IEEE 1484.11.3/Draft 6
Draft Standard for Learning Technology—
Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication

Sponsor
Learning Technology Standards Committee
of the
IEEE Computer Society

Abstract: This Standard specifies a World Wide Web Consortium (W3C) Extensible Markup Language (XML) binding of the data model defined in IEEE 1484.11.1–2004, "IEEE Standard for Learning Technology – Data Model for Content Object Communication." The purpose of this Standard is to allow the creation of IEEE 1418.11.1–2004 data-model instances in XML. This Standard uses the W3C XML Schema definition language as the encoding. This allows for interoperability and the exchange of data-model instances between various systems.

Keywords: content object, Content Object Communication Data Model, IEEE 1484.11.1–2004, Extensible Markup Language, XML, XML binding, XML data instance, XML Schema definition, W3C XML Schema definition language.

_____

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

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

[Note: Information about IEEE LTSC P1484.12.3 can be found at:

`http://ltsc.ieee.org/wg11/`

This note will be removed upon reaching the final draft of this IEEE document.]

Introduction

(This introduction is not a part of P1484.11.3, Draft Standard for Learning Technology—Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication.)

This Standard specifies a World Wide Web Consortium (W3C) Extensible Markup Language (XML) schema binding of the data model defined in IEEE 1484.11.1–2004, "IEEE Standard for Learning Technology – Data Model for Content Object Communication." The purpose of this Standard is to allow the creation of IEEE 1418.11.1–2004 data-model instances in XML. This Standard uses the W3C XML Schema definition language as the encoding. This allows for inter-operability and the exchange of data-model instances between various systems.

Participants

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

Kerry Blinco, *Chair*
Tyde Richards, *Chair 2001 – 2004*
Scott Lewis, *Technical Editor*

Jack Hyde	Rolf Lindner	Schawn Thropp
Tom King	Claude Ostyn	

The following persons were on the balloting committee: (To be provided by IEEE editor at time of publication.)

Also included are the following nonvoting IEEE–SA Standards Board liaisons:

To be supplied

Acknowledgements

The working group would like to acknowledge Claude Ostyn for providing the original schema and example instance that were used in the development of this Standard.

Contents

Draft Standard for Learning Technology— Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication

1. Overview

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

1.1 Scope

This Standard specifies a World Wide Web Consortium (W3C) Extensible Markup Language (XML) Schema binding of the data model defined in IEEE 1484.11.1–2004, "IEEE Standard for Learning Technology – Data Model for Content Object Communication."[1] An implementation that conforms to this Standard shall conform to IEEE 1484.11.1–2004.

1.2 Purpose

The purpose of this Standard is to allow the creation of IEEE 1418.11.1–2004 data-model instances in XML. This Standard uses the W3C XML Schema definition language to specify the encoding of these data-model instances (see XML Schema Parts 1 and 2). This allows for interoperability and the exchange of data-model 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 1484.11.1–2004, IEEE Standard for Learning Technology—Data Model for Content Object Communication.

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

[1] For information on normative references, see Clause 2.

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

3. Definitions

For purposes of this Standard, the following terms and definitions apply. IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [B1][2], should be referenced for terms not defined in this Clause.

content object: A collection of digital content that is intended for presentation to a learner by a learning technology system. A content object may include learning material and processing code. *Example:* A content object might be an interactive HTML page with an embedded video clip and an ECMAScript.

content object communication data Extensible Markup Language instance (COCD XML instance): A particular XML representation of the data model defined in IEEE 1484.11.1–2004 that adheres to the requirements and constraints of an XML binding of the data model.

Extensible Markup Language binding (XML binding): In this Standard, the method of encoding the behaviors, attributes, and value spaces of data-model elements in W3C Extensible Markup Language. This method is specified using the W3C XML Schema definition language.

3.1 Acronyms and abbreviations

COCD: content object communication data

SPM: smallest permitted maximum

W3C: World Wide Web Consortium

XML: Extensible Markup Language

XSD: XML Schema definition

4. Conformance

This Standard defines conforming IEEE 1481.11.1–2004 content object communication data (COCD) instances in an XML binding. Hereafter, such instances are referred to as "COCD XML instances."

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

[2] The numbers in brackets correspond to those of the bibliography in Annex A.

A conforming COCD XML instance

- Shall conform to the data-model requirements of IEEE 1484.11.1–2004;
- Shall not contain any extensions to the data model defined in IEEE 1484.11.1–2004;
- Shall be valid according to the XML Schema definition (XSD) specified in Annex B;
- Shall not contain any elements or attributes not defined in the XSD specified in Annex B; and
- Shall consist of a single element and its descendants. The single element shall have the name "cocd" as defined in the XSD specified in Annex B. The single element shall reside within the scope of a namespace declaration using the namespace specified in Clause 5.

NOTES:

1—This Standard does not require that the COCD XML instance be an XML document. The instance may be embedded at any depth in an XML data instance that includes elements from other namespaces.

2—IEEE 1484.11.1–2004 defines smallest permitted maximum (SPM) values. If a COCD XML instance contains more than the SPM number of occurrences of a COCD element, implementers should be aware that it is not guaranteed that an application will process more than the SPM number of occurrences of the COCD element. If a COCD XML instance contains more than the SPM number of characters in a character string, implementers should be aware that it is not guaranteed that an application will process more than the SPM number of characters in the character string.

3—The W3C XML Schema definition language cannot express and enforce all the data-model requirements of IEEE 1484.11.1–2004 (e.g., the requirements for SPMs).

5. XML binding

The namespace for the XML binding is defined by the conforming XSD in Annex B and shall be

```
http://ltsc.ieee.org/xsd/1484_11_3
```

The XSD in Annex B conforms to *XML Schema Parts 1 and 2*, October, 2004.

An example COCD XML instance is given in Annex C.

NOTE—The recommended file name for the XSD is "ieee_1484_11_3_2005.xsd". This file name should be treated as a reserved file name; it should not be used to name any file other than the conforming XSD defined in Annex B.

Annex A

(informative)

Bibliography

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

[B2]ISO/IEC 11404:1996, Information technology – Programming languages, their environments and system software interfaces – Language-independent datatypes.

Annex B

(normative)

Normative XSD

Figure B1 shows the conforming XSD for the data model defined by IEEE 1484.11.1–2004.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://ltsc.ieee.org/xsd/1484_11_3"
  xmlns:t="http://ltsc.ieee.org/xsd/1484_11_3"
  targetNamespace="http://ltsc.ieee.org/xsd/1484_11_3"
  elementFormDefault="qualified" version="1484.11.3-1.0">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      This schema is specified in IEEE 1484.11.3-2005, "IEEE Standard
      for Learning Technology - Extensible Markup Language (XML)
      Binding for Data Model for Content Object Communication."
      This schema is a World Wide Web Consortium (W3C) Extensible
      Markup Language (XML) binding of the data model defined in IEEE
      1484.11.1-2004, "IEEE Standard for Learning Technology - Data
      Model for Content Object Communication."
      The purpose of this schema is to allow the creation of IEEE
      1418.11.1-2004 data-model instances in XML. This schema uses the
      W3C XML Schema definition language as the encoding. This allows
      for interoperability and the exchange of data-model instances
      between various systems.
      This schema shall not be modified but may be included in
      derivative works.

      Copyright (c) 2005 Institute of Electrical and Electronics
      Engineers, Inc.

      USE AT YOUR OWN RISK

    </xs:documentation>
  </xs:annotation>
  <!--  -->
  <!-- The first half of this document follows the order of the
       Data Model document clauses. -->
  <!-- -->
  <xs:element name="cocd" type="cocdType"/>
  <xs:complexType name="cocdType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements IEEE 1484.11.1-2004,
        Clause 6.1: Content object communication
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element ref="commentsFromLearner" minOccurs="0"/>
      <xs:element ref="commentsFromLMS" minOccurs="0"/>
      <xs:element ref="completionStatus" minOccurs="0"/>
      <xs:element ref="completionThreshold" minOccurs="0"/>
      <xs:element ref="credit" minOccurs="0"/>
```

```
      <xs:element ref="dataModelVersion" minOccurs="0"/>
      <xs:element ref="entry" minOccurs="0"/>
      <xs:element ref="exit" minOccurs="0"/>
      <xs:element ref="interactions" minOccurs="0"/>
      <xs:element ref="launchData" minOccurs="0"/>
      <xs:element ref="learnerId" minOccurs="0"/>
      <xs:element ref="learnerName" minOccurs="0"/>
      <xs:element ref="learnerPreferenceData" minOccurs="0"/>
      <xs:element ref="lessonStatus" minOccurs="0"/>
      <xs:element ref="location" minOccurs="0"/>
      <xs:element ref="maxTimeAllowed" minOccurs="0"/>
      <xs:element ref="mode" minOccurs="0"/>
      <xs:element ref="objectives" minOccurs="0"/>
      <xs:element ref="progressMeasure" minOccurs="0"/>
      <xs:element ref="rawPassingScore" minOccurs="0"/>
      <xs:element ref="scaledPassingScore" minOccurs="0"/>
      <xs:element ref="score" minOccurs="0"/>
      <xs:element ref="sessionTime" minOccurs="0"/>
      <xs:element ref="successStatus" minOccurs="0"/>
      <xs:element ref="suspendData" minOccurs="0"/>
      <xs:element ref="timeLimitAction" minOccurs="0"/>
      <xs:element ref="totalTime" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
  <xs:element name="commentsFromLearner">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.1:  Comments from learner
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="commentFromLearner" type="commentType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="collectionType" fixed="array"/>
      <xs:attribute name="spm" fixed="250"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="commentsFromLMS">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.2: Comments from LMS
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="commentFromLMS" type="commentType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="collectionType" fixed="array"/>
      <xs:attribute name="spm" fixed="100"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="completionStatus" type="completionStatusType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.3: Completion status
      </xs:documentation>
    </xs:annotation>
```



```
  </xs:element>
  <xs:element name="completionThreshold" type="progressMeasureType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.4: Completion threshold
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="credit">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.5: Credit
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="credit"/>
        <xs:enumeration value="no_credit"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="dataModelVersion" type="literalString250Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.6: Data model version
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="entry">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.7: Entry
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="ab_initio"/>
        <xs:enumeration value="resume"/>
        <xs:enumeration value=""/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="exit">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.8: Exit
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="logout"/>
        <xs:enumeration value="normal"/>
        <xs:enumeration value="suspend"/>
        <xs:enumeration value="timeout"/>
        <xs:enumeration value=""/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="interactions" type="interactionsType">
    <xs:annotation>
```



```
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.9: Interactions
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="launchData" type="literalString4000Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.10: Launch data
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="learnerId" type="longIdentifierType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.11: Learner ID
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="learnerName" type="localizedString250Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.12: Learner name
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="learnerPreferenceData"
    type="learnerPreferenceType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.13: Learner preference data
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="lessonStatus" type="legacyStatusType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.14: Lesson status
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="location" type="literalString1000Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.15: Location
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="maxTimeAllowed" type="timeIntervalType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.16: Max time allowed
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="mode">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.17: Mode
      </xs:documentation>
```

