqueness constraints within a LOM XML instance by introducing the attribute `uniqueElementName` for each LOM data element that appears with a multiplicity of at most one (see E.3.3). *Note*: For most applications, enforcing uniqueness constraints using the `unique/strict.xsd` component XSD is desirable. Although adding the attribute `uniqueElementName` is undesirable, it is unlikely to cause problems.
- `vocab/custom.xsd`: Enforces that vocabulary values are only those specified in Clause 5 of this Standard and in an extended vocabulary. This component XSD enforces adherence to the combined use of standard plus extended vocabulary values by checking that both sources and values are taken from either a token set defined in Clause 5 or from an extended vocabulary. *Note*: Strict adherence to the vocabulary values defined in Clause 5 of this Standard and extended vocabulary values using the `vocab/custom.xsd` component XSD is desirable under some circumstances, but may complicate the schema validation process.
- `vocab/loose.xsd`: Enforces that vocabulary sources and values are character strings. This component XSD relaxes the validation constraints by allowing both sources and values to be arbitrary character strings. LOM XML instances that use the `vocab/loose.xsd` component XSD may be non-conforming. Applications that require validation of the source/value pairs will have to process the PSVI. *Note*: The absence of the enforcement of vocabulary value validation by the `vocab/loose.xsd` component XSD does not relieve a particular LOM XML instance from satisfying the vocabulary constraints described in Clause 5 of this

Standard. Applications that use the `vocab/loose.xsd` component XSD have to enforce those vocabulary constraints by other means.

- `vocab/strict.xsd`: Enforces that vocabulary values are only those specified in Clause 5 of this Standard. This component XSD does not allow extended vocabulary values from other sources. This component XSD supports the validation of *strictly conforming* vocabulary values by checking that both sources and values are from a token set defined in Clause 5.

## Annex D

(informative)

### Enabling extended data elements and attributes

The LOMv1.0 base schema anticipates the need to introduce extensions not included in the standard collection of elements (see 5.1.3). These extensions have to meet the following constraints:

- Extensions to the LOMv1.0 base schema retain the value spaces and data types of LOM data elements from the LOMv1.0 base schema; and
- Extensions do not define data types or value spaces for aggregate elements in the LOMv1.0 base schema.

These requirements imply that if an aggregate element contains extensions, the extensions do not redefine the intended meaning (i.e., value space and data type) of the aggregate element. To meet these requirements, the informative schemas (see Annex B) include an any element in the content model for aggregate elements (`<xs:any namespace="##other" processContents="lax"/>`).

The `processContents="lax"` declaration instructs an XML processor to attempt to validate the element's content. This declaration allows elements from namespaces other than the LOMv1.0 base schema namespace to be included and their contents to be validated, if schema information for the elements can be found. If such information is not available, the XML Schema processor will validate any well-formed LOM XML instance.

NOTES:

1—To ensure that conforming LOM XML instances are created, an organization that develops extensions also should provide an XSD that defines the validation constraints for the extensions.

2—Organizations providing extensions may choose to define namespaces for these extensions. If an organization defines a namespace, then the XSDs provided by this Standard may have to be altered to support the namespace.

#### D.1 Enabling extended data elements

LOMv1.0 base schema data elements may be extended. These extensions are new XML elements defined in a namespace other than the namespaces defined in 5.2. Extended data elements may be added to any LOM data element that is defined as an aggregate element.

To enable extended data elements, the `extend/custom.xsd` component XSD may be used in the construction of the composite XSD. This component XSD defines the XML Schema model group `customElements` that references the `customElements` model group defined

in the `common/anyElement.xsd` component XSD. (See the `lomCustom.xsd` file provided with this Standard, [Annex B].) The inclusion of these two component XSDs by a composite XSD enables the validation of extended data elements.

The relevant component XSDs and how they enable extended data elements are described below.

