ADL SCORM Version 1.3 Application Profile

WORKING DRAFT 0.9

November 27, 2002

This page intentionally left blank.

Advanced Distributed Learning ADL SCORM Version 1.3 Application Profile

Available at
www.adlnet.org

For questions and comments visit the ADL Help & Info Center at ADLNet.org.

This page intentionally left blank.

Table of Contents

Abstract

The contents of this document shall be considered a draft. The document is subject to change based on work ongoing in various Standards and Specification working groups.

This document defines the SCORM Version 1.3 Application Profile. It is technical in nature and is meant for content developers, learning designers and learning management system (LMS) vendors. The document introduces the integration of the IMS Simple Sequencing specification, changes to the SCORM Run-Time Environment Data Model to support the integration of the IMS Simple Sequencing Specification and general clarification and cleanup. This document serves as a precursor to the SCORM Version 1.3.

This page intentionally left blank.

SECTION 1
Introduction

This page intentionally left blank.

1.1. Overview

The SCORM Version 1.3 Application Profile is being provided to allow the ADL Community the opportunity to perform test and evaluation activities on the changes being proposed before they are included in the final version of the SCORM Version 1.3.

The purpose of this document is to introduce the changes and additions to the SCORM Version 1.3. The additions and changes include the following:

SCORM Content Aggregation Model

- Content Model
    - New concept of Sharable Content Asset (SCA)
    - Addition of new SCORM Meta-data Components (SCA, Activity)
    - General cleanup and clarification
- Meta-data
    - Integration of changes to support IEEE 1484.12.1-2002 Learning Object Metadata
    - Updates to the SCORM Meta-data Information Model
    - Updates to the SCORM Meta-data XML Binding (*To be supplied*)
    - Updates to the SCORM Meta-data Application Profile (SCA and Activity)
    - General cleanup and clarification
- Content Packaging
    - Terminology changes for what learning resources are launchable (Asset – SCA)
    - Change to only permit leaf nodes to reference learning resources (Based on integration of IMS Simple Sequencing Specification)
    - Clarification on the use of the parameters attribute of an item (*To be supplied*)
    - Clarification on the use of the xml:base element (*To be supplied*)
    - General cleanup and clarification
- Sequencing
    - Integration of the IMS Simple Sequencing Specification
    - Introduction of a basic Navigation capability to compliment the sequencing specification

SCORM Run-Time Environment (RTE)

- Changes to the SCORM Run-Time Environment Data Model to supplement the integration of the IMS Simple Sequencing Specification
- Changes that require the mandatory implementation of all SCORM Run-Time Environment Data Model elements
- General cleanup and clarification

This Application Profile should be used to guide the ADL Community in the development of sequenced learning content that can be interchanged between systems using SCORM content packages. It is anticipated that the Sequencing Application Profile that this document defines will be incorporated into the SCORM Version 1.3.

The SCORM Version 1.3 Application Profile is in a Draft version. This document should still be considered a work in progress. Changes may occur due to ongoing work in IMS Simple Sequencing working group and comment period for this document. These changes, if any, will be reflected in supplemental releases of the SCORM Version 1.3 Application Profile as appropriate. ADL is asking that any feedback on the Application Profile or examples be submitted to the ADL Technical Team via a post to the SCORM Version 1.3 Application Profile Web board located in the Forums area of ADLNet.org(www.adlnet.org).

The IMS Simple Sequencing Specification, is a work in progress. At the time of publishing of this document, the IMS Simple Sequencing Specification is in “Public Draft” which means that it has not yet reached “Final” status. During the Public Draft period, there is an opportunity for evaluation and testing by the community at large. During the period of time that the IMS specification has been public, ADL has engaged in three parallel activities to evaluate and test the specification:

- A trial Sequencing Engine has been developed and integrated with the SCORM Version 1.3 (Beta) Sample Run-Time Environment.
- A number of content examples that illustrate sequencing behaviors have been created. These content examples have been tested and demonstrated in the SCORM Version 1.3 (Beta) Sample Run-Time Environment.
- This Application Profile has been written. The authoring of this document has been informed by ADL’s participation in the IMS Simple Sequencing Working Group.

1.1.1. How to Participate

The ADL Technical Team is asking vendors to participate in further developing and defining the SCORM Version 1.3 Application Profile in the following ways:

- **Implement the SCORM Version 1.3 Application Profile:** Use the SCORM Version 1.3 Application Profile, along with the IMS Simple Sequencing Specification, to develop sequencing support in LMS and content creation activities.
- **Apply application profile to your content use cases:** The IMS Simple Sequencing Working Group focused on a widely used set of sequencing behaviors found in a variety of pedagogical or instructional approaches such as classical behavioral training, discovery learning and collaborative problem solving. The ADL Technical Team is asking content developers to apply this application profile in conjunction with the IMS Simple Sequencing Specification during content creation. If content developers have use cases that the SCORM Version 1.3 Application Profile does not support, the ADL Technical Team asks that these be brought to our attention.

- **<u>Provide feedback via the ADL Forums:</u>** The ADL Technical Team is asking for any feedback and/or questions on the SCORM Version 1.3 Application Profile document and experiences implementing the Application Profile.

SECTION 2
The SCORM Content Aggregation Model

This page intentionally left blank.

2.1. Content Aggregation Model

The SCORM Content Aggregation Model represents a pedagogically neutral means for designers and implementers of instruction to aggregate learning resources for the purpose of delivering a desired learning experience. A learning resource is any representation of information that is used in support of a learning experience. Learning experiences consist of activities that are supported by electronic or non-electronic learning resources.

One step in the process of creating and delivering learning experiences involves the creation, discovery and aggregation of simple electronic assets into more complex learning resources and then organizing those resources into a predefined sequence of delivery. The SCORM Version 1.2 Content Aggregation Model did not support a robust sequencing model. With the addition of the IMS Simple Sequencing Specification, the SCORM Version 1.3 adds this key component to the Content Aggregation Model. The SCORM Version 1.2 Content Aggregation Model consists of the following:

- Content Model: Nomenclature defining the content components of a learning experience.
- Meta-data: A mechanism for describing specific instances of the components of the content model.
- Content Packaging: Defines how to represent the intended behavior of a learning experience (Content Structure) and how to package learning resources for movement between different environments (Content Packaging).

The SCORM Version 1.3 Content Aggregation Model incorporates the following changes to the Version 1.2 Content Aggregation Model::

- Addition of nomenclature derived from the IMS Simple Sequencing specification
- Addition of a Sequencing Definition Model, which is a rule-based model for defining how to sequence and deliver learning resources at run-time. This model defines a set of conditions and corresponding actions or behaviors that must be performed it the set of conditions evaluates to true.
- Meta-data changes based on the finalization of the IEEE 1484.12.1-2002 Learning Object Metadata (LOM) Standard

2.1.1. Content Model Overview

The SCORM Content Model describes the SCORM components used to build a learning experience from reusable learning resources. The Content Model also defines how these lower-level sharable and reusable learning resources are aggregated to compose higher-level units of instruction. The Content Model components are all considered specializations of learning resources. The SCORM Version 1.3 Application Profile introduces the following components: Assets, Sharable Content Assets (SCA), Sharable Content Objects (SCO) and Content Aggregations.

