

# **IEEE 1484.11.1, Draft 5**

## **Draft Standard for Learning Technology— Data Model for Content Object Communication**

Sponsored by the Learning Technology Standards Committee  
of the IEEE Computer Society

Copyright © 2004 by the Institute of Electrical and Electronics Engineers, Inc.  
Three Park Avenue  
New York, NY 10016-5997, USA

All rights reserved. This document is an unapproved draft of a proposed IEEE Standard. As such, this document is subject to change. **USE AT YOUR OWN RISK!** Because this is an unapproved draft, this document must not be utilized for any conformance/compliance purposes. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only. Prior to submitting this document to another standards development organization for standardization activities, permission must first be obtained from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department. Other entities seeking permission to reproduce this document, in whole or in part, must obtain permission from the Manager, Standards Licensing and Contracts, IEEE Standards Activities Department.

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

### **Abstract:**

This Standard describes a data model to support the interchange of data elements and their values between a content object and a runtime service (RTS). It is based on a current industry practice called "CMI—Computer Managed Instruction." The work upon which this Standard is based was developed to support a client/server environment in which a learning technology system, generically called a learning management system (LMS), delivers digital content, called content objects, to learners. The data model supports learner data and preferences, interactions, objectives, content-object entry, exit, and status information, time parameters, and scores.

**Keywords:**

content object, computer managed instruction, CMI, data model, interoperability, learning content, learning management system, LMS, runtime service, RTS.

*[NOTE: Information about IEEE LTSC P1484.11 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 P1484.11.1, Draft Standard for Learning Technology—Data Model for Content Object Communication.)

This Standard describes a data model to support the interchange of agreed upon data elements and their values between a learning-related content object and a runtime service (RTS) used to support learning management.

**Participants**

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

Tyde Richards, <i>Chair</i> Jack Hyde, <i>Chair (December, 1998 – March, 2001)</i> Scott Lewis, <i>Technical Editor</i>		
Mitchell Bonnett	Tom King	Claude Ostyn
Frank Farance	Rolf Lindner	Daniel Rehak
Mike Fore	Kiyoshi Nakabayashi	Robby Robson
Leonard Greenberg	Boyd Nielsen	Schawn Thropp

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

## Contents

<b>1. Overview.....</b>	<b>1</b>
1.1 Scope .....	1
1.2 Purpose .....	1
<b>2. Normative references.....</b>	<b>2</b>
<b>3. Definitions.....</b>	<b>2</b>
3.1 Abbreviations and acronyms.....	3
<b>4. Conformance .....</b>	<b>4</b>
4.1 Data instances .....	4
4.2 Sending implementations .....	4
4.3 Receiving implementations.....	4
4.4 Repository implementations.....	4
4.5 Implementation-defined values.....	4
4.6 Smallest permitted maximum values.....	4
<b>5. Conceptual model (informative) .....</b>	<b>5</b>
<b>6. Data model .....</b>	<b>6</b>
6.1 Content object communication .....	7
6.1.1 Comments from learner .....	8
6.1.2 Comments from LMS .....	9
6.1.3 Completion status .....	9
6.1.4 Completion threshold.....	10
6.1.5 Credit.....	10
6.1.6 Data model version.....	10
6.1.7 Entry.....	11
6.1.8 Exit .....	11
6.1.9 Interactions.....	12
6.1.10 Launch data.....	25
6.1.11 Learner ID.....	25
6.1.12 Learner name .....	25
6.1.13 Learner preference data.....	26
6.1.14 Lesson status.....	28
6.1.15 Location .....	28
6.1.16 Max time allowed.....	29
6.1.17 Mode .....	29
6.1.18 Objectives .....	30
6.1.19 Progress measure.....	34
6.1.20 Raw passing score .....	34
6.1.21 Scaled passing score.....	34
6.1.22 Score.....	35
6.1.23 Session time.....	35
6.1.24 Success status .....	36
6.1.25 Suspend data .....	36
6.1.26 Time limit action .....	37
6.1.27 Total time.....	37
6.2 Auxiliary data types.....	38

6.2.1 Comment type .....	38
6.2.2 Completion status type.....	39
6.2.3 Date time type.....	40
6.2.4 Language type.....	40
6.2.5 Localized string type .....	41
6.2.6 Long identifier type.....	42
6.2.7 Progress measure type.....	43
6.2.8 Score type .....	43
6.2.9 Short identifier type.....	45
6.2.10 Success status type .....	45
<b>Annex A (informative) Bibliography .....</b>	<b>46</b>
<b>Annex B (informative) Understanding the ISO/IEC 11404:1996 real and time interval data types definitions used in this Standard.....</b>	<b>47</b>
B.1 Real data type .....	47
B.2 Time interval data type .....	47
<b>Annex C (informative) ISO 8601:2000 representation of the date time types .....</b>	<b>49</b>

# Draft Standard for Learning Technology— 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 describes a data model to support the interchange of agreed upon data elements and their values between a learning-related content object and a runtime service (RTS) used to support learning management. This Standard does not specify the means of communication between a content object and an RTS nor how any component of a learning environment shall behave in response to receiving data in the form specified. This Standard is based on a related data model defined in "Computer Managed Instruction (CMI) Guidelines For Interoperability," version 3.5 [B1]<sup>1</sup> by the Aviation Industry CBT Committee (AICC). To balance the need to support existing implementations with the need to make technical corrections and support emerging practice, this Standard selectively includes those data elements from the CMI specification that are commonly implemented; renames some data elements taken from the CMI specification to clarify their intended meaning; modifies the data types of data elements taken from the CMI specification to reflect ISO standard data types and internationalization requirements; removes some organizational structures used in the CMI specification to group data elements that are specific to the AICC community of practice and not generally applicable; and introduces some data elements not present in the CMI specification to correct known technical deficiencies in data elements taken from that specification.

### 1.2 Purpose

There is widespread acknowledgement that the data model for content object communication defined in the AICC "Computer Managed Instruction (CMI) Guidelines for Interoperability", version 3.5 [B1], has broad applicability to systems used for learning management. The purpose of this Standard is to build consensus around, resolve ambiguities in, and correct defects in the AICC data model for the data exchanged between learning-related content objects and an RTS used to support learning management.

---

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

## 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.

IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax.

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

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

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

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

ISO/IEC 646:1991, Information technology – ISO 7-bit coded character set for information interchange.

ISO/IEC 10646–1, Information technology – Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane.

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

## 3. Definitions

For purposes of this Standard, the following terms and definitions apply. IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [B2], 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 HTML page with an embedded video clip and an ECMAScript component written in accordance with IEEE 1484.11.2–2003, *Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication* [B3].

**implementation defined (adj.):** An indication that the implementation provider shall define and document the requirements for correct program constructs and correct data of a value or behavior. When the value or behavior in the implementation is designed to be variable or customizable on each instantiation of the system, the implementation provider shall document the nature and permissible ranges of this variation.

**interaction:** A recognized and recordable input or group of inputs from a learner to a content object.

**launch (v.):** To cause a content object to be delivered to a learner.

**learner:** An individual engaged with a learning technology system in order to acquire knowledge or skills.

**learner attempt:** A tracked effort by a learner to satisfy the requirements of a learning activity that uses a content object. A learner attempt may span one or more learner sessions and may be suspended between learner sessions. *See also:* **learner session**.

NOTE—The learner attempt begins with the beginning of the first learner session and continues until the learning activity terminates.

**learner session:** An uninterrupted period of time during which a learner is accessing a content object. *See also:* **learner attempt**.

**learning management system (LMS):** A computer system that may include the capabilities to register learners, schedule learning resources, control and guide the learning process, analyze and report learner performance, and schedule and track learners. *See also:* **runtime service**.

NOTE—Some implementations of learning management systems also have the ability to launch and deliver content. For this Standard, these capabilities are known as a runtime service.

**runtime service (RTS):** Software that controls the execution and delivery of learning content and that may provide services such as resource allocation, scheduling, input-output control, and data management. *See also:* **learning management system**.

**score:** A numerical value or a point on a descriptive scale. A score may be the result of a learner assessment.

### 3.1 Abbreviations and acronyms

AICC: Aviation Industry CBT Committee

CMI: computer managed instruction

IANA: Internet Assigned Numbers Authority

LMS: learning management system

RTS: runtime service

SPM: smallest permitted maximum

URI: Uniform Resource Identifier

URN: Uniform Resource Name

## 4. Conformance

Conformance to this Standard is discussed in 4.1 – 4.6.