- `common/anyElement.xsd`: Defines the base XML Schema model group for elements used for extended data elements.
- `common/dataTypes.xsd` and `common/elementTypes.xsd`: For each aggregate `complexType` declaration defined in these component XSDs, an XML Schema model group reference to the `customElements` declaration is defined.
- `common/vocabTypes.xsd`: For each vocabulary `complexType` declaration defined in this component XSD, an XML Schema model group reference to the `customElements` declaration is defined.
- `extend/custom.xsd`: Defines the XML Schema model group `customElements` to support validation of extended data elements. The XML Schema model group in this component XSD references the model group defined in the `common/anyElement.xsd` component XSD.

NOTE—Extensions may be disabled by using the `extend/strict.xsd` composite XSD.

## D.2 Enabling extended attributes

LOMv1.0 base schema data elements may be extended with XML attributes defined in a namespace other than the namespaces defined in 5.2. Extended XML attributes may be added to any LOM data element.

To enable extended attributes, the `extend/custom.xsd` component XSD may be used in the construction of the composite XSD. This component XSD defines the XML Schema model group `customAttributes`. This model group defines the ability to add any attribute from a namespace other than the namespaces defined in 5.2. The inclusion of this component XSD by a composite XSD enables the validation of extended attributes.

The relevant component XSDs and how they enable extended attributes are described below.

- `common/anyElement.xsd`: Defines the base XML Schema model group for elements used for extended data elements.
- `common/dataTypes.xsd` and `common/elementTypes.xsd`: For each aggregate `complexType` declaration defined in these component XSDs, an XML Schema model group reference to the `customAttributes` declaration is defined.
- `common/vocabTypes.xsd`: For each vocabulary and value `complexType` declaration defined in this component XSD, an XML Schema `attributeGroup`

reference to the `customAttributes` XML Schema model group declaration is defined.

- `extend/custom.xsd`: Defines the XML Schema model group `customAttributes` to support validation of extended attributes.

NOTE—Extensions may be disabled by using the `extend/strict.xsd` composite XSD.

## Annex E

(informative)

### XSD implementation choices

Major implementation choices contained in the example XSDs (see Annex B) are discussed in E.1 – E.6.

#### E.1 Data types

The `common/dataTypes.xsd` component XSD defines a collection of global W3C XML Schema definition language type declarations for the data types used to constrain values for the data elements defined in Clause 5 of this Standard. The type declarations are provided so that the logical data types in the LOMv1.0 base schema are defined in terms of their underlying XML Schema data type once, only. This modularization of the schema definition provides a means to change the underlying schema data type, if necessary, without making any other changes to the schema definition.

The following global W3C XML Schema definition language types are defined :

- `CharacterString`: an alias for the XML Schema primitive data type `string`.
- `LanguageId`: an alias for the XML Schema derived data type `language`.
- `VCard`: an alias for `CharacterString`.
- `MimeType`: an alias for `CharacterString`.
- `Size`: an alias for the XML Schema derived data type `nonNegativeInteger`.
- `LanguageString`: a sequence of zero or more `string` elements with an optionally defined `language` attribute. The `language` attribute describes the language of the value held by the `string` element.
- `DateTime`: optional subelements `dateTime` and `description`.
- `Duration`: optional subelements `duration` and `description`.

`LanguageId` is used to constrain the values of the various `language` elements.

The `LanguageString` type allows each `string` element to contain the optional attribute `language` of type `LanguageId`, which is defined as the XML Schema derived data type `language`.

The `DateTime` type has the optional subelements `dateTime` of type `CharacterString` and `description` of type `LanguageString`. The `CharacterString` type for the

`dateTime` subelement contains a restricted pattern of characters. The pattern, defined using a regular expression in W3C XML Schema definition language, is

```
([0-9]{3}[1-9] | [0-9]{2}[1-9][0-9] | [0-9][1-9][0-9]{2} | [1-9]
[0-9]{3})(\-(0[1-9] | 1[0-2])\-(0[1-9] | [1-2][0-9] | 3[0-1]))(T
([0-1][0-9] | 2[0-3])(:[0-5][0-9])(:[0-5][0-9](\.[0-9]{1,}(Z |
((\+|\-)([0-1][0-9] | 2[0-3]):[0-5][0-9])))?)?)?)?)?
```

The restricted character string pattern is used instead of the XML Schema primitive data type because of the restrictions placed on the value space defined in IEEE 1484.12.1–2002. These restrictions cannot be enforced by the XML Schema primitive data type `dateTime`.