```
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="browse"/>
        <xs:enumeration value="normal"/>
        <xs:enumeration value="review"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="objectives" type="objectivesType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.18: Objectives
      </xs:documentation>
    </xs:annotation>
    <xs:unique name="uniqueInSetOfObjectives">
      <xs:selector xpath="./t:objective"/>
      <xs:field xpath="t:identifier"/>
    </xs:unique>
  </xs:element>
  <xs:element name="progressMeasure" type="progressMeasureType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.19: Progress measure
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="rawPassingScore" type="real7Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.20: Raw passing score
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="scaledPassingScore" type="scaledScoreType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.21: Scaled passing score
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="score" type="scoreType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.22: Score
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="sessionTime" type="timeIntervalType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.23: Session time
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="successStatus" type="successStatusType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.24: Success status
      </xs:documentation>
```

```
    </xs:annotation>
  </xs:element>
  <xs:element name="suspendData" type="literalString4000Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.25: Suspend data
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="timeLimitAction">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.26: Time limit action
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="continue_message"/>
        <xs:enumeration value="continue_no_message"/>
        <xs:enumeration value="exit_message"/>
        <xs:enumeration value="exit_no_message"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="totalTime" type="timeIntervalType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.27: Total time
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <!--  -->
  <!-- Global type declarations defined by numbered clauses in
    IEEE 1484.11.1 -->
  <!--  -->
  <xs:complexType name="commentType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.1: Comment type
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="comment" type="localizedString4000Type"/>
      <xs:element name="location" type="literalString1000Type"
        minOccurs="0"/>
      <xs:element name="timeStamp" type="dateTimeType" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
  <xs:simpleType name="completionStatusType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.2: Completion status type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="completed"/>
      <xs:enumeration value="incomplete"/>
      <xs:enumeration value="not_attempted"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
```



```
  </xs:simpleType>
  <xs:simpleType name="dateTimeType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.3: Date time type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:dateTime"/>
  </xs:simpleType>
  <xs:simpleType name="languageType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.4: Language type. Must remain simpleType
        because it is used as value for attributes
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:language">
      <xs:annotation>
        <xs:appinfo>
          <!-- <spm>250</spm> -->
        </xs:appinfo>
      </xs:annotation>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="localizedStringType" abstract="true">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.5: Localized string type.
        The Localized String Type is implemented as several
        variations with embedded SPM information.
        The SPM is not enforced by XML validators but may
        be useful for applications.
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="literalStringType">
        <xs:attribute name="lang" type="languageType"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="localizedString250Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.5: Localized string type with SPM=250
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="localizedStringType">
        <xs:attribute name="spm" fixed="250"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="localizedString4000Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.5: Localized string type with SPM=4000
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="localizedStringType">
```



```
        <xs:attribute name="spm" fixed="4000"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="longIdentifierType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.6: Long identifier type
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="spm" fixed="4000"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="progressMeasureType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.7: Progress measure type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="real7Type">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="scoreType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.8: Score type
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="scaled" type="scaledScoreType" minOccurs="0"/>
      <xs:element name="max" type="real7Type" minOccurs="0"/>
      <xs:element name="min" type="real7Type" minOccurs="0"/>
      <xs:element name="raw" type="real7Type" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="shortIdentifierType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.9: Short identifier type
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="spm" fixed="250"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:simpleType name="successStatusType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.2.10: Success status type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="failed"/>
```



```
      <xs:enumeration value="passed"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="real7Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        As explained in IEEE 1484.11.1-2004, Annex B.1 Real data type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal"/>
  </xs:simpleType>
  <xs:simpleType name="timeIntervalType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        As explained in IEEE 1484.11.1-2004, Annex B.2 Time interval
        Data type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:duration"/>
  </xs:simpleType>
  <!--  -->
  <!-- Above this, things follow the order of the Data Model
       document clauses. -->
  <!-- ============================================================= -->
  <!-- Below this are things that did not fit neatly above. They are
       organized as elements, attributes, groups, simple types, complex
       types and alphabetically by name within each of those
       categories.  -->
  <!--  -->
  <!-- === ELEMENTS === -->
  <!-- Organized in alphabetic order by element name -->
  <!--  -->
  <xs:element name="choices" type="setOfChoicesType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Set of short identifiers for interaction type "multiple
        choice" as specified in 6.1.9.5: Correct response and 6.1.9.7:
        Learner response.
    </xs:documentation>
    </xs:annotation>
    <xs:unique name="uniqueInChoicesIds">
      <xs:selector xpath="./t:choice"/>
      <xs:field xpath="."/>
    </xs:unique>
  </xs:element>
  <!--  -->
  <!-- === ATTRIBUTES === -->
  <!-- Organized in alphabetic order by attribute name -->
  <!--  -->
  <xs:attribute name="collectionType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        The collectionType attribute is used to inject Data Model
        information about aggregation that cannot be expressed in XML
        schema. When defined for an element or type in this schema,
        this attribute is given a fixed values. Even if the attribute
        and value are not specified in an XML instance, the XML schema
        processor makes them available to the processing application.
      </xs:documentation>
```



```
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="bag"/>
        <xs:enumeration value="array"/>
        <xs:enumeration value="set"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="spm" type="xs:integer">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        The spm attribute is used to inject Data Model information
        about SPM that cannot be expressed in XML schema. When defined
        for an element or type in this schema, this attribute is given
        a fixed values. Even if the attribute and value are not
        specified in an XML instance, the XML schema processor makes
        them available to the processing application.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <!--  -->
  <!-- === GROUPS === -->
  <!-- Organized in alphabetic order by group name -->
  <!--  -->
  <xs:group name="grpCorrectFillIn">
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>5</spm><collectionType>bag</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="fillMatches" minOccurs="0"
        maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="matchText" type="localizedString250Type"
              maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="caseMatters" type="trueFalseType"
            use="optional" default="false"/>
          <xs:attribute name="orderMatters" type="trueFalseType"
            use="optional" default="true"/>
          <xs:attribute name="collectionType" fixed="array"/>
          <xs:attribute name="spm" fixed="10"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectLikert">
    <xs:sequence>
      <xs:element name="choice" type="shortIdentifierType"
        minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectLongFillIn">
    <xs:sequence>
      <xs:annotation>
```



```
        <xs:appinfo>
          <!--
            <spm>5</spm><collectionType>bag</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="matchText" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="localizedString4000Type">
              <xs:attribute name="caseMatters" type="trueFalseType"
                use="optional" default="false"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectMatching">
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>5</spm><collectionType>bag</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="matchPattern" type="matchingPairsType"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectMultipleChoice">
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>10</spm><collectionType>set</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element ref="choices" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectNumeric">
    <xs:sequence>
      <xs:element name="min" type="real7Type" minOccurs="0"/>
      <xs:element name="max" type="real7Type" minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectOther">
    <xs:sequence>
      <xs:element name="correctOther" type="literalString4000Type"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectPerformance">
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>5</spm><collectionType>bag</collectionType>
```



```
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="performancePattern"
        type="correctPerformancePatternType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectSequencing">
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>5</spm><collectionType>bag</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="stepSequence" type="stepSequenceType"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpCorrectTrueFalse">
    <xs:sequence>
      <xs:element name="trueOrFalse" type="trueFalseType"/>
    </xs:sequence>
  </xs:group>
  <!-- variant groups for interaction responses -->
  <xs:group name="grpResponseFillIn">
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>10</spm><collectionType>array</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="fillString" type="localizedString250Type"
        minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:appinfo>
            <!-- <spm>250</spm> -->
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseLikert">
    <xs:sequence>
      <xs:element name="choice" type="shortIdentifierType"
        minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseLongFillIn">
    <xs:sequence>
      <xs:element name="longFillString" type="localizedString4000Type"
        minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseMatching">
    <xs:sequence>
      <xs:element name="matchPattern" type="matchingPairsType"/>
```



```
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseMultipleChoice">
    <xs:sequence>
      <xs:element ref="choices"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseNumeric">
    <xs:sequence>
      <xs:element name="number" type="real7Type" minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseOther">
    <xs:sequence>
      <xs:element name="responseOther" type="literalString4000Type"/>
    </xs:sequence>
  </xs:group>
  <!--  -->
  <xs:group name="grpResponsePerformance">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        The learner response for interaction type "performance"
        as specified in 6.1.9.7: Learner response
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:annotation>
        <xs:appinfo>
          <!--
            <spm>250</spm><collectionType>array</collectionType>
          -->
        </xs:appinfo>
      </xs:annotation>
      <xs:element name="step" type="learnerPerformanceStepType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseSequencing">
    <xs:sequence>
      <xs:element name="steps" type="stepSequenceType" minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseTrueFalse">
    <xs:sequence>
      <xs:element name="trueOrFalse" type="trueFalseType"
        minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <!--  -->
  <!-- === SIMPLE TYPES === -->
  <!-- Organized in alphabetic order by type name -->
  <!--  -->
  <xs:simpleType name="interactionResultType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Reusable type definition used for 6.1.9.8: Result
        The value of result can be either a numeric value
        or a specified token. This element uses xs:union to avoid
        having to define sub-elements with arbitrary names.
      </xs:documentation>
```



```
    </xs:annotation>
    <xs:union memberTypes="real7Type interactionResultTokenType"/>
  </xs:simpleType>
  <xs:simpleType name="interactionResultTokenType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="correct"/>
      <xs:enumeration value="incorrect"/>
      <xs:enumeration value="neutral"/>
      <xs:enumeration value="unanticipated"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="interactionTypeType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.9.2: Type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="true_false"/>
      <xs:enumeration value="multiple_choice"/>
      <xs:enumeration value="fill_in"/>
      <xs:enumeration value="long_fill_in"/>
      <xs:enumeration value="likert"/>
      <xs:enumeration value="matching"/>
      <xs:enumeration value="performance"/>
      <xs:enumeration value="sequencing"/>
      <xs:enumeration value="numeric"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="literalStringType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Ensures preservation of whitespace
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="legacyStatusType">
    <xs:restriction base="xs:token">
      <xs:enumeration value="browsed"/>
      <xs:enumeration value="completed"/>
      <xs:enumeration value="failed"/>
      <xs:enumeration value="incomplete"/>
      <xs:enumeration value="not_attempted"/>
      <xs:enumeration value="passed"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="scaledScoreType">
    <xs:restriction base="real7Type">
      <xs:minInclusive value="-1"/>
      <xs:maxInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="trueFalseType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        True and false options for interaction type "true_false"
```



```
        as specified in 6.1.9.5: Correct response and 6.1.9.7:
        Learner response.
        Also used for tokens for various other boolean elements.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="true"/>
      <xs:enumeration value="false"/>
    </xs:restriction>
  </xs:simpleType>
  <!--  -->
  <!-- === COMPLEX TYPES === -->
  <!-- Organized in alphabetic order by type name -->
  <!--  -->
  <xs:complexType name="setOfChoicesType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Set of short identifiers for interaction type "multiple choice"
        as specified in 6.1.9.5: Correct response and 6.1.9.7: Learner
        response.
    </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="choice" type="shortIdentifierType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="collectionType" fixed="set"/>
    <xs:attribute name="spm" fixed="36"/>
  </xs:complexType>
  <xs:complexType name="correctPerformancePatternType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        A correct response record (order_matters + answers)
        For interaction type "performance"
        as specified in 6.1.9.5: Correct response
    </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="step" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>
            <xs:element name="stepName" type="shortIdentifierType"
              minOccurs="0"/>
            <xs:element name="stepAnswer" minOccurs="0">
              <xs:complexType>
                <xs:choice>
                  <xs:element name="literal"
                    type="literalString250Type" minOccurs="0"/>
                  <xs:element name="numeric" minOccurs="0">
                    <xs:complexType>
                      <xs:attribute name="min" type="real7Type"/>
                      <xs:attribute name="max" type="real7Type"/>
                    </xs:complexType>
                  </xs:element>
                </xs:choice>
              </xs:complexType>
            </xs:element>
          </xs:all>
        </xs:complexType>
      </xs:element>
```