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

### 4.1 Data instances

A conforming data instance shall be an instance of the data model as defined in 6.1.

### 4.2 Sending implementations

A conforming sending implementation shall send data instances that conform to this Standard.

### 4.3 Receiving implementations

A conforming receiving implementation shall accept data instances that conform to this Standard.

### 4.4 Repository implementations

A conforming repository implementation shall accept, store, and provide data that conforms to this Standard upon request.

### 4.5 Implementation-defined values

The processing and meanings of values that are not specified by this Standard (e.g., sentinel, missing, and empty values) are implementation-defined.

NOTE—For example, bindings, application profiles, or implementations may specify the processing or meanings of default values or sentinel values for specific data elements. An application profile might specify that in the absence of another value, the default value for `mode` is `normal`.

### 4.6 Smallest permitted maximum values

This Standard defines smallest permitted maximum (SPM) values for data elements with data types that include bag, array, set, and characterstring. For these data elements, a receiving im-

plementation or a repository implementation that conforms to this Standard shall accept and process at least that number of entries or characters specified by the SPM for the element and may accept and process a larger number.

NOTES:

1—The intent is for the SPM values to cover most cases.

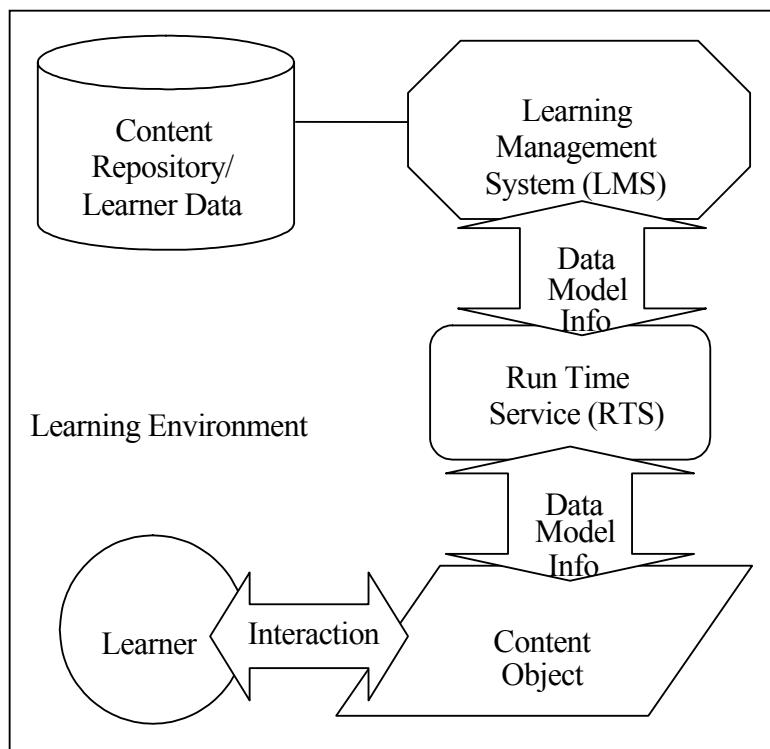
2—In this subclause, the meaning of "processing" is dependent upon the nature of the application.

3—This Standard defines no provisions for how and whether a sending system can determine whether a receiving system can process more than the SPM for a particular data element.

## 5. Conceptual model (informative)

*This Clause is informative.*

In one conceptual model for the use of this Standard, shown in Figure 1, the learner interacts with a content object in the learning environment. The content object may require information about the learner. It acquires this information through an RTS, which, in turn, gets the information from an LMS.



**Figure 1—Conceptual model diagram**

As the learner interacts with the content object, the content object may gather information on the learner's performance. This information is passed to the RTS, which passes it on to the LMS.

Other conceptual models exist that may use the data model. Although this conceptual model includes an RTS and an LMS, they are not required for the use of the data model. This conceptual model is designed to describe only one possible use of the data model.

## 6. Data model

This Clause defines a data model that a content object can use for obtaining information from an RTS to enable the content object to perform its expected functions and that an RTS can use for obtaining information from a content object to enable the RTS to manage the content object properly. The data model provides a description of the information that may pass to and from the content object.

This Standard does not specify how, when, or in which direction the information may flow. In addition, this Standard does not specify persistence of the data, how often it may be written or rewritten, or by whom it may be created or destroyed.

Unless noted otherwise, all components of "records" are optional in a data instance.

### NOTES:

1—The use of ISO/IEC 11404 notation in the synopses in 6.1 and 6.2 is for descriptive purposes only.<sup>1</sup> A complete implementation of the operations defined in ISO/IEC 11404 is not required for conformance.

2—The ISO/IEC 11404 notation describes the semantics of the language-independent data types across all bindings (e.g., implementation of a data type as itself, its subtypes, its subclasses, and its specializations). For example, an ISO/IEC 11404 "array" may be implemented as an SQL table (because SQL tables support indexing, a requirement for arrays); an ISO/IEC 11404 "state" may be implemented as a C programming language bit field; an ISO/IEC 11404 "characterstring" may be implemented in any encoding (e.g., ISO 646, ASCII, ISO 8859-1, UTF-8, UTF-16, UTF-32, etc.) that supports the repertoire specified in the parameter for the characterstring data type.

3—All examples in 6.1 and 6.2 are informative and do not endorse any particular binding.

4—The following language-independent data types are defined in ISO/IEC 11404: array, bag, characterstring, choice, real, record, set, state, time, and timeinterval.

5—The data element and data type labels used in the synopses in 6.1 and 6.2 are for reference only. Implementations are not required to use the exact same labels, as long as the data elements

---

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

and data types are semantically equivalent. It is recommended that implementations use spellings for labels similar to the spellings specified in this Standard.

6—This Standard does not define an extension mechanism for the data model. Implementers may create additional data models for content object communications. Such additional data models may be used to augment this data model to support different communities of practice.

## 6.1 Content object communication

### Synopsis

```

content_object_communication :
record
(
  comments_from_learner :
    array(0..249) of comment_type,
    // the SPM for the array is 250
  comments_from_lms :
    array(0..99) of comment_type,
    // the SPM for the array is 100
  completion_status :
    completion_status_type,
  completion_threshold :
    real(10,7) range(0..1),
  credit :
    state( credit, no_credit ),
  data_model_version :
    characterstring(iso-10646-1),
    // SPM: 250 characters
  entry :
    state( ab_initio, resume, _nil_ ),
  exit :
    state( timeout, suspend, logout, normal, _nil_ ),
  interactions :
    bag of interaction_type,
    // SPM: 250 interaction_type records in the bag
  launch_data :
    characterstring(iso-10646-1),
    // SPM: 4000 characters
  learner_id :
    long_identifier_type,
  learner_name :
    localized_string_type(250),
    // the parameter value is the SPM
  learner_preference_data :
    learner_preference_type,
  lesson_status :
    state( passed, completed, failed, incomplete, browsed,
    not_attempted ),

```

```

location :
    characterstring(iso-10646-1),
    // SPM: 1000 characters
max_time_allowed :
    timeinterval(second,10,2),
mode :
    state( browse, normal, review ),
objectives :
    set of objective_type,
    // SPM: 100 objective_type records in the bag
progress_measure :
    progress_measure_type,
raw_passing_score :
    real(10,7),
scaled_passing_score :
    real(10,7) range(-1..1),
score :
    score_type,
session_time :
    timeinterval(second,10,2),
success_status :
    success_status_type,
suspend_data :
    characterstring(iso-10646-1),
    // SPM: 4000 characters
time_limit_action :
    state( exit_message, continue_message, exit_no_message,
          continue_no_message ),
total_time :
    timeinterval(second,10,2),
),

```

## Description

The components of `content_object_communication` are defined in 6.1.1 – 6.1.27. Depending on the direction, destination, and purpose of the communication, an instance of `content_object_communication` shall include zero or more of the defined components.

## 6.1.1 Comments from learner

### Synopsis

```

comments_from_learner :
    array(0..249) of comment_type,
    // the SPM for the array is 250

```

### Description

The values of this data element are comments from the learner.

Subclause 6.2.1 defines `comment_type`.

## NOTES:

1—The values of this data element are intended to provide feedback about the content object or the learning experience with the content object from a specific learner. Using this data element for other purposes may adversely affect interoperability.