2.1.1.1 Assets

Learning content in its most basic form is composed of Assets that are electronic representations of media, text, images, sound, Web pages, assessment objects or other pieces of data that can be delivered to a Web client. An Asset can be described with Asset Meta-data (see Asset Meta-data definition below) to allow for search and discovery within online repositories, thereby enhancing opportunities for reuse. The mechanism for binding Assets to Asset Meta-data is the Content Package as described in the SCORM.

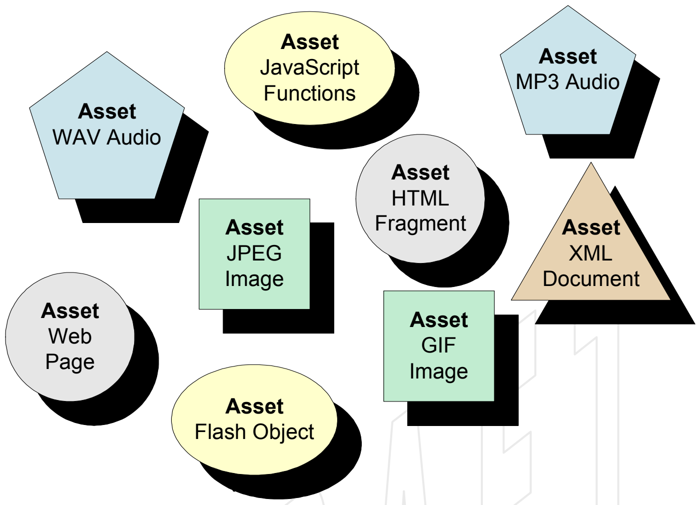


Figure 2.1.1.1a: Assets

2.1.1.2 Sharable Content Asset (SCA)

A Sharable Content Asset (SCA) is a new term being introduced in the SCORM Version 1.3, however, the concept of a SCA is not new. There was always some confusion on what a piece of content was called if the content did not communicate with an LMS. In the SCORM Version 1.2 this was referred to as an Asset. However, the term Asset was also defined as an electronic representations of media, text, images, sound, Web pages, assessment objects or other pieces of data that can be delivered to a Web client

A SCA is therefore defined as a collection of one or more Assets packaged as a single launchable resource. Figure 2.1.1.2a below shows an example of a SCA composed of several Assets. The SCA is different than the Sharable Content Object (SCO) in the sense that the SCA does not communicate with an LMS via the SCORM Communications API Adapter.

A SCA can be described with SCA Meta-data to allow for search and discovery within online repositories, thereby enhancing opportunities for reuse. The mechanism for binding SCAs to SCA Meta-data is the Content Package as described in the SCORM.

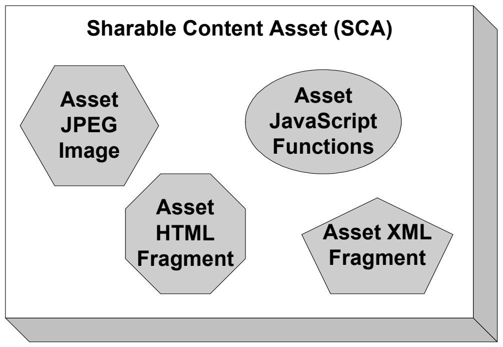


Figure 2.1.1.2a: Sharable Content Asset (SCA)

2.1.1.3 Sharable Content Object (SCO)

A SCO represents a collection of one or more Assets that include a single launchable resource that utilizes the SCORM Run-Time Environment to communicate with Learning Management Systems (LMSs). A SCO represents the lowest level of granularity of learning resources that can be tracked by an LMS using the SCORM Run-Time Environment Data Model. The only difference between a SCO and a SCA, is that the SCO communicates with an LMS using the SCORM Run-Time Environment Communications API. Figure 2.1.1.3a below shows an example of a SCO composed of several Assets.

To be reusable, a SCO by itself should be independent of learning context. For example, a SCO could be reused in different learning experiences to fulfill different learning objectives. In addition, one or more SCOs can be aggregated to form a higher-level unit of instruction or training that fulfills higher-level learning objectives.

SCOs are intended to be subjectively small units, such that potential reuse across multiple learning objectives is feasible. The SCORM does not impose any particular constraints on the exact size of a SCO. During content design and authoring activities, when determining the size of a SCO, thought should be given to the smallest logical size of content that one might desire to have tracked by a LMS at run-time. It is intended that the content developer will determine the size of the SCO based on how much information is needed to achieve the learning outcome and on the level of reuse that the content developer wishes to obtain.

A SCO can be described with SCO Meta-data (see SCO Meta-data definition below) to allow for search and discovery within online repositories, thereby enhancing opportunities for reuse. The mechanism for binding SCOs to SCO Meta-data is the Content Package as described in the SCORM.

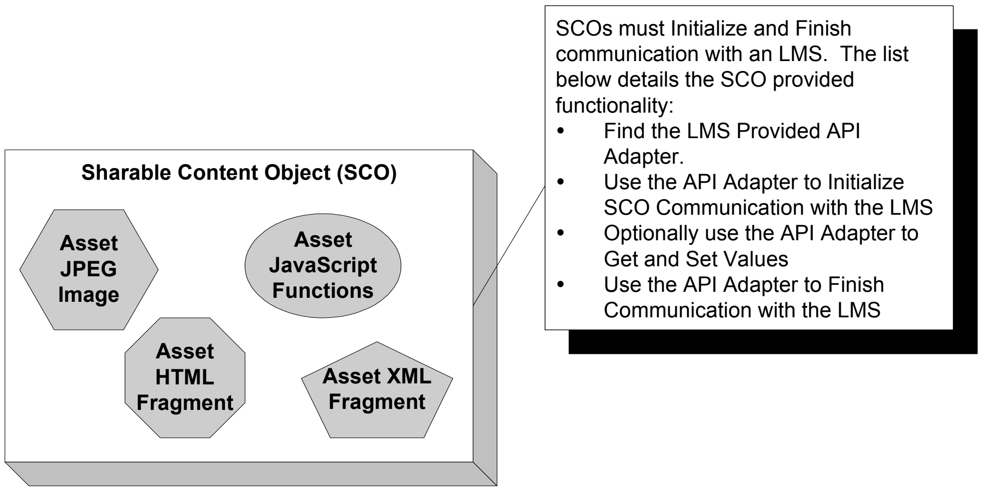

Sharable Content Object (SCO)
Asset JPEG Image
Asset JavaScript Functions
Asset HTML Fragment
Asset XML Fragment
SCOs must Initialize and Finish communication with an LMS. The list below details the SCO provided functionality:
• Find the LMS Provided API Adapter.
• Use the API Adapter to Initialize SCO Communication with the LMS
• Optionally use the API Adapter to Get and Set Values
• Use the API Adapter to Finish Communication with the LMS

Figure 2.1.1.3a: Sharable Content Object (SCO)

A SCO is required to adhere to the SCORM Run-Time Environment. This implies that it must have a means to locate an LMS's Communication API Adapter and must contain the minimum API calls (LMSInitialize("") and LMSFinish("")). There is no obligation to implement any of the other API calls as those are optional and depend upon the nature of the content.