The `Duration` data type has the optional subelements `duration`, of type `CharacterString`, and `description`, of type `LanguageString`. The `CharacterString` type for the `duration` subelement contains a restricted pattern of characters. The pattern, defined using a regular expression in W3C XML Schema definition language, is

```
P([0-9]{1,}Y){0,1}([0-9]{1,}M){0,1}([0-9]{1,}D){0,1}(T([0-9]{1,}
H){0,1}([0-9]{1,}M){0,1}([0-9]{1,}(\.[0-9]{1,}))){0,1}S){0,1}
{0,1}
```

The restricted character string pattern is used instead of the XML Schema primitive data type because of the restrictions placed on the value space defined in IEEE 1484.12.1–2002. These restrictions cannot be enforced by the XML Schema primitive data type `duration`.

## E.2 Elements

The `common/elementNames.xsd` component XSD defines a collection of global W3C XML Schema definition language data type declarations for the LOM data elements. The component XSD contains 78 type declarations, one for each LOM data element. Duplicate type declarations for elements that share element names are included within XML comments for completeness. Those elements included within XML comments should be left within XML comments.

The global schema type declarations are provided so that the composite schema is defined in terms of a collection of underlying types, rather than being defined directly in terms of XML Schema constructs. This modularization of the schema definition provides a means to change the definition of the elements without making any other changes to the schema definition.

For each LOM data element, an XML element name is assigned by the convention of using camel-case capitalization of the full name of the LOM data element with an initial lower-case letter (*lowerCamelCase*) (see 5.4 and 5.5).

Each LOM data element is given a global W3C XML Schema definition language element declaration, type declaration, and attribute-group declaration. The XML names defined in Clause 5 are used for the names of each of these declarations.

Typically, the W3C XML Schema definition language type declaration for an element will provide the content model for the element, referring to the global element declarations for subelements as needed and including the attributes from the W3C XML Schema definition language attribute group. LOMv1.0 base schema aggregate elements will have declarations that are represented by complex types. LOMv1.0 base schema simple elements will have declarations that are represented by simple types.

## E.3 Aggregates

The selection of the W3C XML Schema definition language structure for aggregate elements involves satisfying three competing requirements that arise either directly or indirectly from the LOMv1.0 base schema. These are

- a) An ordering restriction is not defined in IEEE 1484.12.1–2002. Each element should allow arbitrary ordering of its subelements.
- b) Each element should enforce multiplicity constraints defined in IEEE 1484.12.1–2002 on its subelements.
- c) The aggregation structure should be preserved without the introduction of additional container elements (i.e., elements that do not map directly to those defined in IEEE 1484.12.1–2002, but are introduced to meet the first two requirements).

Requirements (a) and (b) above introduce complexity in expressing the content model for aggregate elements. Requirement (c) above arises from the implied constraint that the content model should match the LOMv1.0 base schema as closely as possible. The W3C XML Schema definition language does not provide a way to satisfy requirements (a) and (b) without the introduction of additional container elements (c).

Four alternatives for the selection of the content model for aggregate elements could be implemented. These alternatives are presented in E.3.1 – E.3.4. These four alternatives provide options on the validation approaches taken by XML Schema processors.

NOTE—In E.3.1 – E.3.5, "a", "b", and "c" in parentheses or square brackets refer to the requirements listed above.