2—This Standard does not specify the mechanism for collecting comments.

## 6.1.2 Comments from LMS

### Synopsis

```
comments_from_lms :
    array(0..99) of comment_type,
    // the SPM for the array is 100
```

### Description

The values of this data element are comments and annotations intended to be made available to the learner.

Subclause 6.2.1 defines `comment_type`.

## NOTES:

1—The values of this data element are intended to provide information about the content object or the learning experience with the content object. Using this data element for other purposes may adversely affect interoperability.

2—This Standard does not specify the mechanism for collecting comments.

## 6.1.3 Completion status

### Synopsis

```
completion_status :
    completion_status_type,
```

### Description

The value of this data element indicates whether the learner has completed the content object.

Subclause 6.2.2 defines `completion_status_type`.

NOTE—This Standard does not specify how to determine `completion_status`. It may be reported by a content object, determined by an RTS by comparing a progress measure with a threshold, determined on the basis of objectives set by an outside agent (e.g., an instructor), or determined by some other means.

## 6.1.4 Completion threshold

### Synopsis

```
completion_threshold :
    real(10,7) range(0..1),
```

### Description

The value of this data element is a value against which the measure of the progress the learner has made toward completing the content object can be compared to determine whether the content object should be considered completed.

NOTE—The `completion_threshold` is designed to be used in conjunction with the `progress_measure` (see 6.1.19). For example, if the `completion_threshold` for a content object is 0.85 and a learner achieves a `progress_measure` of 0.90, a `completion_status` of `completed` (see 6.1.3) may be assigned to that content object for that learner. However, this Standard does not specify or require that an RTS, content object, or any other system component interpret or take action in response to a `completion_threshold`.

## 6.1.5 Credit

### Synopsis

```
credit :
    state( credit, no_credit ),
```

### Description

The value of this data element indicates whether the learner will be credited for performance in this content object. This data element shall have one of the following permissible values:

- `credit`: The learner is taking the content object for credit.
- `no_credit`: The learner is taking the content object for no credit.

NOTE—This Standard does not specify how to determine the value of `credit` or what it means to give credit for performance.

## 6.1.6 Data model version

### Synopsis

```
data_model_version :
    characterstring(iso-10646-1),
    // SPM: 250 characters
```

### Description

The value of this data element indicates the version of the data model defined in this Standard. The value shall consist of a period-delimited string containing major and minor release values

as whole numbers; for example, "1.0". Any characters appearing after the minor release value shall be separated from the minor release value by a period ("."). The syntax and semantics of any characters following the minor release value are not specified by this Standard.

For this edition of this Standard, the major version number shall be "1" and the minor version number shall be "0". Therefore, the first three characters of the value shall be "1.0".

An implementation may append additional characters to the value of this data element, in which case the first four characters shall be "1.0."

## 6.1.7 Entry

### Synopsis

```
entry :
    state( ab_initio, resume, _nil_ ),
```

### Description

The value of this data element is information that asserts whether the learner has previously accessed the content object. This data element shall have one of the following permissible values:

- `ab_initio`: Indicates that the learner has not accessed the content object during the current learner attempt.
- `resume`: Indicates that (1) the learner has previously accessed the content object during the current learner attempt, and (2) upon exiting, the `exit` data element had the value `suspend` (see 6.1.8).
- `_nil_`: Indicates all other conditions. There is no knowledge of previous access, or no specific entry condition is indicated.

NOTE—If the value for `entry` is `resume`, it indicates that either `location` or `suspend_data` may contain data stored in a previous learner session that is relevant to resuming the learner session (see 6.1.15 and 6.1.25, respectively).

## 6.1.8 Exit

### Synopsis

```
exit :
    state( timeout, suspend, logout, normal, _nil_ ),
```

### Description

The value of this data element indicates how or why the learner left the content object. This data element shall have one of the following permissible values:

- `timeout`: The content object terminated because the time limit specified by `max_time_allowed` had been exceeded (see 6.1.16).

- suspend: The learner exited the content object with the intent of returning to it.
- logout: The content object signaled a desire to terminate the entire learning activity of which the content object is a part.
- normal: The content object exited normally.
- \_nil\_: The exit conditions are undetermined.

## 6.1.9 Interactions

### Synopsis

```

interactions :
  bag of interaction_type,
  // SPM: 250 interaction records in the bag

type interaction_type =
  record
  (
    id :
      long_identifier_type,
    type :
      state( true_false, multiple_choice, fill_in,
             long_fill_in, likert, matching, performance,
             sequencing, numeric, other ),
    objectives_id :
      array(0..9) of long_identifier_type,
      // the SPM for the array is 10
    time_stamp :
      date_time_type,
    correct_responses :
      correct_responses_type,
    weighting :
      real(10,7),
    learner_response :
      learner_response_type,
    result :
      choice
      (
        state( result_state, numeric ),
      )
    of
      (
        result_state :
          state( correct, incorrect, unanticipated,
                neutral ),
        numeric :
          real(10,7),
      ),
    latency :
      timeinterval(second,10,2),
  )

```

```

        description :
            localized_string_type(250),
            // the parameter value is the SPM
    ),

```

### Description

The values of this data element define information pertaining to an interaction for the purpose of measurement or assessment. An instance of an `interaction_type` record shall include an interaction identifier (see 6.1.9.1). If an instance includes either `correct_responses` or `learner_response`, then the instance shall include `type` (see 6.1.9.2). All other components are optional.

The components of `interaction_type` are defined in 6.1.9.1 – 6.1.9.10.

NOTES:

1—Interactions are intended to be responses to individual questions or tasks that the content developer wants to record. This Standard does not specify how interaction data is to be recorded, used, or interpreted.

2—The interactions data model includes data elements that correspond to a limited set of interaction types, but it does not support logging of discrete learner events.

3—This Standard does not specify how interactions are presented or rendered.

4—This Standard does not specify how interactions are grouped in a question (i.e., one or multiple interactions per question).

5—The primary intent of interaction data is to communicate information about the status of an interaction object, such as a test item, a simulation, or another interactive feature of the content object. Interaction data also may be used to communicate interaction events as they occur, but in that case, only the data elements that carry information specific to the event should be communicated.

### 6.1.9.1 ID

#### Synopsis

```

    id :
        long_identifier_type,

```

#### Description

The value of this data element is a label for the interaction. This label shall be unique at least within the scope of the content object.

Subclause 6.2.6 defines `long_identifier_type`.

NOTE—This Standard does not specify how IDs are created, assigned, or resolved.

## 6.1.9.2 Type

### Synopsis

```

type :
state( true_false, multiple_choice, fill_in, long_fill_in,
       likert, matching, performance, sequencing, numeric,
       other ),

```

### Description

The value of this data element indicates which category of interaction is recorded in an instance of an interaction. It is also used to determine how the interaction response should be interpreted. This data element shall have one of the following permissible values. The content developer may create extended types using *other*.

- *true\_false*: The interaction has two possible responses. *Examples*: "True or False," "Yes or No," "Agree or Disagree."
- *multiple\_choice*: The interaction has a set of two or more possible responses. This interaction type can be used for interactions in which the learner selects just one choice and for interactions in which the learner can select more than one choice.
- *fill\_in*: The interaction requires the learner to supply a short response in the form of one or more strings of characters. *Note*: Typically, the correct response consists of part of a word, one word, or a few words.
- *long\_fill\_in*: The interaction requires the learner to supply a response in the form of a long string of characters. *Notes*: (1) Typically, the correct response is a sentence, paragraph, or short composition, but long composition forms are also possible. (2) Typically, the interaction is presented as an examination statement the learner must analyze and respond to by creating a written answer of a specified length, such as a short or long essay.
- *likert*: The interaction asks the learner to select from a discrete set of choices on a scale. *Note*: This Standard does not specify the number of choices, the scale, nor the meaning of the scale. *Example*: A typical response scale has five choices ranging from "strongly disagree" to "strongly agree."
- *matching*: The interaction consists of two sets of items. Members of the first set are related to zero or more members of the second set. Responding to the interaction requires the learner to indicate matches between members of the first set and members of the second set.
- *performance*: The interaction requires the learner to perform a task that requires multiple steps. *Example*: The task is a simulation for the changing of a sparkplug on an automobile engine involving six steps: (1) pull off the rubber boot from the plug, (2) unscrew the spark plug, (3) gap the replacement plug to a specific dimension, (4) screw in the replacement, (5) torque the plug using a torque wrench set to 12 foot-pounds, and (6) push the boot back on.
- *sequencing*: The interaction requires the learner to identify a logical order for the members of a list. *Example*: The learner may be asked to place a series

of events in chronological order or to rank a group of items by the order of their importance.