The requirement that a SCO participate in the SCORM Run-Time Environment yields the following benefits:

- Any LMS that supports the SCORM Run-Time Environment can launch SCOs and track them, regardless of who generated them;
- Any LMS that supports the SCORM Run-Time Environment can track any SCO and know when it has been started and when it has ended; and
- Any LMS that supports the SCORM Run-Time Environment can launch any SCO in the same way.

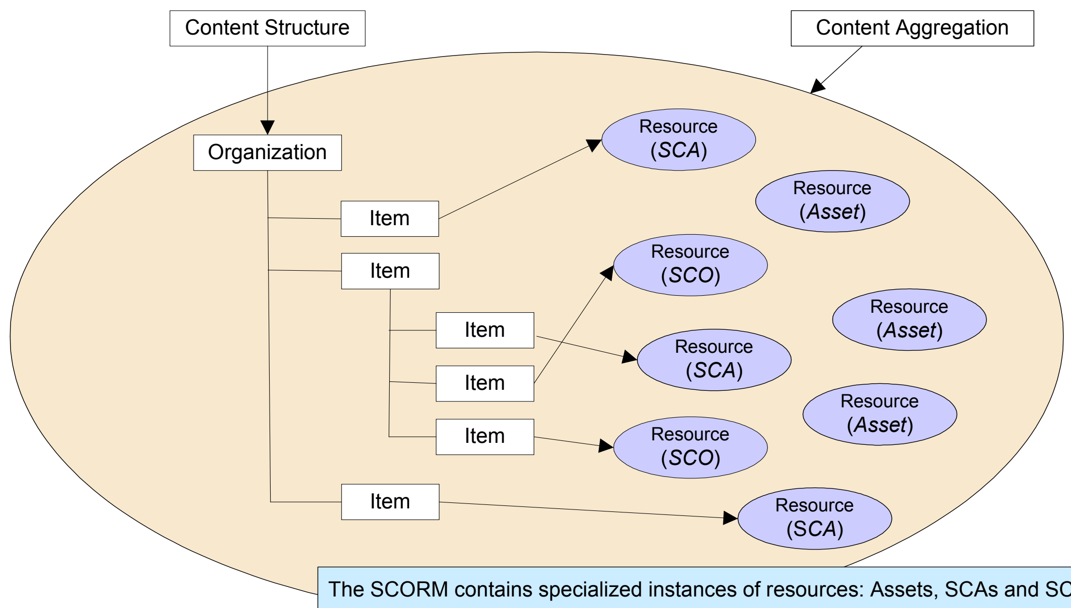
See the SCORM Version 1.2 Run-Time Environment for more details.

2.1.1.4 Content Aggregation

A Content Aggregation is a map (content structure) that can be used to aggregate learning resources into a cohesive unit of instruction (e.g. course, chapter, module, etc.), apply structure and associate learning taxonomies. The content structure defines the taxonomic representation of the learning resources. A content aggregation can reference Content Aggregation Meta-data to allow for search and discovery within online repositories, thereby enhancing opportunities for reuse.

The mechanism for binding content aggregations to Content Aggregation Meta-data is the Content Package as described in the SCORM. Figure 2.1.1.4a below shows an example of a Content Aggregation.

The Content Aggregation defines the content structure that provides the mechanisms for defining the sequence that learning resources are to be presented to the user.



The SCORM contains specialized instances of resources: Assets, SCAs and SCOs.

Assets are learning content in its most basic form. Assets are electronic representations of media, text, images, sound, web pages, chat sessions, assessment objects or other pieces of data that can be delivered to a Web client.

SCAs represent a collection of one or more Assets packaged as a single launchable resource. SCAs do not communicate with an LMS system.

SCOs represent a collection of one or more Assets packaged as a single launchable resource that utilizes the SCORM Run-Time Environment to communicate with LMSs. A SCO represents the lowest level of granularity of content that is able to be tracked by an LMS using the SCORM Run-Time Environment Data Model.

Content Aggregation is the process of pooling resources (Assets/SCAs/SCOs) into a defined structure (content structure) to build a particular learning experience.

Figure 2.1.1.4a: Content Aggregation

Navigation and sequencing between learning resources is defined in the content structure using sequencing rules associated with each learning resource or content aggregation. The LMS is responsible for interpreting the intended sequence described in the content structure and controlling the actual sequencing of the learning resources at run-time. (See Section 3 of this document for details on Sequencing)

This represents a major departure from the way courseware has been developed using stand-alone computer-based training (CBT) authoring tools. In the past, these tools typically embedded all of the navigation information that governs what part of the course the student will view next in proprietary data formats. In nearly all cases, authoring tools or systems defined and implemented proprietary and sometimes unique course sequencing methods. This means that it is difficult or impossible to share content between different authoring environments and equally difficult to reuse content in other contexts that involved different sequencing requirements.

Within the SCORM, which is deliberately browser-based, learning resource sequencing is defined in the content structure and is external to the learning resource. It is the responsibility of the LMS to launch the learning resources in the appropriate defined sequence at run-time. This is conceptually important because learning resource reuse cannot really happen if the learning resource has embedded information that is *context specific* to the course. In particular, if a learning resource contained a "hardwired" branching to another learning resource under specific conditions, it could not be used in a different course in which the second learning resource might not be applicable or available. The reusability of a learning resource depends on it being independent and not tied to a particular aggregation.

The SCORM recognizes, however, that some learning resources may contain internal logic to accomplish a particular learning task. Such a learning resource might branch within itself depending on user interactions. These branches are all self contained, relevant to a stand-alone learning resource (and therefore reusable) and are not usually visible to the LMS. Importantly, internal branching must not reference external learning resources that may or may not be present in other content aggregations. This is an important area that content developers should pay attention to when determining what learning resources should be used and how they are aggregated.

2.1.2. The SCORM Meta-data Components

Meta-data represents a mapping and recommended usage of the IEEE LTSC LOM elements for each of the SCORM Content Model components. The details of this mapping are provided in the SCORM. In general, guidance is provided for meta-data to be applied to Assets, SCAs, SCOs, Learning Activities and Content Aggregations to describe them in a consistent fashion such that they can be searched for and discovered within and across systems to further facilitate sharing and reuse.

Policies governing the application of meta-data to the components of the Content Aggregation Model should be defined within organizations that wish to enable reuse based on the requirements of those organizations. The SCORM does not seek to impose requirements related to the scope of meta-data tagging of Content Model components, but rather seeks to provide practical, standards-based guidance for those organizations wishing to enable sharing and reuse.

2.1.2.1 Content Aggregation Meta-data

Content Aggregation Meta-data describes the content aggregation. The purpose of applying Content Aggregation Meta-data is enable discoverability within, for example, a content repository and to provide descriptive information about the aggregated content represented by the content package as a whole, autonomous unit.

2.1.2.2 Activity Meta-data

Activity Meta-data describes the individual activities found in a content package. The purpose of applying Activity Meta-data is to make the activity accessible (enabling discovery) within a content repository. The meta-data should describe the activity as a whole. This is a new meta-data application profile for the SCORM Version 1.3. The requirements for any meta-data built for an activity shall match those requirements set forth in the content aggregation application profile.