### E.3.1 Using the XML Schema element sequence

The use of the XML Schema element *sequence* provides a way to meet the requirements of multiplicity (b) and avoiding additional container elements (c) but does not provide arbitrary ordering (a). If subelements are required to appear in a specific order, the XSD can easily enforce the multiplicity constraints without introducing additional container elements. Figure E.1 shows how the *sequence* element can be used to define the LOMv1.0 base schema.

```
<xs:complexType name="lom">
  <xs:sequence>
    <xs:element minOccurs="0" ref="general"/>
    <xs:element minOccurs="0" ref="lifeCycle"/>
  </xs:sequence>
</xs:complexType>
```

```

<xs:element minOccurs="0" ref="metaMetadata"/>
<xs:element minOccurs="0" ref="technical"/>
<xs:element minOccurs="0" maxOccurs="unbounded"
  ref="educational"/>
<xs:element minOccurs="0" ref="rights"/>
<xs:element minOccurs="0" maxOccurs="unbounded" ref="relation"/>
<xs:element minOccurs="0" maxOccurs="unbounded"
  ref="annotation"/>
<xs:element minOccurs="0" maxOccurs="unbounded"
  ref="classification"/>
</xs:sequence>
</xs:complexType>

```

**Figure E.1—An example using sequence construct**

According to Figure E.1 the `general` element can appear zero or one time (satisfies [b]). The example does not introduce additional container elements (satisfies [c]). However, the general element, if present, has to precede the `lifeCycle` element (does not satisfy [a]).

LOM XML instances created using this approach may be *non-conforming*.

### E.3.2 Using the XML Schema element `all`

The XML Schema element `all` provides a way to meet the requirements of arbitrary ordering (a). Using the `all` element requires the introduction of additional container elements (c) to meet the requirement of multiplicity (b). The `all` element requires the introduction of additional container elements to collect multiple instances of a given element. The LOMv1.0 base schema contains elements that can have multiplicities of more than one (e.g., `Educational`). Figure E.2 illustrates how the `all` element can be used to define the LOMv1.0 base schema.

```

<xs:complexType name="lom">
  <xs:all>
    <xs:element minOccurs="0" ref="general"/>
    <xs:element minOccurs="0" ref="lifeCycle"/>
    <xs:element minOccurs="0" ref="metaMetadata"/>
    <xs:element minOccurs="0" ref="technical"/>
    <xs:element minOccurs="0" ref="educational"/> <!-- container -->
    <xs:element minOccurs="0" ref="rights"/>
    <xs:element minOccurs="0" ref="relations"/> <!-- container -->
    <xs:element minOccurs="0" ref="annotations"/> <!-- container -->
    <xs:element minOccurs="0" ref="classifications"/>
    <!-- container -->
  </xs:all>
</xs:complexType>

```

**Figure E.2—An example using the `all` element**

According to Figure E.2 the `general` element can appear zero or one time (satisfies [b]) anywhere in the sequence of elements (satisfies [a]). However, the example introduces additional container elements, such as the `educational`s element (does not satisfy [c]), where the plural `educational`s indicates zero or more `educational` elements.

LOM XML instances created using this approach may be *non-conforming*.

### E.3.3 Using the XML Schema element choice

The use of the XML Schema element `choice` provides a way to meet the requirements of arbitrary ordering (a) and avoiding additional containers (c) but does not enforce multiplicity constraints (b). By allowing any subelement to appear any number of times, the W3C XML Schema definition language types can easily be constructed to allow arbitrary ordering of the subelements without having to introduce additional container elements. Figure E.3 shows how the `choice` element can be used to define the LOMv1.0 base schema.

```
<xs:complexType name="lom">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="general"/>
    <xs:element ref="lifeCycle"/>
    <xs:element ref="metaMetadata"/>
    <xs:element ref="technical"/>
    <xs:element ref="educational"/>
    <xs:element ref="rights"/>
    <xs:element ref="relation"/>
    <xs:element ref="annotation"/>
    <xs:element ref="classification"/>
  </xs:choice>
</xs:complexType>
```

**Figure E.3—An example using the choice element**

The example in Figure E.3 does not introduce additional container elements (satisfies [c]) and allows arbitrary ordering (satisfies [a]). However, all of the elements may appear any number of times (does not satisfy [b]).