- `numeric`: The interaction requires a numeric response from the learner.
- `other`: Any other type of interaction not defined by this Standard. The semantics and structure of the `correct_responses` and `learner_response` data element values are not defined by this Standard when the interaction type is `other` (see 6.1.9.5 and 6.1.9.7, respectively). *Note*: When the interaction type is `other`, information identifying this extended type should be embedded in the `correct_responses` and `learner_response` data element values. For example, this may take the form of a prefix in the string used to communicate those values.

### 6.1.9.3 Objectives ID

#### Synopsis

```
objectives_id :
  array(0..9) of long_identifier_type,
  // the SPM for the array is 10
```

#### Description

The values of this data element are labels for objectives (see 6.1.18) associated with the interaction. The labels shall be unique at least within the scope of the content object.

Subclause 6.2.6 defines `long_identifier_type`.

NOTE—This Standard does not specify how objective IDs are created, assigned, or resolved.

### 6.1.9.4 Time stamp

#### Synopsis

```
time_stamp :
  date_time_type,
```

#### Description

The value of this data element is the point in time at which the interaction was first made available to the learner for learner interaction and response.

Subclause 6.2.3 defines `date_time_type`.

NOTES:

1—This Standard does not specify how the `time_stamp` value is obtained.

2—If several interactions are presented at the same time, they have the same `time_stamp` value. If an interaction was never available for response, such as an interaction that is not used in an adaptive test, no `time_stamp` value is available for that interaction.

3—If a `time_stamp` value is available for an interaction but no learner response data is available, it should be assumed that the interaction was made available to the learner but the learner did not respond.

### 6.1.9.5 Correct responses

#### Synopsis

```

correct_responses :
    correct_responses_type,

type correct_responses_type =
    choice
    (
        state( true_false, multiple_choice, fill_in, long_fill_in,
            likert, matching, performance, sequencing, numeric,
            other ),
    )
of
(
    true_false :
        state( true, false ),
    multiple_choice :
        set of set of short_identifier_type,
        // set of set SPM: 10 sets
        // set of short_identifier_type SPM: 36
        // short identifiers
    fill_in :
        bag of record
        // SPM: 5 records
        (
            case_matters :
                boolean,
            order_matters :
                boolean,
            match_text :
                array(0..9) of localized_string_type(250),
                // the SPM for the array is 10
                // the parameter value is the SPM for the
                // localized string
        ),
    long_fill_in :
        bag of record
        // SPM: 5 records
        (
            case_matters :

```



```

sequencing :
    bag of array(0..35) of short_identifier_type,
        // bag SPM: 5 arrays
        // the SPM for the array is 36
numeric :
    record
    (
        min :
            real(10,7),
        max :
            real(10,7),
    ),
other :
    characterstring(iso-1046-1),
        // SPM: 4000 characters
),

```

## Description

The values of this data element indicate the correct response(s) to the interaction. This data element shall have one of ten possible variants that shall match the conditions described below. The content developer may create extended types using `other`.

Several response types support more than one correct response. For these types, a list (bag) of correct response(s) is provided. A correct response may require multiple inputs. For these responses, a collection of input(s) is provided.

- `true_false`: A state that contains the values `true` and `false`. The state `true` means true or an equivalence of true in a particular context (e.g., agree, yes, richtig). The state `false` means false or an equivalence false in a particular context (e.g., disagree, no, falsch).
- `multiple_choice`: A set that contains one or more sets of short identifiers. Any of the sets of short identifiers satisfies the requirement for a correct response. Multiple sets may be defined if more than one correct response exists. A set of short identifiers may contain zero or more short identifiers, all of which are required for a correct response. Each of the short identifiers represents an expected choice. If a set of short identifiers is empty, it represents that the correct response is no choice. *Examples*: 1) A single choice may be allowed: "alpha." 2) Multiple sets of choices may be allowed: "alpha," "bravo," "charlie," and "alpha," "bravo," "delta."
- `fill_in`: A bag of records. The bag contains one or more records, any of which satisfies the requirement for a correct response. Each record consists of an array of localized strings and two flags. The localized strings represent a correct response. The `case_matters` flag indicates whether the case of the string is used to evaluate the correctness of the response. If the value of the flag is true, the case of the learner response shall match the correct response. If the value of the flag is false, the case of the learner response is not used in evaluating the response. If `case_matters` is not specified, it is assumed to be false. The `order_matters` flag indicates whether the order of the inputs

for a correct response matters. If the value of the flag is true, then order matters, and the order of the learner's responses should be used to evaluate correctness of the response. If the value of the flag is false, then order does not matter, and the order of the learner's responses should not be used to evaluate correctness of the response. If `order_matters` is not specified, it is assumed to be true.

- `long_fill_in`: A bag of records. The bag contains one or more records, any of which satisfies the requirement for a correct response. Each record consists of a localized string and a flag. The localized string represents a correct response. The flag indicates whether the case of the string is used to evaluate the correctness of the learner response. If the value of the flag is true, the case of the learner response shall match the correct response. If the value of the flag is false, the case of the learner response is not used in evaluating the response. If `case_matters` is not specified, it is assumed to be false. *Note*: Although a correct response for `long_fill_in` can be specified, the evaluation of a `long_fill_in` response typically involves an interpretative process that is outside of the scope of this Standard.
- `likert`: A short identifier that matches a choice on a scale. *Note*: Although a correct response for `likert` can be specified, `likert` interactions typically do not include correct responses.
- `matching`: A bag of bags of records. The single outer bag contains one or more inner bags. Each inner bag contains one or more records. If more than one inner bag exists, any of the inner bags satisfies the requirement for a correct response. If more than one record is contained by an inner bag, all the records are required for the correct response specified by that inner bag. Each of the records is a pair of short identifiers representing an expected matching input. Each of the correct response pairs consists of a source and a target. Each source and each target shall be represented by a short identifier. The scope for the short identifiers used for sources and targets shall be the interaction. The same short identifier may appear in more than one source-target pair.
- `performance`: A bag of records. The bag contains one or more records, any of which satisfies the requirement for a correct response. Each record consists of a flag and an array. The array represents a set of correct responses. The `order_matters` flag indicates whether the order of the inputs matters for a correct response. If the value of the flag is true, the order of the learner's responses should be used to evaluate correctness of the response. If the value of the flag is false, the order of the learner's responses should not be used to evaluate correctness of the response. If `order_matters` is not specified, it is assumed to be true. Each correct response consists of a name and either a single literal value or a numeric range. If the correct response is expressed as a literal value, this Standard does not specify how to use the value to evaluate the corresponding response. If the correct response is expressed as a numeric range, the learner's response should be within the specified range to be judged correct.
- `sequencing`: A bag of arrays of short identifiers. The bag contains one or more arrays, any of which satisfies the requirement for a correct response.

Each array represents a sequence of zero or more short identifiers for a correct response. Each short identifier identifies one element available to be sequenced when the interaction is presented to the learner. Each array shall contain a different sequence of short identifiers. Different arrays may contain different short identifiers.

- `numeric`: Two real numbers. The numbers may be used to express an inclusive range for the correct response. If a `min` value is specified with no `max` value, the upper limit of the range is unbounded. If a `max` value is specified with no `min` value, the lower limit of the range is unbounded. If both the `min` and `max` values are unspecified, both the upper and lower limits of the range are unbounded. If the `min` and `max` values are equal, the range is a single value. *Note*: This Standard does not specify the number of significant digits that should be considered in evaluating results against the specified range.
- `other`: A string defined by the specific "other" interaction type (see 6.1.9.2). The content of this string is not defined by this Standard.

Subclauses 6.2.5 and 6.2.9 define `localized_string_type` and `short_identifier_type`, respectively.

NOTE—The `correct_responses` data element is a structured mechanism for identifying the correct learner response or responses relating to each of the types of interactions described in 6.1.9.2. The determination of correctness is an implementation-defined feature (see 6.1.9.8).

### 6.1.9.6 Weighting

#### Synopsis

```
weighting :
    real(10,7),
```

#### Description

The value of this data element is a weight given to the interaction that may be used by the content object to compute a value for a score.