2.1.2.3 SCO Meta-data

SCO meta-data can be applied to SCOs that provides descriptive information about the content represented in the SCO. This meta-data is used to facilitate reuse and discoverability of such content within, for example, a content repository.

2.1.2.4 SCA Meta-data

SCA meta-data can be applied to SCAs that provides descriptive information about the content represented in the SCA. This meta-data is used to facilitate reuse and discoverability of such content within, for example, a content repository.

NOTE: SCA is new terminology introduced in the scope of the SCORM Version 1.3. The SCO meta-data profiles discussed in the SCORM Version 1.2 should be used when describing a SCA in a context insensitive manner.

2.1.2.5 Asset Meta-data

A definition of meta-data that can be applied to "raw media" Assets that provides descriptive information about the Asset independent of any usage or potential usage within courseware content. This meta-data is used to facilitate reuse and discoverability, principally during content creation, of such Assets within, for example, a content repository.

2.1.2.6 Application of Meta-data

The mechanism for binding the Content Model components, discussed earlier, to the matching Meta-data application profile is the Content Package as described in the SCORM. There are currently five places meta-data can be applied with in a content package.

- **Manifest:** Meta-data at the manifest level must be conformant to the IEEE LTSC LOM binding but has no additional SCORM restrictions. This Meta-data is outside the scope of the SCORM and does not fall into one of the aforementioned categories.
- **Organization**: Meta-data at the organization level describes the content aggregation as a whole. This may be a course, unit, lesson or any other organized instructional unit. Meta-data placed at the organization level is SCORM *Content Aggregation Meta-data*.
- **Item:** Meta-data at the item level describes a nested hierarchy of SCOs and/or SCAs in a context sensitive manner. When associated with an item, the SCORM *Activity Meta-data* definition must be used.
- **Resource:** Meta-data at the resource level describes a SCO or SCA in a context insensitive manner. This meta-data is bound by either the SCORM *SCO Meta-*

data or SCORM *SCA Meta-data* definitions (Determined by the item's adlcp:scormtype attribute).

- **File:** Meta-data at the file level describes an Asset in a context insensitive manner. This meta-data is bound by the SCORM *Asset Meta-data* definition.

SECTION 3
Meta-data

This page intentionally left blank.

3.1. Meta-Data

This section provides specific guidance for applying meta-data to learning resources. The SCORM meta-data application profiles defined in this section directly reference the IEEE LTSC Learning Object Meta-data (LOM) standard. The IEEE LTSC LOM specification provides roughly 64 meta-data elements – more than would be practical for everyday use. This section defines, in the SCORM context, which data elements are mandatory in meta-data used for tagging assets, sharable content objects and aggregations (collections) of learning resources. While SCORM fully adheres to the IEEE LTSC LOM specification, this section provides additional specific guidance for using meta-data in SCORM conforming environments.

There are five basic subsections to the SCORM Meta-data section:

- **Section 3.1.1 Overview** provides background on the evolution of LOM.
- **Section 3.1.2 Information Model** defines the meta-data elements. This is essentially the "dictionary" of meta-data tags that define the intended usage of each element. The Information Model is the same as the IEEE LTSC LOM specification and does not define how to encode these tags in any particular way.
- **Section 3.1.3 The SCORM Meta-data XML Binding** is a work in progress and this section does not contain information at this time. When this section is completed an update to this document will be made available.
- **Section 3.1.4 The SCORM Meta-data Application Profiles** provides specific guidance for how to implement meta-data in the SCORM environment. It defines the SCORM mandatory elements and how they are to be encoded to be SCORM conformant.

3.1.1. Overview

The purpose of meta-data (data about data) is to provide a common nomenclature enabling learning resources to be described in a common way. Meta-data can be collected in catalogs or directly packaged with the learning resource it describes. Learning resources that are described with meta-data can be systematically searched for and retrieved for use and reuse.

3.1.1.1 The History of the SCORM Meta-data

Meta-data for learning resources has been under development within a number of national and international organizations over the past few years. ADL has looked to the IEEE LTSC Standard for Information Technology -- Education and Training Systems -- Learning Objects and Metadata Working Group, the IMS Global Learning Consortium, Inc. and the Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE) as the bodies that are defining meta-data specifically for learning resources. These groups, which have been working collaboratively, have developed a core set of specifications to which this document refers.

The SCORM references the 1484.12.1-2002 (IEEE LTSC LOM Standard). The LOM was developed as a result of a joint effort between the IMS Global Learning Consortium, Inc. and the ARIADNE to define a standard set of meta-data element definitions that can be used to describe learning resources. The SCORM has adopted the same set of meta-data elements described in the IEEE LTSC LOM Standard. The SCORM will also reference a binding specification at such time that the binding specification becomes available. The binding specification will provide an XML representation for the IEEE LTSC LOM Standard.

The SCORM applies the IMS meta-data element definitions to the three content model components: Asset, SCO and Content Aggregation. These three components define the meta-data portion of the SCORM Content Aggregation Model.

This mapping of standardized definitions from IEEE to the SCORM Content Aggregation Model provides the missing link between general specifications and specific content models. The following sections define the SCORM application of IEEE definitions to the meta-data portion of the SCORM Content Aggregation Model.

3.1.2. The 1484.12.1-2002 Information Model

The 1484.12.1-2002 (Learning Object Metadata Standard) Information Model describes the starting set of data elements that are defined to build SCORM conformant meta-data records. Along with the requirements defined in the information model, the SCORM defines application profiles for several types of meta-data instances. These requirements and definitions of the application profiles can be found in Section 3.1.4. SCORM conformant meta-data records may contain additional data elements, as described in Section 2.2.3.4 XML Extension Mechanism. The 1484.12.1-2002 Information Model is broken up into nine categories. These categories are based on the definitions found in the 1484.12.1-2002. The nine categories of meta-data elements are:

1. The *General* category groups the general information that describes the resource as a whole.
2. The *Lifecycle* category groups the features related to the history and current state of this resource and those who have affected this resource during its evolution.
3. The *Meta-metadata* category groups information about the meta-data record itself (rather than the resource that the record describes).
4. The *Technical* category groups the technical requirements and characteristics of the resource.
5. The *Educational* category groups the educational and pedagogic characteristics of the resource.
6. The *Rights* category groups the intellectual property rights and conditions of use for the resource.
7. The *Relation* category groups features that define the relationship between this resource and other targeted resources.

8. The *Annotation* category provides comments on the educational use of the resource and information on when and by whom the comments were created.
9. The *Classification* category describes where this resource falls within a particular classification system.

The 1484.12.1-2002 Information Model table lists the meta-data elements and how they are organized hierarchically. Each element is described with the following pieces of information:

- Nr: Hierarchical number system
- Name: Element name
- Explanation: Detailed description of the element
- Multiplicity: How many instances of the element are allowed within the immediate parent element.
- Data Type: Whether the element's value is textual, numerical or a date; and any constraints on its size and format. There are four general-purpose types used in the information model: LangString Type, Date Type, Duration Type and Vocabulary Type. The information model for each of these data types is specified following the SCORM Meta-data Information Model

Some elements use the term "smallest permitted maximum" in the multiplicity and/or data type columns. The smallest permitted maximum means that applications that process meta-data shall impose a maximum on the number of entries its supports, but that maximum shall not be lower than the smallest permitted maximum value.