```
    </xs:sequence>
    <xs:attribute name="orderMatters" type="trueFalseType"
      use="optional" default="true"/>
    <xs:attribute name="collectionType" fixed="array"/>
    <xs:attribute name="spm" fixed="250"/>
  </xs:complexType>
  <xs:complexType name="correctResponsesType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.9.5: Correct responses.
        Note: It is up to the implementation to choose the correct
        group to match the interaction type. This correspondence cannot
        be expressed or validated using XML Schema.
      </xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:group ref="grpCorrectTrueFalse"/>
      <xs:group ref="grpCorrectMultipleChoice"/>
      <xs:group ref="grpCorrectFillIn"/>
      <xs:group ref="grpCorrectLongFillIn"/>
      <xs:group ref="grpCorrectLikert"/>
      <xs:group ref="grpCorrectMatching"/>
      <xs:group ref="grpCorrectPerformance"/>
      <xs:group ref="grpCorrectSequencing"/>
      <xs:group ref="grpCorrectNumeric"/>
      <xs:group ref="grpCorrectOther"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="interactionType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Reusable type to implements a single interaction record as
        defined in Clause 6.1.9.
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="identifier" type="longIdentifierType">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.1: ID
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="type" type="interactionTypeType">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.2: Type
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="objectiveIds" type="objectiveIdsType"
        minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.3: Objectives ID
          </xs:documentation>
        </xs:annotation>
        <xs:unique name="uniqueInObjectivesIds">
          <xs:selector xpath="./t:objectiveId"/>
          <xs:field xpath="."/>
```



```
        </xs:unique>
      </xs:element>
      <xs:element name="timeStamp" type="dateTimeType" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.4: Time stamp
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="correctResponses" type="correctResponsesType"
        minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.5: Correct responses
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="weighting" type="real7Type" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.6: Weighting
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="learnerResponse" type="learnerResponseType"
        minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.7: Learner response
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="result" type="interactionResultType"
        minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.8: Result
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="latency" type="timeIntervalType" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.9.9: Latency
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="description" type="localizedString250Type"
        minOccurs="0"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="interactionsType">
    <xs:sequence>
      <xs:element name="interaction" type="interactionType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="collectionType" fixed="bag"/>
    <xs:attribute name="spm" fixed="250"/>
  </xs:complexType>
  <xs:complexType name="learnerPerformanceStepType">
```



```
    <xs:all>
      <xs:element name="stepName" type="shortIdentifierType"
        minOccurs="0"/>
      <xs:element name="stepAnswer" minOccurs="0">
        <xs:complexType>
          <xs:choice>
            <xs:element name="literal" type="literalString250Type"
              minOccurs="0"/>
            <xs:element name="numeric" type="real7Type" minOccurs="0"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="learnerPreferenceType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implementslearner_preference_type in Clause 6.1.13:
        Learner preference data
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="audioLevel" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.13.1: Audio level
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="real7Type">
            <xs:minInclusive value="0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="language" type="languageType" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.13.2: Language
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="deliverySpeed" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.13.3: Delivery speed
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="real7Type">
            <xs:minInclusive value="0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="audioCaptioning" minOccurs="0">
        <xs:annotation>
          <xs:documentation xml:lang="en">
            Implements Clause 6.1.13.4: Audio captioning
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
```



```
          <xs:restriction base="xs:token">
            <xs:enumeration value="off"/>
            <xs:enumeration value="no_change"/>
            <xs:enumeration value="on"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="learnerResponseType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implements Clause 6.1.9.7: Learner response.
        Note: It is up to the implementation to choose the correct
        group or element name to match the interaction type. This
        correspondence cannot be expressed or validated using XML
        Schema.
          </xs:documentation>
    </xs:annotation>
    <xs:choice>
      <xs:group ref="grpResponseTrueFalse"/>
      <xs:group ref="grpResponseMultipleChoice"/>
      <xs:group ref="grpResponseFillIn"/>
      <xs:group ref="grpResponseLongFillIn"/>
      <xs:group ref="grpResponseLikert"/>
      <xs:group ref="grpResponseMatching"/>
      <xs:group ref="grpResponsePerformance"/>
      <xs:group ref="grpResponseSequencing"/>
      <xs:group ref="grpResponseNumeric"/>
      <xs:group ref="grpResponseOther"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="literalString250Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implement any literal string with SPM=250
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="literalStringType">
        <xs:attribute name="spm" fixed="250"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="literalString1000Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implement any literal string with SPM=1000
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="literalStringType">
        <xs:attribute name="spm" fixed="1000"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="literalString4000Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Implement any literal string with SPM=4000
```



```
      </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="literalStringType">
        <xs:attribute name="spm" fixed="4000"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="matchingPairType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        A pair of matched short identifiers for interaction type
        "matching" as specified in 6.1.9.5: Correct response and
        6.1.9.7: Learner response.
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="source" type="shortIdentifierType"/>
      <xs:element name="target" type="shortIdentifierType"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="matchingPairsType">
    <xs:sequence>
      <xs:annotation>
        <xs:documentation xml:lang="en">
          A collection of 0 or more instances of matchingPairType.
        </xs:documentation>
      </xs:annotation>
      <xs:element name="pair" type="matchingPairType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="collectionType" fixed="bag"/>
    <xs:attribute name="spm" fixed="36"/>
  </xs:complexType>
  <xs:complexType name="objectiveIdsType">
    <xs:sequence>
      <xs:element name="objectiveId" type="longIdentifierType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="collectionType" fixed="array"/>
    <xs:attribute name="spm" fixed="10"/>
  </xs:complexType>
  <xs:complexType name="objectivesType">
    <xs:sequence>
      <xs:element name="objective" type="objectiveType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="collectionType" fixed="set"/>
    <xs:attribute name="spm" fixed="100"/>
  </xs:complexType>
  <xs:complexType name="objectiveType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Reusable type to implement the objective_type record
        defined in Clause 6.1.18
      </xs:documentation>
    </xs:annotation>
    <xs:all>
      <xs:element name="identifier" type="longIdentifierType"/>
      <xs:element name="score" type="scoreType" minOccurs="0"/>
```



```
      <xs:element name="status" type="legacyStatusType" minOccurs="0"/>
      <xs:element name="progressMeasure" type="progressMeasureType"
        minOccurs="0"/>
      <xs:element name="completionStatus" type="completionStatusType"
        minOccurs="0"/>
      <xs:element name="successStatus" type="successStatusType"
        minOccurs="0"/>
      <xs:element name="description" type="localizedString250Type"
        minOccurs="0"/>
    </xs:all>
  </xs:complexType>
  <xs:complexType name="responseOtherType">
    <xs:simpleContent>
      <xs:extension base="literalString4000Type"/>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="stepSequenceType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        A sequence of steps for interaction type "sequencing"
        as specified in 6.1.9.5: Correct response and 6.1.9.7:
        Learner response.
        The same identifier may appear more than once in the list,
        because a step may be repeated in an interaction.
.      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="step" type="shortIdentifierType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="collectionType" fixed="array"/>
    <xs:attribute name="spm" fixed="36"/>
  </xs:complexType>
</xs:schema>
```

Figure B1—Conforming XSD

Annex C

(informative)

An example COCD XML instance

Figure C1 shows a COCD XML instance that instantiates all of the elements and attributes defined in the XSD in Annex B.

```
<?xml version="1.0" encoding="UTF-8"?>
<cocd xmlns="http://ltsc.ieee.org/xsd/1484_11_3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ltsc.ieee.org/xsd/1484_11_3
    ieee_1484_11_3_2005.xsd">
  <commentsFromLearner>
    <commentFromLearner>
      <comment lang="en-us">Having a good time</comment>
      <location>Somewhere</location>
      <timeStamp>2005-10-17T09:30:47-05:00</timeStamp>
    </commentFromLearner>
  </commentsFromLearner>
  <commentsFromLMS>
    <commentFromLMS>
      <comment lang="en-us">Have a good time</comment>
      <location>Wherever it may be</location>
      <timeStamp>2005-10-17T08:30:02-05:00</timeStamp>
    </commentFromLMS>
  </commentsFromLMS>
  <completionStatus>completed</completionStatus>
  <completionThreshold>0.9</completionThreshold>
  <credit>credit</credit>
  <dataModelVersion>1484.11.1</dataModelVersion>
  <entry>ab_initio</entry>
  <exit>logout</exit>
  <launchData>Nothing special</launchData>
  <learnerId>1234-foobar-1234</learnerId>
  <learnerName lang="en-us">John Doe</learnerName>
  <learnerPreferenceData>
    <audioLevel>1</audioLevel>
    <language>en-us</language>
    <deliverySpeed>1.25</deliverySpeed>
    <audioCaptioning>off</audioCaptioning>
  </learnerPreferenceData>
  <lessonStatus>browsed</lessonStatus>
  <location>Page 4, paragraph 6</location>
  <maxTimeAllowed>P2M3DT10H30M</maxTimeAllowed>
  <mode>browse</mode>
  <interactions>
    <interaction>
      <identifier>urn:ostyn.com:TFQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>1234-foobar-1234</objectiveId>
        <objectiveId>1235-barfoo-1345</objectiveId>
     </objectiveIds>
      <timeStamp>2005-10-17T09:30:17-05:00</timeStamp>
```



```
      <description>Vanilla is lighter than chocolate. True or
false?</description>
      <result>unanticipated</result>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>true_false</type>
      <correctResponses>
        <trueOrFalse>true</trueOrFalse>
      </correctResponses>
      <learnerResponse/>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:MCQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234</objectiveId>
        <objectiveId>1235-barfoo-1345
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:32:47.45-05:00</timeStamp>
      <description>What was J.H.'s favorite ice cream
combination?</description>
      <result>incorrect</result>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>multiple_choice</type>
      <correctResponses>
        <choices>
          <choice>Vanilla</choice>
        </choices>
        <choices>
          <choice>Vanilla</choice>
        </choices>
        <choices>
          <choice>Chocolate</choice>
        </choices>
        <choices>
          <choice>Chocolate</choice>
          <choice>Vanilla</choice>
        </choices>
      </correctResponses>
      <learnerResponse>
        <choices>
          <choice>Pistachio</choice>
          <choice>Chocolate</choice>
        </choices>
      </learnerResponse>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:FIQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>1234-foobar-1234</objectiveId>
        <objectiveId>1235-barfoo-1345</objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:34:47-05:00</timeStamp>
      <description>Match things with numbers</description>
      <result>3.5926</result>
      <latency>PT30M</latency>
      <weighting>3</weighting>
      <type>fill_in</type>
```