NOTE—Interaction weights typically are used to explain the effect of an interaction on the value of the `score` data element for an objective or for the content object (see 6.1.18.2 and 6.1.22, respectively), but are not intended to be used by systems other than the content object to compute a score.

### 6.1.9.7 Learner response

#### Synopsis

```
learner_response :
    learner_response_type,
type learner_response_type =
    choice
```

```

(
  state( true_false, multiple_choice, fill_in, long_fill_in,
        likert, matching, performance, sequencing, numeric,
        other ),
)
of
(
  true_false :
    state( true, false ),
  multiple_choice :
    set of short_identifier_type,
    // SPM: 36 short identifiers
  fill_in :
    array(0..9) of localized_string_type(250),
    // the SPM for the array is 10
    // the parameter value is the SPM for the localized
    // string
  long_fill_in :
    localized_string_type(4000),
    // the parameter value is the SPM
  likert :
    short_identifier_type,
  matching :
    bag of record
    // SPM: 36 records
    (
      source :
        short_identifier_type,
      target :
        short_identifier_type,
    ),
  performance :
    array(0..249) of record
    // the SPM for the array is 250
    (
      step_name :
        short_identifier_type,
      step_answer :
        choice
        (
          state( literal, numeric ),
        )
    of
    (
      literal :
        characterstring(iso-10646-1),
        // SPM: 250 characters
      numeric :
        real(10,7),
    ),
  ),
),

```

```

sequencing :
    array(0..35) of short_identifier_type,
        // the SPM for the array is 36
numeric :
    real(10,7),
other :
    characterstring(iso-1046-1),
        // SPM: 4000 characters
),

```

## Description

The values of this data element consist of data generated when a learner responds to an interaction. This data element shall have one of the ten possible variants that shall match the conditions described below. The content developer may create extended types using "other".

- `true_false`: A state that contains the values `true` and `false`. The state `true` means true or an equivalence of true in a particular context (e.g., agree, yes, richtig). The state `false` means false or an equivalence of false in a particular context (e.g., disagree, no, falsch).
- `multiple_choice`: A set of short identifiers. The values of the identifiers in the set represent the choices made by the learner. The set may contain zero or more short identifiers. *Examples*: If a single choice was allowed the set would contain a single identifier, e.g., "alpha." If a combination of choices was allowed, the set would contain multiple identifiers, e.g., "alpha," "bravo," "delta," the order of which is insignificant.
- `fill_in`: An array of localized strings.
- `long_fill_in`: A localized string.
- `likert`: A short identifier. The value of the identifier represents the choice made by the learner.
- `matching`: A bag that contains zero or more records. Each record contains a source and a target that are represented by short identifiers. Each record represents a match made by the learner.
- `performance`: An array of responses in the order in which they were provided by the learner in response to the interaction. Each response consists of a step name (a short identifier) and either a single literal value (a character string) or a number. The step names and types of the responses shall match those provided in the `correct_responses` for the interaction (see 6.1.9.5), but the responses in the `learner_response` may be in a different order. Because a learner may perform the same step more than once, the step names of the responses may not be unique (i.e., a step name may appear more than once with the same value or with a different value). *Example*: If the performance involves setting several valves to specific positions, the learner may adjust the position of the same valve more than once in the course of the performance. The name-value pairs for the response might be "valve 1:open, valve 2:closed, valve 1:closed." *Notes*: (1) The SPM for `performance` for `learner_response` is twice the size of the SPM for `performance` for `correct_responses` (see 6.1.9.5) to allow the recording of extra steps, as

in the example above. (2) The syntax of the name-value pairs is not specified by this Standard.

- sequencing: An array of zero or more short identifiers. The sequence determined by the learner is represented by the order of the elements in the array. Each short identifier identifies one element that was available to be sequenced.
- numeric: A real number.
- other: A string defined by the specific "other" interaction type (see 6.1.9.2). The content of this string is not defined by this Standard.

Subclauses 6.2.5 and 6.2.9 define `localized_string_type` and `short_identifier_type`, respectively.

#### NOTES:

1—The `learner_response` data element is a structured mechanism for identifying the exact learner response relating to each of the types of interactions described in 6.1.9.2. The determination of correctness is an implementation-defined feature of the content object.

2—The type of the interaction, as defined in 6.1.9.2, has to be known in order to select the appropriate variant.

### 6.1.9.8 Result

#### Synopsis

```

result :
  choice
  (
    state( result_state, numeric ),
  )
of
  (
    result_state :
      state( correct, incorrect, unanticipated, neutral ),
    numeric :
      real(10,7),
  ),

```

#### Description

The value of this data element is a judgment of the correctness of the learner response. This data element shall have one of the following permissible values:

- `correct`: The learner response was correct.
- `incorrect`: The learner response was incorrect.
- `unanticipated`: The learner response was not expected.
- `neutral`: The learner response was neither correct nor incorrect.
- `numeric`: A real number.

## NOTES:

1—This Standard does not specify where or how the value of `result` is determined.

2—The numeric value `real(10,7)` is included to provide the capability of reporting a numeric estimate of the correctness of the learner response. This Standard does not specify how correctness is represented in the numeric value.

### 6.1.9.9 Latency

#### Synopsis

```
latency :
    timeinterval(second,10,2),
```

#### Description

The value of this data element is the time elapsed between the time the interaction was made available to the learner for response and the time of the first response.

A string binding conforming to ISO 8601:2000 may be used to communicate time interval values (see Annex C).

NOTE—The `latency` information is not available for an interaction if the learner did not respond. The `latency` is, in effect, the time difference between the `time_stamp` (see 6.1.9.4) of the interaction and the time of the first response. If several interactions have the same `time_stamp` because they became available for response at the same time, the `latency` recorded for each interaction can be used to determine the order in which the learner responded to these interactions.

### 6.1.9.10 Description

#### Synopsis

```
description :
    localized_string_type(250),
    // the parameter value is the SPM
```

#### Description

The value of this data element is a brief informative description of the interaction.

Subclause 6.2.5 defines `localized_string_type`.

## 6.1.10 Launch data

### Synopsis

```
launch_data :  
    characterstring(iso-10646-1),  
    // SPM: 4000 characters
```

### Description

The value of this data element provides data specific to a content object that the content object can use for initialization. The value of this data element is not specified.

NOTE—The allowable values for this data element are defined by the implementer of the content object. Typically, the documentation for the content object would specify what data can or has to be provided.

## 6.1.11 Learner ID

### Synopsis

```
learner_id :  
    long_identifier_type,
```

### Description

The value of this data element identifies the learner on behalf of whom this content object instance was launched. The label shall be unique at least within the scope of the content object.

Subclause 6.2.6 defines `long_identifier_type`.

NOTE—This Standard does not specify how learner IDs are created, assigned, or resolved.

## 6.1.12 Learner name

### Synopsis

```
learner_name :  
    localized_string_type(250),  
    // the parameter value is the SPM
```

### Description

The value of this data element is the name of the learner.

Subclause 6.2.5 defines `localized_string_type`.

NOTE—This Standard does not specify how learner names are created, assigned, or resolved.

## 6.1.13 Learner preference data

### Synopsis

```

learner_preference_data :
  learner_preference_type,
type learner_preference_type =
  record
  (
    audio_level :
      real(10,7) range(0..*),
    language :
      language_type,
    delivery_speed :
      real(10,7) range(0..*),
    audio_captioning :
      state( off, no_change, on ),
  ),

```

### Description

The values of this data element specify learner preferences associated with the learner's use of the content object.

The components of `learner_preference_data` are defined in 6.1.13.1 – 6.1.13.4.

NOTE—This Standard does not specify whether the content object, the RTS, or both have the ability to set or interpret learner preferences.

### 6.1.13.1 Audio level

#### Synopsis

```

audio_level :
  real(10,7) range(0..*),

```

#### Description

The value of this data element is a multiplier value that specifies an intended change in perceived audio level relative to an implementation-specific reference level with the value 1 meaning "no change." For example, the value 0 specifies infinite attenuation, the value 0.5 specifies an attenuation of 10 decibels, and the value 2 specifies an amplification of 10 decibels.

NOTE—The multiplier value is not intended to be applied to the effect of previous changes communicated through this data element, but rather to the same implementation-specific reference level.

### 6.1.13.2 Language

#### Synopsis

```
language :
    language_type ,
```

#### Description

The value of this data element is the learner's preferred language for a content object with multilingual capability.