For those elements that have a data type of a Vocabulary Type, additional information is provided on whether or not the vocabulary is a Restricted or Best Practice Vocabulary. Restricted indicates that the meta-data element is restricted to the vocabulary entries listed. Best Practice indicates that the SCORM recommends to use the listed vocabulary entries as the "best practice" for doing so.

14841.12.1-2002: Information Model				
Nr	**Name**	**Explanation**	**Multiplicity**	**Data Type**
1.	General	This category groups the general information that describes this learning object as a whole.	0 or 1	-
1.1.	Identifier	A globally unique label that identifies the learning object.	0 or More (smallest permitted maximum: 10)	-
1.1.1.	Catalog	The name or designator of the identification or cataloging scheme for this entry. A namespace scheme.	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)

1.1.2.	Entry	The value of the identifier within the identification or cataloging scheme that designates or identifies this learning object. A namespace specific string.	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)
1.2.	Title	Name given to this learning object.	0 or 1	LangString (smallest permitted maximum: 1000 characters)
1.3.	Language	The primary human language or languages used within this learning object to communicate to the intended user.	0 or More (smallest permitted maximum: 10)	CharacterString (smallest permitted maximum: 100 characters)
1.4.	Description	A textual description of the content of this learning object.	0 or More (smallest permitted maximum: 10)	LangString (smallest permitted maximum: 2000 characters)
1.5.	Keyword	A keyword or phrases describing the topic of this learning object. This data element should not be used for characteristics that can be described by other elements.	0 or More (smallest permitted maximum: 10)	LangString (smallest permitted maximum: 1000 characters)
1.6.	Coverage	The time, culture, geography or region to which this learning object applies. The extent or scope of the content of the learning object. Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges.	0 or More (smallest permitted maximum: 10)	LangString (smallest permitted maximum: 1000 characters)

1.7.	Structure	Underlying organizational structure of this learning object. **IEEE LOM Vocabulary:** • atomic - an object that is indivisible (in this context). • collection - a set of objects with no specified relationship between them. • networked - a set of objects with relationships that are unspecified. • hierarchical - a set of objects whose relationships can be represented by a tree structure. • linear - a set of objects that are fully ordered. Example: A set of objects that are connected by “previous” and “next” relationships.	0 or 1	Vocabulary (State) (**ADL Note:** Restricted)
1.8.	Aggregation Level	The functional granularity of this learning object. **IEEE LOM Vocabulary:** • 1: the smallest level of aggregation, e.g., raw media data or fragments. • 2: a collection of level 1 learning objects, e.g., a lesson. • 3: a collection of level 2 learning objects, e.g., a course. • 4: the largest level of granularity, e.g., a set of courses that lead to a certificate.	0 or 1	Vocabulary (Enumerated) (**ADL Note:** Restricted)
2.	Life Cycle	This category describes the history and current state of this learning object and those entities that have affected this learning object during its evolution.	0 or 1	-
2.1.	Version	The edition of this learning object..	0 or 1	LangString (smallest permitted maximum: 50 characters)
2.2.	Status	The completion status or condition of this learning object. **IEEE LOM Vocabulary:** • draft • final • revised • unavailable	0 or 1	Vocabulary (State) (**ADL Note:** Restricted)
2.3.	Contribute	Those entities (i.e., people, organizations) that have contributed to the state of this learning object during its life cycle (e.g., creation, edits, publication)..	0 or More (smallest permitted maximum: 30)	-

2.3.1.	Role	Kind of contribution. **IEEE LOM Vocabulary:** • author • publisher • unknown • initiator • terminator • validator • editor • graphical designer • technical implementer • content provider • technical validator • educational validator • script writer • instructional designer • subject matter expert	0 or 1	Vocabulary (State) (**ADL Note:** Best Practice)
2.3.2.	Entity	The identification of and information about entities (i.e., people, organizations) contributing to this learning object. The entities shall be ordered as most relevant first.	0 or More (smallest permitted maximum: 40)	CharacterString (smallest permitted maximum: 1000 characters)
2.3.3.	Date	The date of the contribution..	0 or 1	DateTime
3.	Meta-Metadata	This category describes this metadata record itself (rather than the learning object that this record describes). This category describes how the metadata instance can be identified, who created this metadata instance, how, when and with what references.	0 or 1	-
3.1.	Identifier	A globally unique label that identifies this metadata record.	0 or More (smallest permitted maximum: 10)	-
3.1.1.	Catalog	The name or designator of the identification or cataloging scheme for this entry. A namespace scheme.	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)
3.1.2.	Entry	Actual value of the identifier within the identification or cataloging scheme that designates or identifies this meta-data record.	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)

3.2.	Contribute	Those entities (i.e., people, organizations) that have affected the state of this metadata instance during its life cycle (e.g., creation, validation)	0 or More (smallest permitted maximum: 10)	-
3.2.1.	Role	Kind of contribution. Exactly one instance of this data element with value “creator” should exist. **IEEE LOM Vocabulary:** • creator • validator	0 or 1	Vocabulary (State) (**ADL Note:** Best Practice)
3.2.2.	Entity	The identification of and information about entities (i.e., people, organizations) contributing to this metadata instance. The entities shall be ordered most relevant first.	0 or More (smallest permitted maximum: 10)	CharacterString (smallest permitted maximum: 1000 characters)
3.2.3.	Date	The date of the contribution..	0 or 1	DateTime
3.3.	Metadata Scheme	The name and version of the authoritative specification used to create this metadata instance.	0 or More (smallest permitted maximum: 10)	CharacterString (smallest permitted maximum: 30 characters)
3.4.	Language	Language of this metadata instance. This is the default language for all LangString values in this metadata instance. If a value for this data element is not present in a metadata instance, then there is no default language for LangString values..	0 or 1	CharacterString (smallest permitted maximum: 100 characters)
4.	Technical	This category describes the technical requirements and characteristics of this learning object.	0 or 1	-
4.1.	Format	Technical datatype(s) of (all the components of) this learning object. This data element shall be used to identify the software needed to access the learning object.	0 or More (smallest permitted maximum: 40)	CharacterString (smallest permitted maximum: 500 characters)
4.2.	Size	The size of the digital learning object in bytes (octets). The size is represented as a decimal value (radix 10). Consequently, only the digits “0” through “9” should be used. The unit is bytes, not Mbytes, GB, etc. This data element shall refer to the actual size of this learning object. If the learning object is compressed, then this data element shall refer to the uncompressed size.	0 or 1	CharacterString (smallest permitted maximum: 30 characters)