This solution accepts a superset of what the LOMv1.0 base schema allows without introducing additional complexity into a LOM XML instance or a LOM XSD. For applications willing to enforce LOM data element multiplicity constraints on the PSVI, this may represent a sufficient solution.

LOM XML instances created using this approach may be *non-conforming*.

### E.3.4 Using the XML Schema element choice with the XML Schema constraint unique

The use of the XML Schema element `choice` with the XML Schema constraint `unique` provides a way to meet the requirements of arbitrary ordering (a) and avoiding additional containers (c) and enforces multiplicity constraints (b). Using this approach, the multiplicity constraints can be enforced using the XML Schema constraint `unique`. For each element with a multiplicity of at most one in its aggregate element, define an attribute called `uniqueElementName` with a fixed default value equal to the name of the element, and use this attribute as the element key for `unique` constraint. Figure E.4 shows how the `unique` constraint can be used to enforce multiplicity constraints for element uniqueness when the LOMv1.0 base schema defines that an element cannot appear more than one time.

```
<xs:element name="lom" type="lom">
  <xs:unique name="lomUnique">
    <xs:selector xpath="*" />
    <xs:field xpath="@uniqueElementName" />
  </xs:unique>
</xs:element>
```

**Figure E.4—An example using the unique constraint**

One such constraint using the `unique` constraint is needed for each aggregate element that contains subelements with multiplicities of at most one.

The disadvantages of this approach are the introduction of the `uniqueElementName` attribute and the introduction of additional complexity in the XSD.

This solution maximizes the amount of validation that can be done by an XML schema processor. For applications that cannot process the PSVI, the addition of the `uniqueElementName` attribute may be a sufficient solution.

This solution supports the validation of both *conforming* and *strictly conforming* LOM XML instances.

### E.3.5 Summary

Subclauses E.3.1 – E.3.4 defined alternative approaches to aggregate elements. A summary of those alternatives is given below

- `sequence`: Does not satisfy requirement (a), which states that the subelements appear in an arbitrary order. This alternative may not conform to IEEE 1484.12.1-2002.

- `all`: Does not satisfy requirement (c), which preserves the aggregation structure defined in IEEE 1484.12.14–2002. The use of the XML Schema element `all` requires the introduction of arbitrary container elements (e.g., `educational`s). This alternative requires all subelements of the container to be collected under the aggregate element.
- `choice`: Does not enforce uniqueness constraints. All of the elements may appear any number of times. This alternative does not satisfy requirement (b) and may not conform to IEEE 1484.12.1-2002. This alternative is defined in the `unique/loose.xsd` component XSD (see C.2).
- `choice with unique`: Overcomes the problems with the XML Schema element `choice`. An attribute is applied to each element with a multiplicity of at most one. The attribute is enforced by applying an identity constraint. This alternative is defined in the `unique/strict.xsd` component XSD (see C.2).

## E.4 Vocabularies

The selection of the content model for LOM data elements of type Vocabulary (see 5.5.5) involves satisfying three requirements that arise either directly or indirectly from the LOMv1.0 base schema and from how vocabularies are to be used by applications. These are

- a) Data elements of type Vocabulary should allow arbitrary values (source/value pairs).
- b) If the source of a vocabulary value is the LOMv1.0 base schema (i.e., `<source>LOMv1.0</source>`), then the token used to represent the vocabulary value is defined in Clause 5 of this Standard.
- c) If the source of a vocabulary value is not the LOMv1.0 base schema (i.e., not `<source>LOMv1.0</source>`), then the token used to represent the vocabulary value should not conflict with any of the tokens defined in Clause 5 of this Standard.

The W3C XML Schema definition language cannot easily accommodate the goal of strict type checking for vocabulary values in a one-pass model. It can provide validation of a fixed set of vocabularies, but cannot validate arbitrary vocabularies. In LOM XML instances, the fixed vocabularies are defined based on the source of the vocabulary. If the source of the vocabulary is defined, W3C XML Schema definition language constructs can be defined to aid in the validation of the vocabularies.