Subclause 6.2.4 defines `language_type`.

### 6.1.13.3 Delivery speed

#### Synopsis

```
delivery_speed :
    real(10,7) range(0..*),
```

#### Description

The value of this data element is a multiplier that specifies the learner's preferred relative speed of content delivery expressed as a change in speed relative to an implementation-specific reference speed. For example, the value 2 is twice as fast as the reference speed and the value 0.5 is one half the reference speed.

NOTES:

1—A value of 0 indicates that delivery is stopped.

2—The multiplier value is not intended to be applied to the effect of previous changes communicated through this data element, but rather to the same implementation-specific reference speed.

### 6.1.13.4 Audio captioning

#### Synopsis

```
audio_captioning :
    state( off, no_change, on ),
```

#### Description

The value of this data element specifies whether captioning text corresponding to audio is displayed. This data element shall have one of the following permissible values:

- `off`: Captioning is off, and text corresponding to audio is not displayed.
- `no_change`: The current captioning setting.
- `on`: Captioning is on, and text corresponding to audio is displayed.

## 6.1.14 Lesson status

### Synopsis

```
lesson_status :
    state( passed, failed, completed, incomplete, browsed,
          not_attempted ),
```

### Description

This data element is included for backward compatibility with legacy implementations. The data elements `completion_status` and `success_status` should be used (see 6.1.3 and 6.1.24, respectively).

The value of this data element indicates whether the learner has attempted, completed, passed, failed, or browsed the associated content object. This data element shall have one of the following permissible values:

- `passed`: The learner has satisfied the requirements to pass the content object.
- `failed`: The learner has not satisfied the requirements to pass the content object.
- `completed`: The learner has satisfied the requirements to complete the content object.
- `incomplete`: The learner has not satisfied the requirements to complete the content object.
- `browsed`: The learner has accessed the content object with a mode of browse or elected to browse while in the content object after a normal launch.
- `not_attempted`: The learner has not accessed the content object, or the learner previously has accessed the content object but has experienced so little of it that it is considered to be not attempted.

### NOTES:

1—This Standard does not specify how to determine `lesson_status`. It may be reported by a content object, determined by an RTS by comparing scores to mastery scores, determined on the basis of objectives set by an outside agent (e.g., an instructor), or by some other means.

2—The `completion_status` and `success_status` data elements should be used because the `lesson_status` data element may exist only in legacy implementations.

## 6.1.15 Location

### Synopsis

```
location :
    characterstring(iso-10646-1),
    // SPM: 1000 characters
```

**Description**

The value of this data element is a location in the content object. The value and its meaning are defined by the content object and are not specified by this Standard. This Standard does not specify how an implementation shall represent that there is no value for `location`.

## NOTES:

1—Depending on the implementation, the absence of a value for `location` could be represented as an empty string, a null element, or the absence of the data element.

2—If a content object communicates a `location` on exit, this data element provides support for a mechanism that lets the learner return to the content object at the same place he or she left it. This data element can identify the learner's exit point with a value that is meaningful to the content object only, and that location information can be used by the content object as an entry point the next time the learner enters the content object. This data element also can be used by the content object to communicate its location to the RTS on an ongoing basis. *Example:* An RTS may be able to use this information to create bookmarks or to synchronize reference materials or annotations with the `location` reported by the content.

**6.1.16 Max time allowed****Synopsis**

```
max_time_allowed :
    timeinterval(second,10,2),
```

**Description**

The value of this data element is the amount of accumulated time the learner is allowed to use a content object in the learner attempt. (See 6.1.26 for the content object's expected response to exceeding the limit.)

A string binding conforming to ISO 8601:2000 may be used to communicate time interval values (see Annex C).

NOTE—The learner attempt begins with the beginning of the first learner session and continues until the activity terminates.

**6.1.17 Mode****Synopsis**

```
mode :
    state( browse, normal, review ),
```

## Description

The value of this data element identifies one of three possible modes in which a content object may be presented to a learner. This data element shall have one of the following permissible values:

- *browse*: The content object is presented without the intent of recording any information about the current learner session.
- *normal*: The content object is presented with the intent of recording information about the current learner session.
- *review*: The content object has previously recorded information about the learner attempt and is presented without the intent of updating this information with data from the current learner session. *Note*: The learner attempt begins with the beginning of the first learner session and continues until the activity terminates.

## 6.1.18 Objectives

### Synopsis

```

objectives :
  set of objective_type,
  // SPM: 100 objective_type records in the bag

type objective_type =
  record
  (
    id :
      long_identifier_type,
    score :
      score_type,
    status :
      state( passed, completed, failed, incomplete, browsed,
             not_attempted ),
    progress_measure :
      progress_measure_type,
    completion_status :
      completion_status_type,
    success_status :
      success_status_type,
    description :
      localized_string_type(250),
      // the parameter value is the SPM
  ),

```

### Description

The values of this data element specify learning or performance objectives associated with a content object. An instance of an *objective\_type* record shall include an objective identifier (see 6.1.18.1); all other components are optional.

The components of `objective_type` are defined in 6.1.18.1 – 6.1.18.7.

NOTES:

1—Information about `objectives` may come from a content object, from an RTS, or from some other source.

2—This Standard does not define any relationship between `objectives` and the content object's `completion_status`, `lesson_status`, `score`, or `success_status` (see 6.1.3, 6.1.14, 6.1.22, and 6.1.24, respectively).

### 6.1.18.1 ID

#### Synopsis

```
id :
    long_identifier_type,
```

#### Description

The value of this data element is a label for the objective. This label shall be unique at least within the scope of the content object.

Subclause 6.2.6 defines `long_identifier_type`.

NOTE—This Standard does not specify how IDs are created, assigned, or resolved.

### 6.1.18.2 Score

#### Synopsis

```
score :
    score_type,
```

#### Description

The value of this data element is the score achieved by the learner for the objective.

Subclause 6.2.8 defines `score_type`.

NOTE—This Standard does not specify how the value of `score` is created or assigned.

### 6.1.18.3 Status

#### Synopsis

```
status :
    state( passed, completed, failed, incomplete, browsed,
           not_attempted ),
```

## Description

This data element is included for backward compatibility with legacy implementations. The data elements `completion_status` and `success_status` should be used (see 6.1.18.5 and 6.1.18.6, respectively).

The value of this data element indicates whether the learner has engaged with that portion of the content object related to the objective and, if so, whether the learner has demonstrated mastery of the objective. This data element shall have one of the following permissible values:

- `passed`: The objective was passed.
- `completed`: All parts of the content object related to the objective were accessed. The objective may or may not have been passed.
- `failed`: The objective was failed.
- `incomplete`: Not all parts of the content object related to the objective were accessed.
- `not_attempted`: No part of the content object related to the objective was accessed.
- `browsed`: No specific status information for the objective is available because the content object related to the objective was launched with a mode of `browse` (see 6.1.17).

### NOTES:

1—This Standard does not specify how to determine `status`. Status may be provided by the content object, by an RTS, or by some other means.

2—The `completion_status` and `success_status` data elements should be used because the `status` data element may exist only in legacy implementations.

## 6.1.18.4 Progress measure

### Synopsis

```
progress_measure :
    progress_measure_type ,
```

### Description

The value of this data element is a measure of the progress the learner has made toward completing the objective.

Subclause 6.2.7 defines `progress_measure_type`.

NOTE—This Standard does not specify how to determine the value of `progress_measure`.

### 6.1.18.5 Completion status

#### Synopsis

```
completion_status :  
    completion_status_type,
```

#### Description

The value of this data element indicates whether the learner has completed the objective.

Subclause 6.2.2 defines `completion_status_type`.

NOTE—This Standard does not specify how to determine `completion_status`. It may be reported by a content object, determined by an RTS, determined on the basis of objectives set by an outside agent (e.g., an instructor), or by some other means.

### 6.1.18.6 Success status

#### Synopsis

```
success_status :  
    success_status_type,
```

#### Description

The value of this data element indicates whether the learner has mastered the objective.

Subclause 6.2.10 defines `success_status_type`.

NOTE—This Standard does not specify how to determine `success_status`. It may be reported by a content object, determined by an RTS, determined on the basis of objectives set by an outside agent (e.g., an instructor), or by some other means.

### 6.1.18.7 Description

#### Synopsis

```
description :  
    localized_string_type(250),  
    // the parameter value is the SPM
```

#### Description

The value of this data element is a brief informative description of the objective.