```
      <correctResponses>
        <fillMatches caseMatters="false" orderMatters="false">
          <matchText lang="en">some</matchText>
          <matchText lang="en">thing</matchText>
        </fillMatches>
        <fillMatches caseMatters="false" orderMatters="true">
          <matchText>1</matchText>
          <matchText>2</matchText>
        </fillMatches>
      </correctResponses>
      <learnerResponse>
        <fillString lang="en">This is just any short response.</fillString>
      </learnerResponse>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:LFQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234</objectiveId>
        <objectiveId>
        1235-barfoo-1345
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:36:47-05:00</timeStamp>
      <description lang="fr">Début de la Ballade des Pendus</description>
      <result>-0.1415926</result>
      <latency>PT23.4S</latency>
      <weighting>33</weighting>
      <type>long_fill_in</type>
      <correctResponses>
        <matchText lang="fr" caseMatters="false">Frères humains qui après nous
vivez,
N'ayez les coeurs contre nous endurcis</matchText>
        <matchText lang="fr" caseMatters="false">Frères humains qui après nous
vivez,
N'ayez les cuers contre nous endurcis</matchText>
        <matchText lang="fr" caseMatters="true">Frères humains qui après nous
vivez</matchText>
      </correctResponses>
      <learnerResponse>
        <longFillString lang="fr-BE">Ça commence avec "Frères
humains..."</longFillString>
      </learnerResponse>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:LIQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:38:47-05:00</timeStamp>
      <description>Which approach is most likely to succeed?</description>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>likert</type>
      <learnerResponse>
        <choice>option_5</choice>
      </learnerResponse>
    </interaction>
```



```
    <interaction>
      <identifier>urn:ostyn.com:MAQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234</objectiveId>
        <objectiveId>
        1235-barfoo-1345
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:40:47-05:00</timeStamp>
      <description>Connect the shmiblicks to the corresponding
garfubles</description>
      <result>incorrect</result>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>matching</type>
      <correctResponses>
        <matchPattern>
          <pair>
            <source>something_A</source>
            <target>something_B</target>
          </pair>
          <pair>
            <source>something_C</source>
            <target>something_D</target>
          </pair>
          <pair>
            <source>something_E</source>
            <target>something_F</target>
          </pair>
        </matchPattern>
        <matchPattern>
          <pair>
            <source>something_C</source>
            <target>something_D</target>
          </pair>
          <pair>
            <source>something_E</source>
            <target>something_F</target>
          </pair>
        </matchPattern>
      </correctResponses>
      <learnerResponse>
        <matchPattern>
          <pair>
            <source>something_C</source>
            <target>something_D</target>
          </pair>
          <pair>
            <source>something_E</source>
            <target>something_F</target>
          </pair>
        </matchPattern>
      </learnerResponse>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:PEQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234</objectiveId>
```



```
        <objectiveId>
        1235-barfoo-1345
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:42:47-05:00</timeStamp>
      <description>Steps to diagnose the schmiblick</description>
      <result>3.5</result>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>performance</type>
      <correctResponses>
        <performancePattern orderMatters="true">
          <step>
            <stepName>StepB</stepName>
            <stepAnswer>
              <numeric min="10" max="10"/>
            </stepAnswer>
          </step>
          <step>
            <stepName>StepC</stepName>
            <stepAnswer>
              <literal>Green whatchamakalit</literal>
            </stepAnswer>
          </step>
          <step>
            <stepName>stepD</stepName>
          </step>
        </performancePattern>
        <performancePattern orderMatters="false">
          <step>
            <stepName>stepA</stepName>
            <stepAnswer>
              <literal>Push the diagnostic button</literal>
            </stepAnswer>
          </step>
          <step>
            <stepName>stepE</stepName>
          </step>
        </performancePattern>
      </correctResponses>
      <learnerResponse>
        <step>
          <stepName>StepC</stepName>
          <stepAnswer>
            <literal>Blue whatchamakalit</literal>
          </stepAnswer>
        </step>
        <step>
          <stepName>StepD</stepName>
          <stepAnswer>
            <numeric>8.7</numeric>
          </stepAnswer>
        </step>
        <step>
          <stepName>StepD</stepName>
        </step>
      </learnerResponse>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:INT2003090412345</identifier>
```



```
      <objectiveIds>
        <objectiveId>
        urn:bar.com/RCD/2345-800df-4%20test</objectiveId>
        <objectiveId>
        machinchose1234
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:44:47-05:00</timeStamp>
      <description>Steps to buy and enjoy ice cream.</description>
      <result>3.14159</result>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>sequencing</type>
      <correctResponses>
        <stepSequence>
          <step>Choose_flavor</step>
          <step>Order_ice_cream</step>
          <step>Eat_ice_cream</step>
          <step>Wipe_chin</step>
        </stepSequence>
        <stepSequence>
          <step>Raid_fridge</step>
          <step>Choose_flavor</step>
          <step>Eat_ice_cream</step>
          <step>Wipe_chin</step>
        </stepSequence>
      </correctResponses>
      <learnerResponse>
        <steps>
          <step>Order_ice_cream</step>
          <step>Choose_flavor</step>
          <step>Eat_ice_cream</step>
          <step>Wipe_chin</step>
          <step>Eat_ice_cream</step>
          <step>Wipe_chin</step>
        </steps>
      </learnerResponse>
    </interaction>
    <interaction>
      <identifier>urn:ostyn.com:NUQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:17:47-05:00</timeStamp>
      <description>Pick a likely number for the result of this
operation.</description>
      <result>3.14159</result>
      <latency>PT23.4S</latency>
      <weighting>7</weighting>
      <type>numeric</type>
      <correctResponses>
        <min>0</min>
        <max>123456783453.1415926</max>
      </correctResponses>
      <learnerResponse>
        <number>3.1415926</number>
      </learnerResponse>
    </interaction>
```



```
    <interaction>
      <identifier>urn:ostyn.com:OTQ2003090412345</identifier>
      <objectiveIds>
        <objectiveId>
        1234-foobar-1234
      </objectiveId>
      </objectiveIds>
      <timeStamp>2005-10-17T09:13:47-05:00</timeStamp>
      <description>Some other kind of interaction</description>
      <result>correct</result>
      <latency>PT23.4S</latency>
      <weighting>.33</weighting>
      <type>other</type>
      <correctResponses>
        <correctOther>Something and <![CDATA[<Something>more or less
complicated</Something>]]></correctOther>
      </correctResponses>
      <learnerResponse>
        <responseOther>Something more or less complicated</responseOther>
      </learnerResponse>
    </interaction>
  </interactions>
  <objectives>
    <objective>
      <identifier>urn:ostyn.com:id200309041234578</identifier>
      <completionStatus>incomplete</completionStatus>
      <description>Answer 10 questions</description>
      <score>
        <scaled>1.0</scaled>
        <max>77</max>
        <min>0</min>
        <raw>77</raw>
      </score>
      <status>browsed</status>
      <successStatus>failed</successStatus>
    </objective>
    <objective>
      <identifier>urn:ostyn.com:id200309041234576</identifier>
      <completionStatus>completed</completionStatus>
    </objective>
   </objectives>
  <progressMeasure>0.95</progressMeasure>
  <rawPassingScore>600</rawPassingScore>
  <scaledPassingScore>0.5</scaledPassingScore>
  <score>
    <scaled>0.5</scaled>
    <max>800</max>
    <min>400</min>
    <raw>600</raw>
  </score>
  <sessionTime>P3DT10H30M</sessionTime>
  <successStatus>passed</successStatus>
  <suspendData>Something=4; line break here:
save this white space (10 spaces)[          ]end of line
beginning of line. Something else on third line.</suspendData>
  <timeLimitAction>continue_message</timeLimitAction>
  <totalTime>P5DT10H30M</totalTime>
</cocd>
```


1798 **Figure C1—An example COCD XML instance**

Annex D

(informative)

Explanatory XSD notes

This annex is a guide to the understanding and use of the XSD in Annex B. This annex references specific subclauses of Clause 6 of IEEE 1484.11.1–2004. IEEE 1484.11.1–2004 is required to understand this annex.

Where possible, this annex is organized in the same order as the subclauses of Clause 6 of IEEE 1484.11.1–2004. The intent is to facilitate a parallel reading of this annex with IEEE 1484.11.1–2004, the XSD (see Annex B), and the sample XML instance (see Annex C).

NOTES

1—This annex is not intended to provide an explanation or rationale for every design or syntax choice implemented in the XSD nor is it intended to be an XML or XML Schema definition language tutorial. Familiarity with W3C XML Schema, Parts 1 and 2 is required to understand this annex. This annex mentions features specified by the W3C XML Schema definition language and tested with current implementations of XML schema processors.

2—In all examples XML fragments in this annex, the `xs:` prefix denotes a type, element, group, or attribute name defined by the W3C namespace "http://www.w3.org/2001/XMLSchema".

3—The first half of the XSD follows the order of the subclauses of Clause 6 of IEEE 1484.11.1–2004. The second half of the XSD uses conventional groupings of elements and type declarations: first elements, then attributes, groups, simple types, and complex types. Within each grouping the elements are arranged alphabetically by name.

D.1 Limitations of the W3C XML Schema definition language for the representation of the data model

Some of the requirements defined in the data model cannot be expressed in the W3C XML Schema definition language in any way that would allow automatic validation or constraint enforcement by a generic XML Schema processor. These requirements are discussed in D.1.1 – D.1.4.

D.1.1 Encoding of bags, arrays and sets

The bags, arrays, and sets defined by the data model are implemented with the XML Schema compositor `xs:sequence`. For collections of like elements, `xs:sequence` is used because it is the only way to allow a cardinality of more than one for the contained element, even though `xs:sequence` implies that the contents of the complex type have to appear as an ordered list. A typical XSD construct for a bag, array, or set is similar to the following example.


```
<xs:element name="objectives">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="objective" type="objectiveType"
            minOccurs="0" maxOccurs="unbounded">
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

This may be instantiated in a COCD XML instance as in the following example.

```
<objectives>
   <objective>
      <identifier>urn:foo.com:id200309041234578</identifier>
      <completionStatus>incomplete</completionStatus>
   </objective>
   <objective>
      <identifier>urn:foo.com:id200309041234534</identifier>
      <completionStatus>completed</completionStatus>
   </objective>
</objectives>>
```

The W3C XML Schema definition language cannot specify whether a collection implemented with `xs:sequence` should be treated as a bag (unordered) or an array (ordered). However, it is possible in some cases to specify whether the items in the collection have to have different content, which is one of the characteristics of a set model. Implementers should refer to IEEE 1484.11.1–2004 to determine whether a particular element defined as `xs:sequence` with a single element defined in the sequence should be treated as a bag, a set, or an array. As stated in W3C XML Schema Part 2, "The fact that this specification does not define an order-relation for some datatype does not mean that some other application cannot treat that datatype as being ordered by imposing its own order relation."

Whenever an `xs:sequence` construct is defined as part of a type definition, the XSD defines an attribute named `collectionType`. The value of this attribute is fixed as one of `bag`, `array`, or `set`. XML Schema syntax does not allow the addition of this attribute for element sequences defined in an `xs:group` construct. The `collectionType` attribute has no effect on XML validation, but it may be useful for applications, because an XML Schema processor has to make the fixed attribute and its value available to applications even if they are not explicitly specified in the COCD XML instance. This Standard does not define any conformance requirements regarding the use or interpretation of the `collectionType` attribute.

An implementation might use the fixed `collectionType` attribute as follows:

- Instantiate an XML document that contains an element for which the `collectionType` attribute is defined in the XSD in an XML Schema processor environment.
- Get the `collectionType` attribute and its value by calling usual XML Schema processor methods. For example, using an XPath expression, such as `"./@collectionType"`, will access the attribute `collectionType` for the current element. The value can then be used to determine whether to treat the sequence of elements as ordered or unordered.

D.1.2 Uniqueness

Where possible, the standard XML Schema element `xs:unique` is used in the XSD to enforce uniqueness in collections that are defined as sets in the data model or that otherwise require uniqueness. These are

- 6.1.9 Interactions—Each interaction requires an identifier, and the identifier has to be unique within the context of the content object.
- 6.1.9.3 Objectives ID in an interaction record.
- The choices in a set of choices in 6.1.9.5 Correct responses and 6.1.9.7 Learner response for multiple choice interactions.
- 6.1.18 Objectives—Each objective requires an identifier, and the identifier has to be unique within the context of the content object.