4.3.	Location	A string that is used to access this learning object. It may be a location (e.g. Universal Resource Locator), or a method that resolves to a location (e.g. Universal Resource Identifier). The first element of this list shall be the preferable location.	0 or More (smallest permitted maximum: 10)	CharacterString (smallest permitted maximum: 1000 characters)
4.4.	Requirement	The technical capabilities necessary for using this learning object. If there are multiple requirements, then all are required, i.e. the logical connector is AND.	0 or More (smallest permitted maximum: 40)	-
4.4.1.	OrComposit e	Grouping of multiple requirements. The composite requirement is satisfied when one of the component requirements is satisfied, i.e., the logical connector is OR..	0 or More (smallest permitted maximum: 40)	-
4.4.2.	Type	The technology required to use this learning object, e.g., hardware, software, network, etc. **IEEE LOM Vocabulary:** • operating system • browser	0 or 1	Vocabulary (State) (**ADL Note:** Best Practice)
4.4.3.	Name	Name of the required technology to use this learning object. **IEEE LOM Vocabulary:** • If Type-"operating system", then: o pc-dos o ms-windows o macos o unix o multi-os o none • If Type="browser", then: o any o netscape communicator o ms-internet explorer o opera o amaya	0 or 1	Vocabulary (State) (**ADL Note:** Best Practice)
4.4.4.	Minimum Version	Lowest possible version of the required technology to use this learning object.	0 or 1	ChjaracterString (smallest permitted maximum: 30 characters)

4.4.5.	Maximum Version	Highest possible version of the required technology to use of this learning object.	0 or 1	CharacterString (smallest permitted maximum: 30 characters)
4.5.	Installation Remarks	Description of how to install this learning object.	0 or 1	LangString Type (smallest permitted maximum: 1000 characters)
4.6.	Other Platform Requirements	Information about other software and hardware requirements.	0 or 1	LangString (smallest permitted maximum: 1000 characters)
4.7.	Duration	Time continuous learning object takes when played at intended speed.	0 or 1	Duration
5.	Educational	This category describes the key educational or pedagogic characteristics of this learning object.	0 or 1 0 or More (smallest permitted maximum: 100)	-

5.1.	Interactivity Type	Predominant mode of learning supported by this learning object. **IEEE LOM Vocabulary:** • active • expositive • mixed "Active" learning (e.g., learning by doing) is supported by content that directly induces productive action by the learner. An active learning object prompts the learner for semantically meaningful input or for some other kind of productive action or decision, not necessarily performed within the learning object's framework. Active documents include simulations, questionnaires, and exercises. "Expositive" learning (e.g., passive learning) occurs when the learner's job mainly consists of absorbing the content exposed to him (generally through text, images, or sound). An expositive learning object displays information but does not prompt the learner for any semantically meaningful input. Expositive documents include essays, video clips, all kinds of graphical material, and hypertext documents. When a learning object blends the active and expositive interactivity types, then its interactivity type is "mixed".	0 or 1	Vocabulary (State) (**ADL Note:** Restricted)

5.2.	Learning Resource Type	Specific kind of learning object. The most dominant kind shall be first. The vocabulary is adapted for the specific purpose of *learning resource*s. **IEEE LOM Vocabulary:** • exercise • simulation • questionnaire • diagram • figure • graph • index • slide • table • narrative text • exam • experiment • problem statement • self assessment • lecture	0 or More (smallest permitted maximum: 10)	Vocabulary (State) (**ADL Note:** Best Practice)
5.3.	Interactivity Level	The degree of interactivity characterizing this learning object. Interactivity in this context refers to the degree to which the learner can influence the aspect or behavior of the learning object. **IEEE LOM Vocabulary:** • very low • low • medium • high • very high	0 or 1	Vocabulary (Enumerated) (**ADL Note:** Restricted)

5.4.	Semantic Density	The degree of conciseness of a learning object. The semantic density of a learning object may be estimated in terms of its size, span, or -- in the case of self-timed resources such as audio or video -- duration. The semantic density of a learning object is independent of its difficulty. It is best illustrated with examples of expositive material, although it can be used with active resources as well. **IEEE LOM Vocabulary:** • very low • low • medium • high • very high	0 or 1	Vocabulary (Enumerated) (**ADL Note:** Restricted)
5.5.	Intended End User Role	Principal user(s) for which this learning object was designed, most dominant first. **IEEE LOM Vocabulary:** • teacher • author • learner • manager	0 or More (smallest permitted maximum: 10)	Vocabulary (State) (**ADL Note:** Restricted)
5.6.	Context	The principal environment within which the learning and use of this learning object is intended to take place. **IEEE LOM Vocabulary:** • school • higher education • training • other	0 or More (smallest permitted maximum: 10)	Vocabulary (State) (**ADL Note:** Best Practice)
5.7.	Typical Age Range	Age of the typical intended user. This data element shall refer to developmental age, if that would be different from chronological age. When applicable, the string should be formatted as minimum age-maximum age or minimum age-. (NOTE: This is a compromise between adding three component elements (minimum age, maximum age and description) and having just a free text field.)	0 or More (smallest permitted maximum: 5)	LangString (smallest permitted maximum: 1000 characters)

5.8.	Difficulty	How hard it is to work with or through this learning object for the typical target audience. **IEEE LOM Vocabulary:** • very easy • easy • medium • difficult • very difficult	0 or 1	Vocabulary (Enumerated) (ADL Note: Restricted)
5.9.	Typical Learning Time	Approximate or typical time it takes to work with or through this learning object for the typical intended target audience.	0 or 1	Duration
5.10.	Description	Comments on how this learning object is to be used.	0 or More (smallest permitted maximum: 10)	LangString (smallest permitted maximum: 1000 characters)
5.11.	Language	The human language used by the typical intended user of the learning object.	0 or More (smallest permitted maximum: 10)	CharacterString (smallest permitted maximum: 100 characters)
6.	Rights	This category describes the intellectual property rights and conditions of use for this learning object.	0 or 1	-
6.1.	Cost	Whether use of the learning object requires payment. **IEEE LOM Vocabulary:** • yes • no	0 or 1	Vocabulary (State) (**ADL Note:** Restricted)
6.2.	Copyright and Other Restrictions	Whether copyright or other restrictions apply to the use of this learning object. **IEEE LOM Vocabulary:** • yes • no	0 or 1	Vocabulary (State) (**ADL Note:** Restricted)
6.3.	Description	Comments on the conditions of use of this learning object.	0 or 1	LangString (smallest permitted maximum: 1000 characters)

7.	Relation	This category defines the relationship between this learning object and other learning objects, if any. To define multiple relationships, there may be multiple instances of this category. If there is more than one target learning object, then each target shall have a new relationship instance.	0 or More (smallest permitted maximum: 100)	-
7.1.	Kind	Nature of the relationship between this learning object and the target learning object, identified by 7.2:Relation.Resource. **IEEE LOM Vocabulary**: • ispartof • haspart • isversionof • hasversion • isformatof • hasformat • references • isreferencedby • isbasedon • isbasisfor • requires • isrequiredby	0 or 1	Vocabulary (State) (**ADL Note:** Best Practice)
7.2.	Resource	The target learning object that this relationship references.	0 or 1	-
7.2.1.	Identifier	A globally unique identifier that identifies the target learning object.	0 or More (smallest permitted maximum: 10)	-
7.2.1.1.	Catalog	The name or designator of the identification or cataloging scheme for this entry. A namespace scheme.	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)
7.2.1.2.	Entry	The value of the identifier within the identification or cataloging scheme that designates or identifies the target learning object. A namespace specific string..	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)
7.2.2.	Description	Description of the target learning object.	0 or More (smallest permitted maximum: 10)	LangString (smallest permitted maximum: 1000 characters)