Three alternatives for expressing LOM vocabularies in W3C XML Schema definition language are presented in E.4.1 – E.4.3. These three alternatives provide options on the validation approaches taken by XML Schema processors.

NOTE—In E.4.1 – E.4.3, "a", "b", and "c" in parentheses refer to the requirements listed above.

### E.4.1 vocab/loose.xsd

If no requirements exist for validation of vocabulary values from the LOMv1.0 base schema, then the `vocab/loose.xsd` component XSD may be used. This component XSD provides a way to meet the goal of allowing values from arbitrary sources (a) but does not validate that the vocabulary value is a valid member of the set of values defined by the source (b and c). This approach is the simplest alternative in terms of the complexity of the XSD. Each element that takes its values from a vocabulary is defined as a `CharacterString` as shown in Figure E.5.

```
<xs:simpleType name="difficulty">
  <xs:restriction base="lom:CharacterString"/>
</xs:simpleType>
```

**Figure E.5—An example from the `vocab/loose.xsd` component XSD**

Both the `source` and the `value` elements are optional and may appear in any order. There are no constraints on vocabulary data elements.

This approach accepts a superset of what the LOMv1.0 base schema allows without introducing additional complexity in LOM XML instances. The disadvantage of this approach is that it does not support validation of vocabulary values. Using the `vocab/loose.xsd` component XSD requires processing the PSVI to validate vocabulary values used in both *strictly conforming* and *conforming* LOM XML instances.

### E.4.2 vocab/strict.xsd

If vocabularies are limited to only those defined in the LOMv1.0 base schema, then the `vocab/strict.xsd` component XSD may be used. This component XSD defines constraints for a validation of LOMv1.0 base schema vocabularies. The use of this component XSD requires the source to be LOMv1.0 (i.e., `<source>LOMv1.0</source>`) and the value to be a valid LOM-defined vocabulary (b) but does not allow values from arbitrary sources (a and c). This approach is a straightforward implementation of the values allowed by the vocabulary elements. As shown in Figure E.6, each element that takes its values from a vocabulary is defined where the `source` element is given a fixed value of LOMv1.0, and the value of the `value` element is derived from an enumerated list of the appropriate vocabulary values.

```
<xs:simpleType name="difficulty">
  <xs:union memberTypes="lom:difficultyValues"/>
</xs:simpleType>

<xs:complexType name="difficultyVocab">
```

```

<xs:choice minOccurs="0" maxOccurs="unbounded">
  <xs:element name="source" type="sourceValue"/>
  <xs:element name="value" type="difficultyValue"/>
  <xs:group ref="ex:customElements"/>
</xs:choice>
<xs:attributeGroup ref="ex:customAttributes"/>
</xs:complexType>

<xs:simpleType name="difficultyValues">
  <xs:restriction base="xs:token">
    <xs:enumeration value="very easy"/>
    <xs:enumeration value="easy"/>
    <xs:enumeration value="medium"/>
    <xs:enumeration value="difficult"/>
    <xs:enumeration value="very difficult"/>
  </xs:restriction>
</xs:simpleType>

```

**Figure E.6—An example from the vocab/strict.xsd component XSD**

Both the `source` and `value` elements are optional and may appear in any order. Values are validated against the vocabulary tokens defined in Clause 5 of this Standard.

This approach accepts a subset of what the LOMv1.0 base schema would allow by excluding vocabulary values with sources other than the LOMv1.0 base schema. The advantage of this approach is that it can validate the vocabulary values defined in this Standard.