This allows an XML Schema processor to enforce the uniqueness constraint automatically. However, validation of uniqueness for the other sets defined in the data model cannot be enforced automatically. It is not always possible to specify this constraint in the W3C XML Schema definition language. For example, in the set of set of choices for the correct response for the interaction type `multiple_choice`, uniqueness can be specified for the inner set but not for the outer set. Enabling the constraint for the outer set would require adding considerable complexity, such as additional arbitrary identifiers for the elements of the outer set, and still would not guarantee uniqueness, because the actual content of the elements cannot be inspected. The outer set elements would be guaranteed to have unique identifiers, but not that their content is unique.

D.1.3 Smallest permitted maximums

An SPM cannot be expressed in the W3C XML Schema definition language in any way that would allow automatic validation of an SPM constraint as defined in the data model. Therefore, the XSD always specifies `maxOccur="unbounded"` when the data model allows multiplicity, and does not set a `maxLength` attribute for other types for which an SPM is specified in the data model. Implementers should refer to the IEEE 1484.11.1–2004 to determine the SPM that applies to a particular element.

Where possible, the XSD defines an attribute named `spm`. This attribute is added to several elements and types defined in the XSD with a fixed value which is the SPM value. However, XML Schema syntax does not allow the addition of this attribute for element sequences defined in an `xs:group` construct nor is this attribute defined for string types that have to remain simple types, such as short and long identifiers. This attribute has no effect on XML validation, but it may be useful for applications, because an XML Schema processor has to make the fixed attribute and its value available to applications even if they are not explicitly specified in the COCD XML instance. This Standard does not define any conformance requirements regarding the use or interpretation of the `spm` attribute.

An implementation might use the fixed `spm` attribute as follows:

- Instantiate an XML document that contains an element for which the `spm` attribute is defined in the XSD in an XML Schema processor environment.
- Get the `spm` attribute and its value by calling usual XML Schema processor methods. For example, using an XPath expression, such as `"./@spm"`, will access the attribute `spm` for

the current element. The attribute value can then be used to compare the length of the value in the node with the SPM.

D.1.4 Machine-readable annotations in the XSD

The W3C XML Schema definition language allows the inclusion of annotations that are intended to be machine readable in an XSD by encapsulating such annotation in an `xs:appInfo` element inside an `xs:annotation` element. The XSD contains annotations with information about requirements of the data model for particular elements.

An implementation that is aware of this notational convention may be able to use an annotation to discover the SPM associated with an element when the XSD does not provide an SPM value as an `spm` attribute with a fixed value, or may be able to use the collection type when the XSD does not provide a `collectionType` attribute with a fixed value indicating the type of collection. This Standard does not define any conformance requirements regarding annotations or the existence, use, or interpretation of the data expressed in `appInfo` elements in the XSD. (To make the examples in this annex more readable, the annotations have been removed from most of the XSD fragments.)

The element names used within `appInfo` annotation elements are

- `spm`: This name denotes the SPM value for the element.
- `collectionType`: This name denotes the type of collection encoded using `xs:sequence`. The associated value is one of `bag`, `array`, or `set`.

Ideally, such elements would be defined in some additional namespace specific to the data model, but creating such a namespace is outside of the scope of this Standard. Also, the namespace would have to be provided as a separate document, and the XSD would not be usable without the document. Referencing an external, arbitrary namespace would complicate deploying and using the XSD. To avoid such complications but still allow a machine-readable representation in the spirit of the W3C XML Schema recommendations, the data-model information data is "hidden" within a comment within an `appInfo` element. To discover the SPM or collection type for an element defined in the XSD an implementation can

- Instantiate an XML fragment from the string contained in the `appInfo` element, if present, and extract the comment from that fragment;
- Instantiate an XML fragment from the string contained in the comment; and
- Get the element `spm` or `collectionType`, if present, and inspect its value.

D.2 Encoding of 6.1 Content object communication

The element specified in the data model as

```
content_object_communication : record
```

is represented in the XSD by the element `cocd` of type `cocdType`.

A conforming COCD XML instance is expected to implement a `cocd` element as the root element of the XML document or fragment that contains a content object communication record.

Every element in the complex type `cocdType` is optional and may appear at most one time in a COCD XML instance. Although the elements are listed in the XSD in the same order as in subclause 6.1 of IEEE 1484.11.1–2004, the elements may appear in any order.

```
<xs:complexType name="cocdType">
  <xs:all>
    <xs:element ref="commentsFromLearner" minOccurs="0"/>
    <xs:element ref="commentsFromLMS" minOccurs="0"/>
...
    <xs:element ref="timeLimitAction" minOccurs="0"/>
    <xs:element ref="totalTime" minOccurs="0"/>
  </xs:all>
</xs:complexType>
```

Every element in the `cocdType` type definition references an element defined at the top level of the XSD. These top-level elements appear in the XSD in the same order as in this type definition.

D.2.1 Encoding of 6.1.1 Comments from learner

This data-model element is implemented in the XSD as the element `commentsFromLearner`, which is a sequence of zero or more `commentFromLearner` elements. Each `commentFromLearner` element is of type `commentType`, which is an encoding of the comment type defined in subclause 6.2.1 of IEEE 1484.11.1–2004.

```
<xs:element name="commentsFromLearner">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="commentFromLearner" type="commentType"
        minOccurs="0" maxOccurs="unbounded">
  </xs:complexType>
</xs:element>
```

D.2.2 Encoding of 6.1.2 Comments from LMS

This data-model element is implemented in the XSD as the element `commentsFromLMS`, which is a sequence of zero or more `commentFromLMS` elements. Each `commentFromLMS` element is of type `commentType`, which is an implementation of the comment type defined in subclause 6.2.1 of IEEE 1484.11.1–2004.

```
<xs:element name="commentsFromLMS">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="commentFromLMS" type="commentType"
        minOccurs="0" maxOccurs="unbounded">
  </xs:complexType>
</xs:element>
```


D.2.3 Encoding of 6.1.3 Completion status

This data-model element is implemented in the XSD as the element `completionStatus` of type `completionStatusType`, which is an implementation of the completion status type defined in subclause 6.2.2 of IEEE 1484.11.1–2004.

```
<xs:element name="completionStatus" type="completionStatusType"/>
```

D.2.4 Encoding of 6.1.4 Completion threshold

This data-model element is implemented in the XSD as the element `completionThreshold` of type `progressMeasureType`, which is an implementation of the progress measure type defined in subclause 6.2.7 of IEEE 1484.11.1–2004.

```
<xs:element name="completionThreshold" type="progressMeasureType"/>
```

D.2.5 Encoding of 6.1.5 Credit

This data-model element is implemented in the XSD as the element `credit`, which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.5 of IEEE 1484.11.1–2004.

```
<xs:element name="credit">
  <xs:simpleType>
    <xs:restriction base="xs:token">
      <xs:enumeration value="credit"/>
      <xs:enumeration value="no_credit"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

D.2.6 Encoding of 6.1.6 Data model version

This data-model element is implemented in the XSD as the element `dataModelVersion` of type `literalString250Type`, which is a custom type defined in the XSD to avoid accidental modification of the string value if it contains white space. This type has the fixed attribute `spm` with a value of 250.

```
<xs:element name="dataModelVersion" type="literalString250Type"/>
```

D.2.7 Encoding of 6.1.7 Entry

This data-model element is implemented in the XSD as the element `entry`, which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.7 of IEEE 1484.11.1–2004.

```
<xs:element name="entry">
  <xs:simpleType>
    <xs:restriction base="xs:token">
      <xs:enumeration value="ab_initio"/>
      <xs:enumeration value="resume"/>
      <xs:enumeration value=""/>
    </xs:restriction>
  </xs:simpleType>
```



```
</xs:element>
```

NOTE—An empty string ("") is represented in a COCD XML instance as an empty element (e.g., `<cocd><entry/></cocd>`).

D.2.8 Encoding of 6.1.8 Exit

This data-model element is implemented in the XSD as the element `exit`, which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.8 of IEEE 1484.11.1–2004.

```
<xs:element name="exit">
  <xs:simpleType>
    <xs:restriction base="xs:token">
      <xs:enumeration value="logout"/>
      <xs:enumeration value="normal"/>
      <xs:enumeration value="suspend"/>
      <xs:enumeration value="timeout"/>
      <xs:enumeration value=""/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

NOTE—An empty string ("") is represented in a COCD XML instance as an empty element (e.g., `<cocd><exit/></cocd>`).

D.2.9 Encoding of 6.1.9 Interactions

This data model element is implemented in the XSD as a sequence of elements of type `interactionType`. The type `interactionsType` is defined as a collection of `interaction` elements, each of which represents an interaction record. Although the data model specifies that the collection is a bag of `interaction_type` records, the XSD has to use `xs:sequence` to allow multiplicity of the `interaction` element. In this case, `xs:sequence` should not be interpreted as implying any particular order.

```
<xs:complexType name="interactionsType">
  <xs:sequence>
    <xs:element name="interaction" type="interactionType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Implementation of interaction_type

This data-model structure is implemented in the XSD by the global type `interactionType`, which includes the required elements `identifier` and `type` and the optional elements `objectiveIds`, `timeStamp`, `correctResponses`, `weighting`, `learnerResponse`, `result`, `latency`, and `description`. If present, these elements may occur in any order. The elements `correctResponses` and `learnerResponse` are defined as global elements to facilitate possible reuse.

```
<xs:complexType name="interactionType">
  <xs:all>
```



```
        <xs:element name="identifier" type="longIdentifierType">
        <xs:element name="type" type="interactionTypeType"/>
        <xs:element name="objectiveIds" type="objectiveIdsType"
          minOccurs="0">
          <xs:unique name="uniqueInObjectivesIds">
            <xs:selector xpath="."/>
            <xs:field xpath="./objectiveId"/>
          </xs:unique>
        </xs:element>
        <xs:element name="timeStamp" type="dateTimeType" minOccurs="0"/>
        <xs:element name="correctResponses" type="correctResponsesType"
          minOccurs="0"/>
        <xs:element name="weighting" type="real7Type" minOccurs="0"/>
        <xs:element name="learnerResponse type"="learnerResponseType"
          minOccurs="0"/>
        <xs:element name="result" type="interactionResultType"
          minOccurs="0"/>
        <xs:element name="latency" type="timeIntervalType"
          minOccurs="0"/>
        <xs:element name="description" type="localizedString250Type"
          minOccurs="0"/>
      </xs:all>
    </xs:complexType>
```

Encoding of 6.1.9.1 ID

This data-model element is implemented in the XSD as the element `identifier` of type `longIdentifierType`.

```
    <xs:element name="identifier" type="longIdentifierType"/>
```

Encoding of 6.1.9.2 Type

This data-model element is implemented in the XSD as the element `type` of type `interactionTypeType`.

```
    <xs:element name="type" type="interactionTypeType"/>
```

Interaction type type

This data-model structure is implemented in the XSD as the global type `interactionTypeType`, which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.9.2 of IEEE 1484.11.1–2004.