8.	Annotation	This category provides comments on the educational use of this learning object, and information on when and by whom the comments were created. This category enables educators to share their assessments of learning objects, suggestions for use, etc.	0 or More (smallest permitted maximum: 30)	-
8.1.	Entity	Entity (i.e., person, organization) that created this annotation.	0 or 1	CharacterString (smallest permitted maximum: 1000 characters)
8.2.	Date	Date that this annotation was created.	0 or 1	DateTime
8.3.	Description	The content of this annotation.	0 or 1	LangString (smallest permitted maximum: 1000 characters)
9.	Classification	This category describes where this learning object falls within a particular classification system. To define multiple classifications, there may be multiple instances of this category.	0 or More (smallest permitted maximum: 40)	-
9.1.	Purpose	The purpose of classifying this learning object. **IEEE LOM Vocabulary:** • discipline • idea • prerequisite • educational objective • accessibility • restrictions • educational level • skill level • security level • competency	0 or 1	Vocabulary (State) (**ADL Note:** Best Practice)
9.2.	Taxon Path	A taxonomic path in a specific classification system. Each succeeding level is a refinement in the definition of the higher level. There may be different paths in the same or different classifications, which describe the same characteristic. A taxonomy is a hierarchy of terms or phrases that are taxons.	0 or More (smallest permitted maximum: 15)	-

9.2.1.	Source	The name of the classification system. This element may use any recognized "official" taxonomy, or any user-defined taxonomy.	0 or 1	LangString (smallest permitted maximum: 1000 characters)
9.2.2.	Taxon	A particular term within a taxonomy. A taxon is a node that has a defined label or term. A taxon may also have an alphanumeric designation or identifier for standardized reference. Either or both the label and the entry may be used to designate a particular taxon. An ordered list of Taxons creates a taxonomic path, i.e. "taxonomic stairway": this is a path from a more general to more specific entry in a classification.	0 or More (smallest permitted maximum: 15)	-
9.2.2.1.	Id	The identifier of the taxon, such as a number or letter combination provided by the source of the taxonomy.)	0 or 1	CharacterString (smallest permitted maximum: 100 characters)
9.2.2.2.	Entry	The textual label of the taxon.	0 or 1	LangString (smallest permitted maximum: 500 characters)
9.3.	Description	Description of the learning object relative to the stated 9.1:Classification.Purpose of this specific classification, such as discipline, idea, skill level, educational objective, etc.	0 or 1	LangString (smallest permitted maximum: 2000 characters)
9.4.	Keyword	Keywords and phrases descriptive of the learning object relative to the stated 9.1:Classification.Purpose of this specific classification, such as accessibility, security level, etc., most relevant first.	0 or More (smallest permitted maximum: 40)	LangString (smallest permitted maximum: 1000 characters)

LangString Type Information Model				
Nr	**Name**	**Explanation**	**Multiplicity**	**Data Type**
1	Langstring	A datatype that represents one or more character strings. A LangString value may include multiple semantically equivalent character strings, such as translations or alternative descriptions.	0 or More (smallest permitted maximum: 10)	-
1.1	Language	Human language of the character string	0 or 1	CharacterString(smallest permitted maximum: 100 characters)

1.2	String	Actual string value.	0 or 1	CharacterString

DateTime Information Model				
Nr	**Name**	**Explanation**	**Multiplicity**	**Data Type**

1	DateTime	A point in time with accuracy at least as small as one second.
		
		YYYY[-MM][-DD[Thh[:mm[:ss[:s[TZD]]]]]]
		
		Where:
		
		YYYY = four digit year (>=0001)
		MM = two digit month (01 through 12)
		DD = two digit day of month (01 through 31)
		hh = two digit hour (00 through 23)
		mm = two digit minute (00 through 59)
		ss = two digit second (00 through 59)
		s = one or more digits representing a decimal fraction of a second
		TZD = time zone designator
		At least the four digit year must be present. If additional parts of the DateTime are included, the character literals "-", "T", ":" and "." are part of the character lexical representation for the datetime.
			0 or 1	CharacterString (smallest permitted maximum: 200 characters)

2	Description	Description of the date.	0 or 1	LangString (smallest permitted maximum: 1000 characters)

Duration Type Information Model				
Nr	**Name**	**Explanation**	**Multiplicity**	**Data Type**

1	Duration	An interval in time with accuracy at least as small as one second.
		
		P[yY][mM][dD][T[hH][nM][s[.s]S]]
		
		Where:
		
		y = number of years
		m = number of months
		d = number of days
		h = number of hours
		n = number of minutes
		s = number of seconds or fraction of seconds
		
		The character literal designators "P", "Y", "M", "D", "T", "H", "M", and "S" must appear if the corresponding nonzero value is present.
			0 or 1	CharacterString (smallest permitted maximum: 200 characters)

2	Description	Description of the duration.	0 or 1	LangString (smallest permitted maximum: 1000 characters)

Vocabulary Information Model				
Nr	**Name**	**Explanation**	**Multiplicity**	**Data Type**
1	Source	"LOMv1.0", or an indication of the source of the value, for instance through a URI.	0 or 1	CharacterString Type (smallest permitted maximum: 1000 characters)
2	Value	The actual value. If the source is "LOMv1.0", then the value shall come from the list defined in the LOMv1.0 Base Schema for the data element..	0 or 1	CharacterString Type (smallest permitted maximum: 1000 characters)

3.1.3. The SCORM Meta-data XML Binding

An XML binding is still being developed. Upon completion, this document will be updated and reissued.

3.1.4. The SCORM Meta-data Application Profiles

The SCORM Meta-data Application Profiles describe the integration of the IEEE LTSC LOM Specification within the ADL environment. The application profiles further define the types of meta-data and how they are applied to the content aggregation model described in Section 2.1. The SCORM identifies three types of learning content meta-data: Content Aggregation, SCO and Asset. The following section defines these meta-data application profiles.

Within the SCORM, the SCORM Meta-data Application Profiles are specializations of the IEEE LTSC LOM Specification . The SCORM imposes additional constraints on the application of the specification. Because of this specialization, the SCORM Meta-data Application Profiles fit nicely into the IMS Content Packaging Specification in all of the places that the IMS Content Packaging Specification already accounts for the use of meta-data.

Within the SCORM there are two main types of meta-data: context specific and context independent. Context specific meta-data describes learning content in a particular context – usually, this is a ‘learning’ context within a larger unit of content. Context independent meta-data is meta-data describing learning content in a context (or use) independent manner. The distinction between context specific and context independent meta-data is further specified in Section 2.3 Content Packaging.

3.1.4.1 Content Aggregation Meta-data

Content Aggregation Meta-data is meta-data that describes a content aggregation. This meta-data is used to facilitate reuse and discoverability within a content repository and to provide descriptive information about the content aggregation. Content Aggregation meta-data is information about a content aggregation as a whole that describes what it is for, who can use it, who controls it, etc., and information that can be searched externally such as the content aggregation title, description and version.

3.1.4.2 Activity Meta-data

Activity Meta-data is meta-data that can be applied to an activity that provides descriptive information about the activity independent of a particular context. This meta-data is used to facilitate reuse and discoverability of such activities within, for example, a content repository.