Subclause 6.2.5 defines `localized_string_type`.

## 6.1.19 Progress measure

### Synopsis

```
progress_measure :
    progress_measure_type,
```

### Description

The value of this data element is a measure of the progress the learner has made toward completing the content object.

Subclause 6.2.7 defines `progress_measure_type`.

NOTES:

1—This Standard does not specify an exact relationship between `completion_status` (see 6.1.3) and values for `progress_measure` other than 0 or 1. Any value between 0 and 1 typically corresponds to a `completion_status` value of `incomplete`, unless the value is equal to or above a defined `completion_threshold` (see 6.1.4), in which case the value typically corresponds to a `completion_status` value of `completed`.

2—This Standard does not specify how to determine the value of `progress_measure`.

## 6.1.20 Raw passing score

### Synopsis

```
raw_passing_score :
    real(10,7),
```

### Description

The value of this data element is the raw passing score for a content object. The scale is not defined. This data element is included for backward compatibility with legacy implementations. The data element `scaled_passing_score` should be used (see 6.1.21).

NOTE—The `scaled_passing_score` data element should be used because the `raw_passing_score` data element may exist only in legacy implementations.

## 6.1.21 Scaled passing score

### Synopsis

```
scaled_passing_score :
    real(10,7) range(-1..1),
```

**Description**

The value of this data element is the scaled passing score for a content object. The value of this data element is scaled to fit the range -1 to 1 inclusive.

NOTES:

1—If a `scaled_passing_score` is defined for the use of a content object, this is a statement that the requirements associated with the use of that content object are achieved by obtaining a score (see 6.1.22) greater than or equal to the `scaled_passing_score`. For example, if the `scaled_passing_score` for a content object is 0.85 and a learner achieves a scaled score of 0.90, a `success_status` of `passed` may be assigned to that content object for that learner (see 6.1.24). However, this Standard does not specify or require that an RTS, content object, or any other system component interpret or take action in response to a `scaled_passing_score`.

2—A scaled score range of -1 to +1 is used to allow a content developer to more easily assign a penalty for an incorrect choice.

**6.1.22 Score****Synopsis**

```
score :
    score_type,
```

**Description**

The value of this data element is the learner's score for the content object.

Subclause 6.2.8 defines `score_type`.

**6.1.23 Session time****Synopsis**

```
session_time :
    timeinterval(second,10,2),
```

**Description**

The value of this data element is the amount of time that the learner has spent in the current learner session for this content object. If no learner session is in progress, the session time is the time the learner spent in the last learner session for this content object.

A string binding conforming to ISO 8601:2000 may be used to communicate time interval values (see Annex C).

## NOTES:

- 1—This Standard does not specify how to determine the value of `session_time` or its accuracy.
- 2—The value for `session_time` may be evaluated one or more times during a learner session. The value of `total_time` (see 6.1.27) is not updated until after the learner session has ended.
- 3—If a learner session is in progress, the actual duration of the learner attempt is the `total_time` (see 6.1.27) plus the current `session_time`.

## 6.1.24 Success status

### Synopsis

```
success_status :
    success_status_type,
```

### Description

The value of this data element indicates whether the learner has mastered the content object.

Subclause 6.2.10 defines `success_status_type`.

NOTE—This Standard does not specify how to determine `success_status`. It may be reported by a content object, determined by an RTS by comparing scores to mastery scores, determined on the basis of objectives set by an outside agent (e.g., an instructor), or by some other means.

## 6.1.25 Suspend data

### Synopsis

```
suspend_data :
    characterstring(iso-10646-1),
    // SPM: 4000 characters
```

### Description

The value of this data element provides information that may be created by a content object as a result of a learner accessing or interacting with that content object. The format of the content of this data element is unspecified.

NOTE—The intent is for the content object to store data for later use in the current learner session or a subsequent learner session between the content object and the same learner.

## 6.1.26 Time limit action

### Synopsis

```
time_limit_action :
    state( exit_message, continue_message, exit_no_message,
           continue_no_message ),
```

### Description

The value of this data element indicates what the content object should do when `max_time_allowed` is exceeded (see 6.1.16). This data element shall have one of the following permissible values:

- `exit_message`: The learner should be forced to exit the content object. The content object should provide a message to the learner indicating that the maximum time allowed for the learner attempt was exceeded.
- `continue_message`: The learner should be allowed to continue in the content object. The content object should provide a message to the learner indicating that the maximum time allowed for the learner attempt was exceeded.
- `exit_no_message`: The learner should be forced to exit the content object with no message.
- `continue_no_message`: Although the learner has exceeded the maximum time allowed for the learner attempt, the learner should be given no message and should not be forced to exit the content object.

NOTES:

1—When a message is presented to the learner, the content object defines the content and form of the message.

2—This Standard does not specify how the content object forces the learner to exit the content object.

## 6.1.27 Total time

### Synopsis

```
total_time :
    timeinterval(second,10,2),
```

### Description

The value of this data element is the sum of all of the learner's learner session times (see 6.1.23) accumulated in the current learner attempt prior to the current learner session. The value of `total_time` shall not be updated while a learner session is in progress.

A string binding conforming to ISO 8601:2000 may be used to communicate time interval values (see Annex C).

NOTE—The learner attempt begins with the beginning of the first learner session and continues until the activity terminates.

## 6.2 Auxiliary data types

The following data types are used in conjunction with the data elements described in 6.1.

### 6.2.1 Comment type

#### Synopsis

```

type comment_type =
  record
  (
    comment :
      localized_string_type(4000),
      // the parameter value is the SPM
    location :
      characterstring(iso-10646-1),
      // SPM: 1000 characters
    time_stamp :
      date_time_type,
  ),

```

#### Description

This data type describes textual input. Instances of this data type shall include a comment (see 6.2.1.1).

The components of the `comment_type` are defined in 6.2.1.1 – 6.2.1.3.

#### 6.2.1.1 Comment

##### Synopsis

```

comment :
  localized_string_type(4000),
  // the parameter value is the SPM

```

##### Description

This data element shall describe comments or annotations associated with a content object.

Subclause 6.2.5 defines `localized_string_type`.

NOTE—This Standard does not define a structure or format for the content of the localized string.

### 6.2.1.2 Location

#### Synopsis

```
location :
    characterstring(iso-10646-1),
    // SPM: 1000 characters
```

#### Description

This data element is the point in the content object at which the comment applies. If no value is specified for `location`, the comment is applicable to the entire content object. This Standard does not specify how an implementation shall represent that there is no value for `location`.

NOTES:

1—Depending on the implementation, the absence of a value for `location` could be represented as an empty string, a null element, or the absence of the data element.

2—This Standard does not specify how an implementation defines a location in a content object.

### 6.2.1.3 Time stamp

#### Synopsis

```
time_stamp :
    date_time_type,
```

#### Description

This data element is the point in time at which the comment was created or most recently changed.

Subclause 6.2.3 defines `date_time_type`.

### 6.2.2 Completion status type

#### Synopsis

```
type completion_status_type =
    state( completed, incomplete, not_attempted, unknown ),
```

#### Description

This data type indicates whether the learner has completed a content object or an objective. This data type shall have one of the following permissible values:

- `completed`: The learner has experienced enough of the content object or objective to consider it completed.

- `incomplete`: The learner has not experienced enough of the content object or objective to consider it completed.
- `not_attempted`: The learner is considered not to have used the content object or objective in any significant way. *Note*: The learner has not accessed the content object or objective, or the learner previously has accessed it but has experienced so little of it that it is considered to be not attempted.
- `unknown`: No assertion is made.

### 6.2.3 Date time type

#### Synopsis

```
type date_time_type =
    time(second,10,0),
```

#### Description

This data type represents a point in time. This data type shall have a required precision of 1 second and an optional precision of 0.01 seconds.

Implementations of this data type shall include distinct representations for points in time in the range 1970-01-01 00:00:00 through 2037-12-31 23:59:59, not including leap seconds, with a required precision of one second and an optional precision of 0.01 seconds. Implementations may include distinct representations for values beyond the required date and time range.

A string binding conforming to ISO 8601:2000 may be used to communicate date and time values (see Annex C).

NOTES:

1—This Standard does not specify how to translate times expressed with precisions of hundredths of a second to times expressed with precisions of seconds. This may be done by rounding, truncation, or another method.

2—Conforming implementations are permitted, but not required, to support the representation of leap seconds.