```
    <xs:simpleType name="interactionTypeType">
      <xs:restriction base="xs:token">
        <xs:enumeration value="true_false"/>
        <xs:enumeration value="multiple_choice"/>
        <xs:enumeration value="fill_in"/>
        <xs:enumeration value="long_fill_in"/>
        <xs:enumeration value="likert"/>
        <xs:enumeration value="matching"/>
        <xs:enumeration value="performance"/>
        <xs:enumeration value="sequencing"/>
```



```
        <xs:enumeration value="numeric"/>
        <xs:enumeration value="other"/>
      </xs:restriction>
    </xs:simpleType>
```

Encoding of 6.1.9.3 Objectives ID

This data-model element is implemented in XSD as the element `ObjectiveIds` of type `objectiveIdsType`, which includes zero or more instances of the element `longIdenitifierType`. A uniqueness constraint is applied.

```
    <xs:element name="objectiveIds" type="objectiveIdsType"
      minOccurs="0">
      <xs:unique name="uniqueInObjectivesIds">
        <xs:selector xpath="./objectiveId"/>
        <xs:field xpath="."/>
      </xs:unique>
    </xs:element>
```

The global type `objectiveIdsType` is defined as a collection of `objectiveId` elements of type `longIdentifierType`.

```
    <xs:complexType name="objectiveIdsType">
      <xs:sequence>
        <xs:element name="objectiveId" type="longIdentifierType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
```

Encoding of 6.1.9.4 Time stamp

This data-model element is implemented in the XSD as the element `timestamp` of type `dateTimeType`.

```
    <xs:element name="timeStamp" type="dateTimeType" minOccurs="0"/>
```

Encoding of 6.1.9.5 Correct responses

This data-model element is implemented in the XSD as the element `correctResponses` of type `correctResponsesType`.

```
    <xs:element name="correctResponses" type="correctResponsesType"
      minOccurs="0"/>
```


Correct responses type

This global type implements the various `correct_responses` data structures defined in the data model to correspond to different types of interactions. Regardless of the complexity of the data structure for a type of interaction, the choice is implemented in the XSD by a global group. It is up to the COCD XML instance implementation to choose the correct group to match the interaction type. This correspondence cannot be expressed or validated using the W3C XML Schema definition language. Refer to the sample COCD XML instance (see Annex C) for examples showing how this type can be used in an actual COCD XML instance.

```
<xs:complexType name="correctResponsesType">
    <xs:choice>
    <xs:group ref="grpCorrectTrueFalse"/>
    <xs:group ref="grpCorrectMultipleChoice"/>
    <xs:group ref="grpCorrectFillIn"/>
    <xs:group ref="grpCorrectLongFillIn"/>
    <xs:group ref="grpCorrectLikert"/>
    <xs:group ref="grpCorrectMatching"/>
    <xs:group ref="grpCorrectPerformance"/>
    <xs:group ref="grpCorrectSequencing"/>
    <xs:group ref="grpCorrectNumeric"/>
    <xs:group ref="grpCorrectOther"/>
  </xs:choice>
</xs:complexType>
```

These groups are described in detail in D.6.

Encoding of 6.1.9.6 Weighting

This data-model element is implemented in the XSD as the element `weighting` of type `real7Type`.

```
<xs:element name="weighting" type="real7Type" minOccurs="0"/>
```

Encoding of 6.1.9.7 Learner response

This data-model element is implemented in the XSD as the element `learnerResponse` of type `learnerResponseType`.

```
<xs:element name="learnerResponse" type="learnerResponseType"
  minOccurs="0"/>
```

Learner response type

This global type implements the various `learner_response` data structures defined in the data model to correspond to different types of interaction types. Regardless of the complexity of the data structure for a type of interaction, the choice is implemented in the XSD as a global group. It is up to the COCD XML instance implementation to choose the correct group to match the interaction type. Refer to the sample COCD XML instance (see Annex C) for examples of how this type can typically be used in an actual COCD XML instance.

```
<xs:complexType name="learnerResponseType">
  <xs:choice>
```



```
        <xs:group ref="grpResponseTrueFalse"/>
        <xs:group ref="grpResponseMultipleChoice"/>
        <xs:group ref="grpResponseFillIn"/>
        <xs:group ref="grpResponseLongFillIn"/>
        <xs:group ref="grpResponseLikert"/>
        <xs:group ref="grpResponseMatching"/>
        <xs:group ref="grpResponsePerformance"/>
        <xs:group ref="grpResponseSequencing"/>
        <xs:group ref="grpResponseNumeric"/>
        <xs:group ref="grpResponseOther"/>
      </xs:choice>
    </xs:complexType>
```

These groups are described in detail in D.6.

Encoding of 6.1.9.8 Result

This data-model element is implemented in the XSD as the element `result` of type `interactionResultType`.

```
    <xs:element name="result" type="interactionResultType"
      minOccurs="0"/>
```

Interaction result type

This global type implements the data-model requirement that the value of `result` may be a token that corresponds to a permissible value defined by the data model or a numeric value. This element uses `xs:union` instead of `xs:choice` to avoid having to define subelements with arbitrary names in the XSD, which would force another layer of nesting. The `xs:union` construct allows only a single value to occur, and thus meets the requirement of the data model.

```
    <xs:simpleType name="interactionResultType">
      <xs:union memberTypes="real7Type interactionResultTokenType"/>
    </xs:simpleType>
    <xs:simpleType name="interactionResultTokenType">
      <xs:restriction base="xs:token">
        <xs:enumeration value="correct"/>
        <xs:enumeration value="incorrect"/>
        <xs:enumeration value="neutral"/>
        <xs:enumeration value="unanticipated"/>
      </xs:restriction>
    </xs:simpleType>
```

Encoding of 6.1.9.9 Latency

This data-model element is implemented in the XSD as the element `latency` of type `timeIntervalType`.

```
    <xs:element name="latency" type="timeIntervalType" minOccurs="0"/>
```


Encoding of 6.1.9.10 Description

This data-model element is implemented in the XSD as the element `description` of type `localizedString250Type`.

```
<xs:element name="description" type="localizedString250Type"
  minOccurs="0"/>
```

D.2.10 Encoding of 6.1.10 Launch data

This data-model element is implemented in the XSD as the element `launchData` of type `literalString4000Type` to avoid accidental modification of the string value if it contains white space.

```
<xs:element name="launchData" type="literalString4000Type"/>
```

D.2.11 Encoding of 6.1.11 Learner ID

This data-model element is implemented in the XSD as the element `learnerId` of type `longIdentifierType`.

```
<xs:element name="learnerId" type="longIdentifierType"/>
```

D.2.12 Encoding of 6.1.12 Learner name

This data-model element is implemented in the XSD as the element `learnerName` of type `localizedString250Type`.

```
<xs:element name="learnerName" type="localizedString250Type"/>
```

D.2.13 Encoding of 6.1.13 Learner preference data

This data-model element is implemented in the XSD as the element `learnerPreferenceData` of type `learnerPreferenceType`.

```
<xs:element name="learnerPreferenceData"
  type="learnerPreferenceType" />
```

The global type `learnerPreferenceType` implements the `learner_preference_type` defined in subclause 6.1.13 of IEEE 1484.11.1–2004. It contains four optional elements that may appear in any order, `audioLevel`, `language`, `deliverySpeed`, and `audioCaptioning`, which are described in more detail below.

```
<xs:element name="learnerPreferenceData">
  <xs:complexType>
    <xs:all>
      <xs:element name="audioLevel" minOccurs="0">
        ...
      </xs:element>
      <xs:element name="language" type="languageType"
        minOccurs="0"/>
      <xs:element name="deliverySpeed" minOccurs="0">
        ...
      </xs:element>
      <xs:element name="audioCaptioning" minOccurs="0">
```



```
            ...
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
```

Encoding of 6.1.13.1 Audio level

This data-model element is implemented in the XSD as the element `audioLevel`, which is defined inline as an optional element of `learnerPreferenceData`. It is based on the type `real7Type`, with a restriction that its value is greater than or equal to zero.

```
    <xs:element name="audioLevel" minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="real7Type ">
          <xs:minInclusive value="0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
```

Encoding of 6.1.13.2 Language

This data-model element is implemented in the XSD as the element `language`, which is defined inline as an optional element of `learnerPreferenceData`. It is of type `languageType`.

```
    <xs:element name="language" type="languageType" minOccurs="0"/>
```

Encoding of 6.1.13.3 Delivery speed

This data-model element is implemented in the XSD as the element `deliverySpeed`, which is defined inline as an optional element of `learnerPreferenceData.` It is based on the type `real7Type`, with a restriction that its value is greater than or equal to zero.

```
    <xs:element name="deliverySpeed" minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="real7Type">
          <xs:minInclusive value="0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
```

Encoding of 6.1.13.4 Audio captioning

This data-model element is implemented in the XSD as the element `audioCaptioning`, which is defined inline as an optional element of `learnerPreferenceData`. It is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.13.4 of IEEE 1484.11.1–2004.

```
    <xs:element name="audioCaptioning" minOccurs="0">
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="off"/>
          <xs:enumeration value="no_change"/>
          <xs:enumeration value="on"/>
```



```
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
```

D.2.14 Encoding of 6.1.14 Lesson status

This data-model element is implemented in the XSD as the element `lessonStatus` of type `legacyStatusType`.

```
  <xs:element name="lessonStatus" type="legacyStatusType"/>
```

D.2.15 Encoding of 6.1.15 Location

This data-model element is implemented in the XSD as the element `location` of type `literalString1000Type` to avoid accidental modification of the string value if it contains white space.

```
  <xs:element name="location" type="literalString1000Type"/>
```

D.2.16 Encoding of 6.1.16 Max time allowed

This data-model element is implemented in the XSD as the element `maxTimeAllowed` of type `timeIntervalType`.

```
    <xs:element name="maxTimeAllowed" type="timeIntervalType"/>
```

D.2.17 Encoding of 6.1.17 Mode

This data-model element is implemented in the XSD as the element `mode`, which is defined as an enumerated type with token values that correspond to the permissible values defined in sub-clause 6.1.17 of IEEE 1484.11.1–2004.

```
  <xs:element name="mode">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="browse"/>
        <xs:enumeration value="normal"/>
        <xs:enumeration value="review"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
```

D.2.18 Encoding of 6.1.18 Objectives

This data-model element is implemented in the XSD as a sequence of `objective` elements of type `objectiveType`. Although the data model specifies that this is a set of `objective`, the XSD has to use `xs:sequence` to allow multiplicity of the `objective` element. In this case, `xs:sequence` should not be interpreted as implying any particular order. The value of the mandatory `identifier` element within each `objective` element in this collection has to be unique.

```
  <xs:element name="objectives" type="objectivesType">
    <xs:unique name="uniqueInSetOfObjectives">
```



```
        <xs:selector xpath="./objective"/>
        <xs:field xpath="identifier"/>
      </xs:unique>
    </xs:element>
```

The global type `objectivesType` implements a collection of `objective` elements of type `objectiveType`.

```
    <xs:complexType name="objectivesType">
      <xs:sequence>
        <xs:element name="objective" type="objectiveType" minOccurs="0"
          maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
      <xs:attribute name="spm" fixed="100"/>
    </xs:complexType>
```

Objective type

This data-model structure is implemented in the XSD as the global type `objectiveType`, which includes the required element `identifier` and the optional elements `score`, `status`, `progressMeasure`, `completionStatus`, `successStatus`, and `description`. If present, these elements may occur in any order.