### E.4.3 vocab/custom.xsd

To use both vocabularies that are defined in the LOMv1.0 base schema and extended vocabularies, the `vocab/custom.xsd` component XSD may be used. This component XSD defines constraints for validation of both LOMv1.0 base schema vocabularies and vocabularies that are defined in other schemas. The use of this component XSD provides a way to meet the goal of allowing values from arbitrary sources (a) by supporting the validation of both vocabulary values from other sources (c) and LOMv1.0 base schema vocabulary values (b). This approach provides additional flexibility in the partial validation of both standard and extended vocabulary values. As shown in Figure E.7 each element that takes its values from a vocabulary is defined where the `source` element is allowed to have an arbitrary value, and the value of the `value` element is derived from a custom list of the appropriate vocabulary values.

```

<xs:simpleType name="difficulty">
  <xs:union memberTypes="lom:difficultyValues lx:difficultyValues"/>
</xs:simpleType>

<xs:complexType name="difficulty">
  <xs:complexContent>
    <xs:extension base="difficultyVocab">

```

```

        <xs:attributeGroup ref="ag:difficulty"/>
    </xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:simpleType name="difficultyValues">
    <xs:restriction base="xs:token">
        <xs:enumeration value="very easy"/>
        <xs:enumeration value="easy"/>
        <xs:enumeration value="medium"/>
        <xs:enumeration value="difficult"/>
        <xs:enumeration value="very difficult"/>
    </xs:restriction>
</xs:simpleType>

```

**Figure E.7—An example from the vocab/custom.xsd component XSD**

Both the `source` and `value` elements are optional and may appear in any order. Values are validated against the extended vocabulary enumerations.

This approach accepts a superset of what the LOMv1.0 base schema allows and allows the use of both standard and extended vocabulary values. The advantage of this approach is that it retains the ability for the validation of standard vocabulary values while accommodating the use of extended vocabulary values. Although the example in Figure E.7 restricts vocabulary values to those defined in the LOMv1.0 base schema when the source is LOMv1.0 (i.e., `<source>LOMv1.0</source>`), the example allows the use of vocabulary values defined in the LOMv1.0 base schema along with extended vocabulary values if the source is not LOMv1.0.

NOTE—If an organization plans to extend the vocabulary values and is following an XSD design similar to the one presented in this Standard, the organization should define the namespace that defines the extended vocabularies. The `vocab/custom.xsd` component XSD provides a placeholder for this extension (`xmlns:lx="http://ltsc.ieee.org/XSD/LOM/custom"`). The organization may change this declaration.

## E.5 Additional notes

This subclause discusses additional decisions that were made to enforce conformance to the LOMv1.0 base schema. The following information applies to implementation choices that were used in the XSD files provided with this Standard (see Annex B). These implementation choices are included in the `dataTypes.xsd` component XSD.

- Certain elements are defined in IEEE 1484.12.1–2002 to be represented as valid vCard syntax. At the time of publication of this Standard, no *de facto* standard W3C XML Schema definition language binding for the vCard specification was available. XSD implementations may want to define a type to encapsulate elements of type vCard. The vCard data type does not validate that the associated

- value represents a well-formed Internet Mail Consortium 3.0 vCard (see IETF RFC 2426:1998).
- The `MimeType` global W3C XML Schema definition language type is defined to encapsulate and define a type for those elements that are defined to hold a MIME type. The `MimeType` type does not validate that the `MimeType` value represents a valid MIME type (see IETF RFC 2425 [A2]).
  - An element type of `LanguageId` has been created to encapsulate the declaration for the `Language` element (see the `Educational` element, 5.4.5). This type allows the `Language` element to have a `Language` identifier (as defined in ISO 639–1, ISO 639–2, and ISO 3166–1) and multiplicity greater than one.
  - The global `<description>` element has content model `LanguageString` and multiplicity one.
  - Local element declarations for the `Description` element (see the `General` element, 5.4.1) and for the `Description` element (see the `Educational` element, 5.4.5) allow these elements to have multiplicity greater than one.
  - Because the `Description` element appears with different multiplicities in the LOMv1.0 base schema, two XML elements were created. The global element `description` has content model `LanguageString` and multiplicity one. The global element `descriptionUnbounded` has content model `LanguageString` and unbounded multiplicity.
  - Because the `Source` element is both a subelement of the `Taxon Path` element (see 5.4.9.2) and the `Vocabulary` data type (see 5.5.5) and has different data types depending on the `Source` element's aggregate element, two `source` declarations were developed. One allows validation of `Taxon Path` elements, and the other allows validation of `vocabulary` values.