### 6.2.4 Language type

#### Synopsis

```
type language_type =
    characterstring(iso-646),
    // SPM: 250 characters
```

#### Description

The format of this data type is a character string consisting of a required language code followed by multiple, optional, hyphen-prefixed subcodes (see examples below).

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

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

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

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

Rules for additional subcodes are unspecified.

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

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

### Examples

```
"en-GB"
"de"
"fr-CA"
"it"
"grc" (Ancient Greek, until 1453)
"en-US-philadelphia"
"eng-GB-cockney"
"map-PG-buin" (Austronesian - Papua New Guinea Buin)
"gem-US-pennsylvania"
"i-bnn" (IANA Bunun)
```

## 6.2.5 Localized string type

### Synopsis

```
type localized_string_type(length) =
  record
  (
    language :
      language_type,
    string :
      characterstring(iso-10646-1),
      // SPM: the length parameter
  ),
```

### Description

This data type consists of a language specification for a string and the string itself.

The components of the `localized_string_type` are defined in 6.2.5.1 and 6.2.5.2.

### Examples

The following are three examples of localized strings: "Information Technology" in French, "localization" in British English, and "xxx" in Japanese hiragana.

```
( "fr", "Technologies de l'information" )
( "en-GB", "localisation" )
( "jp-JP-jisx208", "xxx" )
```

#### 6.2.5.1 Language

##### Synopsis

```
language :
    language_type,
```

##### Description

This data element specifies the language of the localized string.

Subclause 6.2.4 defines `language_type`.

#### 6.2.5.2 String

##### Synopsis

```
string :
    characterstring(iso-10646-1),
    // SPM: the length parameter
```

##### Description

This data element contains the text of the localized string.

#### 6.2.6 Long identifier type

##### Synopsis

```
type long_identifier_type =
    characterstring(iso-10646-1),
    // SPM: 4000 characters
```

##### Description

This data type is an identifier (a label) associated with an object that is intended to be unique within the context of usage of the object. The character string shall conform to the syntax for Uniform Resource Identifiers (URIs) as defined by RFC 2396.

NOTE—This Standard recommends that the URI be a globally unique identifier in the form of a Uniform Resource Name (URN) (see RFC 2141 [B4]).

## 6.2.7 Progress measure type

### Synopsis

```
type progress_measure_type :
    real(10,7) range(0..1),
```

### Description

This data type is a measure of the progress the learner has made toward completing a content object or an objective. A value of 0 corresponds to a `completion_status_type` value of `not_attempted` (see 6.2.2). A value of 1 corresponds to a `completion_status_type` value of `completed`.

## 6.2.8 Score type

### Synopsis

```
type score_type =
    record
    (
        raw :
            real(10,7),
        min :
            real(10,7),
        max :
            real(10,7),
        scaled :
            real(10,7) range(-1..1),
    ),
```

### Description

This data type describes scoring information.

The components of the `score_type` are defined in 6.2.8.1 – 6.2.8.4.

### 6.2.8.1 Raw

#### Synopsis

```
raw :
    real(10,7),
```

**Description**

This data element is a number that reflects the performance of the learner relative to the range bounded by the values of `min` and `max`.

NOTE—A raw score is not necessarily an unprocessed score. *Example:* An unprocessed score might be converted to a percentage score as a decimal value. That is, the learner achieved a score of 3 out of 4 possible, which is converted to a raw value of 0.75 with `min` (see 6.2.8.2) equal to 0 and `max` (see 6.2.8.3) equal to 1.

**6.2.8.2 Min****Synopsis**

```
min :
    real(10,7),
```

**Description**

This data element is the minimum value in the range for the raw score (see 6.2.8.1).

**6.2.8.3 Max****Synopsis**

```
max :
    real(10,7),
```

**Description**

This data element is the maximum value in the range for the raw score (see 6.2.8.1).

**6.2.8.4 Scaled****Synopsis**

```
scaled :
    real(10,7) range(-1..1),
```

**Description**

This data element is a number that reflects the performance of the learner. The value of the data element is scaled to fit the range -1 to 1, inclusive.

NOTE—A scaled score range of -1 to +1 is used to allow a content developer to more easily assign a penalty for an incorrect choice, such as in a flight simulation system where the learner's choice would have resulted in the loss of the aircraft and all aboard.

## 6.2.9 Short identifier type

### Synopsis

```
type short_identifier_type =  
    characterstring(iso-10646-1),  
    // SPM: 250 characters
```

### Description

This data type is an identifier (a label). The character string shall conform to the syntax for URIs as defined by RFC 2396.

## 6.2.10 Success status type

### Synopsis

```
type success_status_type =  
    state( passed, failed, unknown ),
```

### Description

This data type indicates whether the learner has mastered a content object or an objective. This data type shall have one of the following permissible values:

- `passed`: The learner has passed the content object or objective.
- `failed`: The learner has failed the content object or objective.
- `unknown`: No assertion is made.

## **Annex A**

(informative)

### **Bibliography**

[B1] AICC CMI001, CMI Guidelines for Interoperability, Version 3.5, April 2001.

[B2] IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition.

[A3] IEEE 1484.11.2–2003, Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication.

[B4] IETF RFC 2141, URN Syntax.

## Annex B

(informative)

### Understanding the ISO/IEC 11404:1996 real and time interval data type definitions used in this Standard

The real and time interval data types used in this Standard are discussed in B.1 and B.2.

#### B.1 Real data type

The declaration `real(10,7)` denotes a real data type with values that have precision to  $10^{-7}$  (i.e., 0.0000001).

For example, according to this type definition

- 5550.000001 and 5550.000002 are different values;
- 5550.000000001 and 5550.0 may evaluate to the same value, because the difference of 0.000000001 is too small to be accounted for according to the precision requirement of the type definition;
- 5550.0 and 5550.000000 are the same value; and
- 5550.0 and 5550 evaluate to the same value.

#### B.2 Time interval data type

The declaration `timeinterval(second,10,2)` denotes that the value for the data element `timeinterval` represents elapsed time with a precision of 0.01 seconds.

This Standard does not require implementations to distinguish between, for example, time intervals of 2.000 seconds and 2.001 seconds, because the difference of 0.001 seconds is less than the precision requirement for this data type.

This Standard recommends that bindings use a string representation conforming to ISO 8601:2000 to communicate the time interval value. However, this Standard does not specify a binding, and different bindings are possible for a value of this data type.

For example, if a binding uses real numbers to represent seconds

- A duration of exactly one hour can be expressed with the real value 3600.0;
- A duration of 2.5 seconds can be expressed with the real value 2.5; and
- A duration of 1 hour and 30 minutes can be expressed with the real value 5800.0.

If a binding uses ISO 8601:2000

- A duration of exactly one hour can be expressed with the string "PT1H";
- A duration of 2.5 seconds can be expressed with the string "PT2.5S"; and
- A duration of 1 hour and 30 minutes can be expressed with the string "PT1H30M".

The format for the string representations above is defined by the following pattern:

`P[yY][mM][dD][T[hH][nM][s[.s]S]]`

where

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

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

## Annex C

(informative)

### ISO 8601:2000 representation of the date time type

A string representation conforming to ISO 8601:2000 may be used to communicate the values of the `date_time_type` (see 6.2.3). This Standard does not specify a binding, and other bindings are possible.

For example, using a string representation conforming to ISO 8601:2000, the point in time July 16th, 1997, 30.17 seconds past 7:20 p.m. with a time offset of 1 hour with respect to UTC, can be expressed with the string

```
"1997-07-16T19:20:30.17+01:00"
```

where the format is defined by the following pattern:

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

where

- YYYY = four-digit year ( $\geq 0001$ );
- MM = two-digit month (01 through 12 where 01 = January, etc.);
- DD = two-digit day of month (01 through 31, depending on value of month and year);
- hh = two digits of hour (00 through 23) (am/pm NOT allowed);
- mm = two digits of minute (00 through 59);
- ss = two digits of second (00 through 59);
- s = one or more digits representing a decimal fraction of a second; and
- TZD = time zone designator ("Z" for UTC or +hh:mm or -hh:mm)

At least the four-digit year must be present. If additional parts of the value of the `date_time_type` are included, the character literals "-", "T", ":", and "." are parts of the character lexical representation for the value.

If the time portion is present, but the time zone designator is not present, the time zone is unspecified and the time is interpreted as "local time."