```
    <xs:complexType name="objectiveType">
      <xs:all>
        <xs:element name="identifier" type="longIdentifierType"/>
        <xs:element name="score" type="scoreType" minOccurs="0"/>
        <xs:element name="status" type="legacyStatusType"
          minOccurs="0"/>
        <xs:element name="progressMeasure"
          type="progressMeasureType" minOccurs="0"/>
        <xs:element name="completionStatus"
          type="completionStatusType" minOccurs="0"/>
        <xs:element name="successStatus" type="successStatusType"
          minOccurs="0"/>
        <xs:element name="description" type="localizedString250Type"
          minOccurs="0">
        </xs:element>
      </xs:all>
    </xs:complexType>
```

D.2.19 Encoding of 6.1.19 Progress measure

This data-model element is implemented in the XSD as the element `progressMeasure` of type `progressMeasureType`.

```
    <xs:element name="progressMeasure" type="progressMeasureType"/>
```

D.2.20 Encoding of 6.1.20 Raw passing score

This data-model element is implemented in the XSD as the element `rawPassingScore` of type `real7Type`.

```
    <xs:element name="rawPassingScore" type="real7Type"/>
```


D.2.21 Encoding of 6.1.21 Scaled passing score

This data-model element is implemented in the XSD as the element `scaledPassingScore` of type `scaledScoreType`.

```
<xs:element name="scaledPassingScore" type="scaledScoreType"/>
```

D.2.22 Encoding of 6.1.22 Score

This data-model element is implemented in the XSD as the element `score` of type `scoreType`, which is a complex type composed of several elements.

```
<xs:element name="score" type="scoreType"/>
```

D.2.23 Encoding of 6.1.23 Session time

This data-model element is implemented in the XSD as the element `sessionTime` of type `timeIntervalType`.

```
<xs:element name="sessionTime" type="timeIntervalType"/>
```

D.2.24 Encoding of 6.1.24 Success status

This data-model element is implemented in the XSD as the element `successStatus` of type `successStatusType`.

```
<xs:element name="successStatus" type="successStatusType"/>
```

D.2.25 Encoding of 6.1.25 Suspend data

This data-model element is implemented in the XSD as the element `suspendData` of type `literalString4000Type` to avoid accidental modification of the string value if it contains white space.

```
<xs:element name="suspendData" type="literalString4000Type"/>
```

D.2.26 Encoding of 6.1.26 Time limit action

This data-model element is implemented in the XSD as the element `timeLimitAction`, which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.26 of IEEE 1484.11.1–2004.

```
<xs:element name="timeLimitAction">
  <xs:simpleType>
    <xs:restriction base="xs:token">
      <xs:enumeration value="continue_message"/>
      <xs:enumeration value="continue_no_message"/>
      <xs:enumeration value="exit_message"/>
      <xs:enumeration value="exit_no_message"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

D.2.27 Encoding of 6.1.27 Total time

This data-model element is implemented in the XSD as the element `totalTime` of type `timeIntervalType`.

```
<xs:element name="totalTime" type="timeIntervalType"/>
```

D.3 Encoding of 6.2 Auxiliary data types

The encodings of the data-model auxiliary data types are discussed in D.3.1 – D.3.10.

D.3.1 Implementation of 6.2.1 Comment type

This data-model type is implemented in the XSD as the global type `commentType`, which includes the required element `comment` and the optional elements `location` and `timeStamp`. If present, these elements may occur in any order.

```
<xs:complexType name="commentType">
  <xs:all>
    <xs:element name="comment" type="localizedString4000Type"/>
    <xs:element name="location" type="literalString1000Type"
      minOccurs="0"/>
    <xs:element name="timeStamp" type="dateTimeType" minOccurs="0"/>
  </xs:all>
</xs:complexType>
```

D.3.2 Implementation of 6.2.2 Completion status type

This data-model type is implemented in the XSD as the global type `completionStatusType`, which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.2.2 of IEEE 1484.11.1–2004.

```
<xs:simpleType name="completionStatusType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="completed"/>
    <xs:enumeration value="incomplete"/>
    <xs:enumeration value="not_attempted"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>
```

D.3.3 Implementation of 6.2.3 Date time type

This data-model type is implemented in the XSD as the global type `dateTimeType`, which is based on the XML Schema primitive data type `dateTime`. The `dateTime` type is a conforming implementation of the requirements specified defined in subclause 6.2.3 of IEEE 1484.11.1–2004.

```
<xs:simpleType name="dateTimeType">
  <xs:restriction base="xs:dateTime"/>
</xs:simpleType>
```

The format for `dateTime` is defined by the following pattern:


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

where anything enclosed in square brackets is optional.

D.3.4 Implementation of 6.2.4 Language type

This data-model type is implemented in the XSD as the global type `languageType`, which is based on the XML Schema derived data type `language`. The `language` type is a conforming implementation of the requirements specified defined in subclause 6.2.4 of IEEE 1484.11.1–2004 except for the specification of an SPM, which is represented in an annotation in the XSD.

```
<xs:simpleType name="languageType">
  <xs:restriction base="xs:language"/>
</xs:simpleType>
```

D.3.5 Implementation of 6.2.5 Localized string type

This data-model type is implemented in the XSD as the global type `localizedStringType`, which is based on the global type `literalStringType`. That base type is used to avoid accidental modification of the string value if it contains white space. The type `localizedStringType` is used as an abstract base type for two other types, `localizedString250Type` and `localizedString4000Type`, each of which has a different fixed value for an `spm` attribute.

```
<xs:complexType name="localizedStringType" abstract="true">
  <xs:simpleContent>
    <xs:extension base="literalStringType">
      <xs:attribute name="lang" type="languageType"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

D.3.6 Implementation of 6.2.6 Long identifier type

This data-model type is implemented in the XSD as the global type `longIdentifierType`, which is based on the XML Schema primitive data type `anyURI`. The `anyURI` type is a conforming implementation of the requirements defined in subclause 6.2.6 of IEEE 1484.11.1–2004 except for the specification of an SPM, which is represented in an annotation in the XSD. Because this type is used where a simple type is required, no `spm` attribute can be attached to it.

```
<xs:simpleType name="longIdentifierType">
  <xs:restriction base="xs:anyURI">
  </xs:restriction>
</xs:simpleType>
```

D.3.7 Implementation of 6.2.7 Progress measure type

This data-model type is implemented in the XSD as the global type `progressMeasureType`, which is based on the global type `real7Type`, which, in turn, is based on the XML Schema primitive data type `decimal`.

```
<xs:simpleType name="progressMeasureType">
```



```
      <xs:restriction base="real7Type">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
```

D.3.8 Implementation of 6.2.8 Score type

This data-model type is implemented in the XSD as the global type `scoreType`, which includes the optional elements `scaled`, `max`, `min`, and `raw`. If present, these elements may occur in any order.

```
    <xs:complexType name="scoreType">
      <xs:all>
        <xs:element name="scaled" type="scaledScoreType"/>
        <xs:element name="max" type="real7Type" minOccurs="0"/>
        <xs:element name="min" type="real7Type" minOccurs="0"/>
        <xs:element name="raw" type="real7Type" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
```

D.3.9 Implementation of 6.2.9 Short identifier type

This data-model type is implemented in the XSD as the global type `shortIdentifierType`, which is based on the XML Schema primitive data type `anyURI`. The `anyURI` type is a conforming implementation of the requirements defined in subclause 6.2.9 of IEEE 1484.11.1–2004 except for the specification of an SPM, which is represented in an annotation in the XSD. Because this type is used where a simple type is required, no `spm` attribute can be attached to it.

```
    <xs:simpleType name="shortIdentifierType">
      <xs:restriction base="xs:anyURI">
    </xs:simpleType>
```

D.3.10 Implementation of 6.2.10 Success status type

This data-model type is implemented in the XSD as the global type `successStatusType` which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.2.10 of IEEE 1484.11.1–2004.

```
    <xs:simpleType name="successStatusType">
      <xs:restriction base="xs:token">
        <xs:enumeration value="failed"/>
        <xs:enumeration value="passed"/>
        <xs:enumeration value="unknown"/>
      </xs:restriction>
    </xs:simpleType>
```

D.4 Implementation of other documented data types

The implementations of other documented data types are discussed in D.4.1 – D.4.2.

D.4.1 Implementation of real(10,7)

An explanation of this type is provided in subclause B.1 of IEEE 1484.11.1–2004. This data-model type is implemented in the XSD as the global type `real7Type`, which is based on the XML Schema primitive data type `decimal`.

```
<xs:simpleType name="real7Type">
  <xs:restriction base="xs:decimal"/>
</xs:simpleType>
```

By neither restricting the number of fraction digits nor the number of digits in total, the XSD provides the required precision in all cases of practical interest.

D.4.2 Implementation of the time interval data type

An explanation of this type is provided in subclause B.2 of IEEE 1484.11.1–2004. This data-model type is implemented in the XSD as the global type `timeIntervalType`, which is based on the XML Schema primitive data type `duration`. The `duration` type is a conforming implementation of the requirements specified in the data model.

```
<xs:simpleType name="timeIntervalType">
  <xs:restriction base="xs:duration"/>
</xs:simpleType>
```

This type definition in the XSD does not enforce a restriction on the number of decimal digits for the seconds part of the duration expression.

D.5 Other global types defined in the XSD

The types in D.5.1 – D.5.3 are used by more than one element in the XSD or as building blocks in the definition of other XSD types.

D.5.1 Literal string type

This type is defined for strings in which white space should not be modified by an XML implementation. The type `literalStringType` is used as an abstract base type for three other types, `literalString250Type`, `literalString1000Type`, and `literalString4000Type`, each of which has a different fixed value for a `spm` attribute.

```
<xs:simpleType name="literalStringType">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="preserve"/>
  </xs:restriction>
</xs:simpleType>
```


D.5.2 Lesson status type

This data-model type is defined as the global type `obsoleteStatusType` which is defined as an enumerated type with token values that correspond to the permissible values defined in subclause 6.1.14 of IEEE 1484.11.1–2004.

```
<xs:simpleType name="obsoleteStatusType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="browsed"/>
    <xs:enumeration value="completed"/>
    <xs:enumeration value="failed"/>
    <xs:enumeration value="incomplete"/>
    <xs:enumeration value="not_attempted"/>
    <xs:enumeration value="passed"/>
  </xs:restriction>
</xs:simpleType>
```

D.5.3 Scaled score type

This data-model type is defined as a numeric type based on `real7Type` with the range constraints defined in subclause 6.1.21 of IEEE 1484.11.1–2004.

```
<xs:simpleType name="scaledScoreType">
  <xs:restriction base="real7Type">
    <xs:minInclusive value="-1"/>
    <xs:maxInclusive value="1"/>
  </xs:restriction>
</xs:simpleType>
```

D.5.4 Literal string type – XML specific

This type is not defined explicitly in IEEE 1484.11.1–2004, but it is implicit in references to the ISO 11404 [B2] characterstring data type. Because XML Schema processors may modify the white space in a string value, it is necessary to specify in the XSD that the values for various string-based elements defined in subclause 6.1. of IEEE 1484.11.1–2004 cannot be modified. This is done by defining the type `literalStringType` that specifies that white space must be preserved. This type is used as the base type to define the type `localizedStringType` as well as the three types `literalString250Type`, `literalString1000Type` and `literalString4000Type` with fixed `spm` attribute values of `250`, `1000` and `4000`, respectively.

```
<xs:simpleType name="literalStringType">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="preserve"/>
  </xs:restriction>
</xs:simpleType>
```


D.6 Elements and groups used to implement response data

The globally defined, reusable, XML elements and groups discussed in D.6.1 – D.6.21 are used in the XSD as parts of the definitions for the complex, data-model types for correct responses and learner response. They are listed in alphabetic order. These XSD fragments are easier to understand by looking at the sample COCD XML instance (see Annex C).

D.6.1 Correct responses for fill-in

In this group definition, each `fillMatches` element is a sequence of predefined matching strings. The use of `xs:sequence` is required to allow more than one instance of a set of `fillMatches`. It should not be interpreted as implying any particular order. However, the order of the `matchText` elements within the `xs:sequence` for each `fillMatches` element is significant. The W3C XML Schema definition language has no provision to express this difference of interpretation of `xs:sequence`. Implementations should be aware of the data-model requirements.

```
<xs:group name="grpCorrectFillIn">
  <xs:sequence>
    <xs:element name="fillMatches" minOccurs="0"
      maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="matchText"
            type="localizedString250Type" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="caseMatters"
          type="trueFalseType" use="optional" default="false"/>
        <xs:attribute name="orderMatters" type="trueFalseType"
          use="optional" default="true"/>
        <xs:attribute name="collectionType" fixed="array"/>
        <xs:attribute name="spm" fixed="10"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>
```

D.6.2 Correct responses for likert

In this group definition, the likert choice is a single, optional identifier.

```
<xs:group name="grpCorrectLikert">
  <xs:sequence>
    <xs:element name="choice" type="shortIdentifierType"
      minOccurs="0"/>
  </xs:sequence>
</xs:group>
```


D.6.3 Correct responses for long fill-in

In this group definition, each `matchText` element is a single, predefined, matching string with an optional attribute specifying whether case matters for this match. The use of `xs:sequence` is required to allow more than one instance of a set of `matchText`. It should not be interpreted as implying any particular order.

```
<xs:group name="grpCorrectLongFillIn">
  <xs:sequence>
    <xs:element name="matchText" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="localizedString4000Type">
            <xs:attribute name="caseMatters"
              type="trueFalseType" use="optional" default="false"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>
```

D.6.4 Correct responses for matching

In this group definition, each `matchPattern` element of type `responseMatchingType` represents a set of matching pairs. The use of `xs:sequence` is required to allow more than one instance of `matchPattern`. It should not be interpreted as implying any particular order.

```
<xs:group name="grpCorrectMatching">
  <xs:sequence>
    <xs:element name="matchPattern" type="matchingPairsType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
```

The global type `matchingPairsType` implements a match pattern, which is a collection of matching pairs. No order is implied. Pairs need not be unique.

```
<xs:complexType name="matchingPairsType">
  <xs:sequence>
    <xs:element name="pair" type="matchingPairType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The global type `matchingPairType` implements a single matching pair, which is defined as an empty element with `source` and `target` attributes.

```
<xs:complexType name="matchingPairType">
  <xs:attribute name="source" type="shortIdentifierType"/>
  <xs:attribute name="target" type="shortIdentifierType"/>
</xs:complexType>
```


D.6.5 Correct responses for multiple choice

This element is defined in the data model as set of sets of short identifiers, each of which represents a choice. The outer set is a collection of `choices` elements. The use of `xs:sequence` is required to allow more than one instance of a set of choices. It should not be interpreted as implying any particular order.

```
<xs:group name="grpCorrectMultipleChoice">
  <xs:sequence>
    <xs:element ref="choices" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

The global element `choices` is of type `bagOfChoiceTypes`, which is a collection of `choice` elements. The use of `xs:unique` is required to constrain the collection to a set of unique `choice` elements.

```
<xs:element name="choices" type="bagOfChoicesType">
  <xs:unique name="uniqueInChoicesIds">
    <xs:selector xpath="./choice"/>
    <xs:field xpath="."/>
  </xs:unique>
  </xs:element>
```

The type `bagOfChoiceTypes` is a collection of `choice` elements. The use of `xs:sequence` should not be interpreted as implying any particular order.

```
<xs:complexType name="bagOfChoicesType">
  <xs:sequence>
    <xs:element name="choice" type="shortIdentifierType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

D.6.6 Correct responses for numeric

This group definition contains two optional elements that specify the `min` and `max` values for the correct response.

```
<xs:group name="grpCorrectNumeric">
  <xs:sequence>
    <xs:element name="min" type="real7Type" minOccurs="0"/>
    <xs:element name="max" type="real7Type" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

D.6.7 Correct responses for other

This group definition contains a generic, literal-string element.

```
<xs:group name="grpCorrectOther">
  <xs:sequence>
    <xs:element name="correctOther" type="literalString4000Type"/>
  </xs:sequence>
</xs:group>
```

D.6.8 Correct responses for performance

In this group definition, each `performancePattern` element of type `correctPerformancePatternType` represents a predefined sequence of steps with associated data. The use of `xs:sequence` is required to allow more than one instance of `performancePattern`. It should not be interpreted as implying any particular order.

```
<xs:group name="grpCorrectPerformance">
  <xs:sequence>
    <xs:element name="performancePattern"
      type="correctPerformancePatternType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
```

The global type `correctPerformancePatternType` specifies the encoding of a single performance response pattern, which is a sequence of steps with an optional, additional, literal or numeric, answer element specified for each step. Each step is identified by the value of the element `stepName`. The order of steps is significant.

```
<xs:complexType name="correctPerformancePatternType">
  <xs:sequence>
    <xs:element name="step" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:all>
          <xs:element name="stepName" type="shortIdentifierType"
            minOccurs="0"/>
          <xs:element name="stepAnswer" minOccurs="0">
            <xs:complexType>
              <xs:choice>
                <xs:element name="literal"
                  type="literalString250Type" minOccurs="0"/>
                <xs:element name="numeric" minOccurs="0">
                  <xs:complexType>
                    <xs:attribute name="min" type="real7Type"/>
                    <xs:attribute name="max" type="real7Type"/>
                  </xs:complexType>
                </xs:element>
              </xs:choice>
            </xs:complexType>
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="orderMatters" type="trueFalseType"
    use="optional" default="true"/>
  <xs:attribute name="collectionType" fixed="array"/>
  <xs:attribute name="spm" fixed="250"/>
</xs:complexType>
```


D.6.9 Correct responses for sequencing

In this group definition, each `stepSequence` element of type `stepSequenceType` represents a predefined sequence of steps. The use of `xs:sequence` is required to allow more than one instance of `stepSequence`. It should not be interpreted as implying any particular order.

```
<xs:group name="grpCorrectSequencing">
  <xs:sequence>
    <xs:element name="stepSequence" type="stepSequenceType"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
```

The type `stepSequenceType` represents a sequence of steps. The use of `xs:sequence` should be interpreted here as implying a specific order.

```
<xs:complexType name="stepSequenceType">
  <xs:sequence>
    <xs:element name="step" type="shortIdentifierType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="collectionType" fixed="array"/>
  <xs:attribute name="spm" fixed="36"/>
</xs:complexType>
```

D.6.10 Correct responses for true false

In this group definition, the true–false choice is a token with the value `true` or the value `false`.

```
<xs:group name="grpCorrectTrueFalse">
  <xs:sequence>
    <xs:element name="trueOrFalse" type="trueFalseType"/>
  </xs:sequence>
```

The global type `trueFalseType` implements true and false options for the interaction type `true_false` specified in subclauses 6.1.9.5 and 6.1.9.7 of IEEE 1484.11.1–2004. This global type is also used for tokens for various other Boolean elements.

```
<xs:simpleType name="trueFalseType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="true"/>
    <xs:enumeration value="false"/>
  </xs:restriction>
</xs:simpleType>
```

D.6.11 Learner response for fill-In

In this group definition, the learner response consists of a sequence of zero or more `fillString` elements of type `localizedString250Type`.

```
<xs:group name="grpResponseFillIn">
  <xs:sequence>
    <xs:element name="fillString" type="localizedString250Type"
      minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
```



```
  </xs:group>
```

D.6.12 Learner response for likert

In this group definition, the learner response is a single, optional identifier.

```
  <xs:group name="grpResponseLikert">
    <xs:sequence>
      <xs:element name="choice" type="shortIdentifierType"
        minOccurs="0"/>
    </xs:sequence>
  </xs:group>
```

D.6.13 Learner response for long fill-in

In this group definition, the learner response is a single localized string.

```
  <xs:group name="grpResponseLongFillIn">
    <xs:sequence>
      <xs:element name="longFillString" type="localizedString4000Type"
        minOccurs="0"/>
    </xs:sequence>
  </xs:group>
```

D.6.14 Learner response for matching

In this group definition, the learner response is a match pattern, which is a collection of matching pairs. No order is implied. Pairs need not be unique.

```
  <xs:group name="grpResponseMatching">
    <xs:sequence>
      <xs:element name="matchPattern" type="matchingPairsType"/>
    </xs:sequence>
  </xs:group>
```

D.6.15 Learner response for multiple choice

In this group definition, the learner response is a list of identifiers, each of which specifies one choice. No order should be implied.

```
  <xs:group name="grpResponseMultipleChoice">
    <xs:sequence>
      <xs:element name="choices" type="setOfChoicesType"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseNumeric">
    <xs:sequence>
      <xs:element name="number" type="real7Type" minOccurs="0"/>
    </xs:sequence>
  </xs:group>
  <xs:group name="grpResponseOther">
    <xs:sequence>
      <xs:element name="responseOther" type="literalString4000Type"/>
    </xs:sequence>
  </xs:group>
```


D.6.16 Learner response for numeric

In this group definition, the learner response is a single numeric value.

```
<xs:group name="grpResponseNumeric">
  <xs:sequence>
    <xs:element name="number" type="real7Type" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

D.6.17 Learner response for other

In this group definition, the learner response is a single literal string.

```
<xs:group name="grpResponseOther">
  <xs:sequence>
    <xs:element name="responseOther" type="literalString4000Type"
      minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

D.6.18 Learner response for performance

In this group definition, the learner response is a single performance pattern that represents the actual sequence of steps with associated data.

```
<xs:group name="grpResponsePerformance">
  <xs:sequence>
    <xs:element name="step" type="learnerPerformanceStepType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:group>
```

The type `learnerPerformanceStepType` implements the learner response elements for a single performance step.

```
<xs:complexType name="learnerPerformanceStepType">
  <xs:all>
    <xs:element name="stepName" type="shortIdentifierType"
      minOccurs="0"/>
    <xs:element name="stepAnswer" minOccurs="0">
      <xs:complexType>
        <xs:choice>
          <xs:element name="literal" type="literalString250Type"
            minOccurs="0"/>
          <xs:element name="numeric" type="real7Type"
            minOccurs="0"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>
```


D.6.19 Learner response for sequencing

In this group definition, the learner response is a list of steps identifiers.

```
<xs:group name="grpResponseSequencing">
  <xs:sequence>
    <xs:element name="steps" type="stepSequenceType" minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

D.6.20 Learner response for true false

In this group definition, the learner response is a sequence of zero or more step identifiers.

```
<xs:group name="grpResponseTrueFalse">
  <xs:sequence>
    <xs:element name="trueOrFalse" type="trueFalseType"
      minOccurs="0"/>
  </xs:sequence>
</xs:group>
```

D.6.21 Matching response type

This type implements a set of matching pairs. No particular order is implied.

```
<xs:complexType name="responseMatchingType">
  <xs:sequence>
    <xs:element name="pair" type="matchingPairType"
      maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
```


Annex E

(informative)

Internet availability of the XSD file and example instance

The XSD file in Annex B and the example instance in Annex C are available on the World Wide Web at the following URL:

– http://standards.ieee.org/reading/ieee/downloads/1484.11.3-2005/

The XSD file is available for downloading and for direct inclusion in applications.