3.1.4.3 SCO Meta-data

SCO Meta-data is meta-data that can be applied to a SCO that provides descriptive information about the learning resource independent of a particular context. This meta-data is used to facilitate reuse and discoverability of such learning resources within, for example, a content repository. SCO meta-data is meta-data that describes a SCO that is not related to a specific content aggregation structure (i.e., context independent meta-data) and information that can be searched externally such as content title, description, date of creation and version.

3.1.4.4 SCA Meta-data

SCA Meta-data is meta-data that can be applied to a SCA that provides descriptive information about the learning resource independent of a particular context. This meta-data is used to facilitate reuse and discoverability of such learning resources within, for example, a content repository. SCA meta-data is meta-data that describes a SCA that is not related to a specific content aggregation structure (i.e., context independent meta-data) and information that can be searched externally such as content title, description, date of creation and version.

3.1.4.5 Asset Meta-data

Asset Meta-data is meta-data that can be applied to Assets such as illustrations, documents, or media streams that provides descriptive information about the Assets independent of learning content. This meta-data is used to facilitate reuse and discoverability principally during learning content creation of such Assets within, for example, a content repository. Asset meta-data is meta-data that describes Assets in a non-context specific way that can be searched externally by Asset title, description, date of creation and version and that can be used to create a searchable repository of sharable Assets.

3.1.4.6 The SCORM Meta-data Application Profile Requirements

The following table (Table 2.2.4.4a) defines the requirements for each of the above mentioned meta-data application profiles. Each of the meta-data application profiles is listed with the corresponding requirements for each of the meta-data elements. Note that these requirements do not imply that every Content Aggregation, SCO or Asset must be described by meta-data. However, the requirements do apply whenever meta-data is used to describe a Content Aggregation, SCO or Asset. An "M" indicates that the element is Mandatory. An "O" indicates that the element is Optional.

Name	Content Aggregation	Activity	SCO	SCA	Asset
1.0 General	M	M	M	M	M
1.1 Identifier	M	M	M	M	M
1.1.1 Catalog	M	M	M	M	M
1.1.2 Entry	M	M	M	M	M
1.2 Title	M	M	M	M	M
1.3 Language	O	O	O	O	O
1.4 Description	M	M	M	M	M
1.5 Keyword	M	M	M	M	O
1.6 Coverage	O	O	O	O	O
1.7 Structure	O	O	O	O	O
1.8 Aggregation Level	O	O	O	O	O
2.0 Life Cycle	M	M	M	M	O

2.1 Version	M	M	M	M	O
2.2 Status	M	M	M	M	O
2.3 Contribute	O	O	O	O	O
2.3.1 Role	O	O	O	O	O
2.3.2 Entity	O	O	O	O	O
2.3.3 Date	O	O	O	O	O
3.0 Meta-Metadata	M	M	M	M	M
3.1 Identifier	M	M	M	M	M
3.1.1 Catalog	M	M	M	M	M
3.1.2 Entry	M	M	M	M	M
3.2 Contribute	O	O	O	O	O
3.2.1 Role	O	O	O	O	O
3.2.2 Entity	O	O	O	O	O
3.2.3 Date	O	O	O	O	O
3.3 Metadata Scheme	M	M	M	M	M
3.4 Language	O	O	O	O	O
4.0 Technical	M	M	M	M	M
4.1 Format	M	M	M	M	M
4.2 Size	O	O	O	O	O
4.3 Location	M	M	M	M	M
4.4 Requirement	O	O	O	O	O
4.4.1 OrComposite	O	O	O	O	O
4.4.1.1 Type	O	O	O	O	O
4.4.1.2 Name	O	O	O	O	O
4.4.1.3 MinimumVersion	O	O	O	O	O
4.4.1.4 MaximumVersion	O	O	O	O	O
4.5 InstallationRemarks	O	O	O	O	O
4.6 Other Platform Requirements	O	O	O	O	O
4.7 Duration	O	O	O	O	O
5.0 Educational	O	O	O	O	O
5.1 Interactivity Type	O	O	O	O	O
5.2 Learning Resource Type	O	O	O	O	O
5.3 Interactivity Level	O	O	O	O	O
5.4 Semantic Density	O	O	O	O	O
5.5 Intended End User Role	O	O	O	O	O
5.6 Context	O	O	O	O	O
5.7 Typical Age Range	O	O	O	O	O
5.8 Difficulty	O	O	O	O	O
5.9 Typical Learning Time	O	O	O	O	O
5.10 Description	O	O	O	O	O
5.11 Language	O	O	O	O	O
6.0 Rights	M	M	M	M	M
6.1 Cost	M	M	M	M	M
6.2 Copyrights and Other Restrictions	M	M	M	M	M
6.3 Description	O	O	O	O	O
7.0 Relation	O	O	O	O	O
7.1 Kind	O	O	O	O	O
7.2 Resource	O	O	O	O	O
7.2.1 Identifier	O	O	O	O	O
7.2.1.1 Catalog	O	O	O	O	O
7.2.1.2 Entry	O	O	O	O	O
7.2.2 Description	O	O	O	O	O

8.0 Annotation	O	O	O	O	O
8.1 Entity	O	O	O	O	O
8.2 Date	O	O	O	O	O
8.3 Description	O	O	O	O	O
9.0 Classification	M	M	M	M	O
9.1 Purpose	M	M	M	M	O
9.2 Taxon Path	O	O	O	O	O
9.2.1 Source	O	O	O	O	O
9.2.2 Taxon	O	O	O	O	O
9.2.2.1 Id	O	O	O	O	O
9.2.2.2 Entry	O	O	O	O	O
9.3 Description	M	M	M	M	O
9.4 Keyword	M	M	M	M	O

Table 2.2.4.4a: SCORM Meta-data Application Profile Requirements

SECTION 4
Introduction to Sequencing

This page intentionally left blank.

4.1. Introduction

The majority of this document is based on the IMS Simple Sequencing Specification that defines a method for representing the intended behavior of an authored learning experience such that any LMS can sequence discrete learning activities in a consistent way. A learning designer or content developer declares the relative order in which elements of content (SCOs or SCAs) are to be presented to the learner and the conditions under which a piece of content is selected and delivered or skipped during presentation.

This document defines the required behaviors and functionality that conforming systems must implement. It incorporates rules that describe the branching or flow of learning activities through content according to the outcomes of a learner's interactions with content. This representation of intended instructional flow may be created manually or with authoring systems, that produce output that conforms to this profile. While learning content developers need to know how to create and describe content sequences, authoring systems may hide the details of the models presented in this profile. The representation of sequencing may be interchanged between systems designed to deliver instructional activities to learners. The components of an LMS used to execute the specified rules and behaviors, when content is delivered to a learner, are referred to as a 'sequencing engine'.

IMS Simple Sequencing, the basis of this document, is labeled as *simple* because it includes a limited number of widely used sequencing behaviors, not because the specification itself is simple. Simple Sequencing is not all-inclusive. In particular, Simple Sequencing does not address, but does not necessarily preclude, artificial intelligence-based sequencing, schedule-based sequencing, sequencing requiring data from closed external systems and services (e.g., sequencing of embedded simulations), collaborative learning, customized learning, or synchronization between multiple parallel learning activities.

Simple Sequencing recognizes only the role of the learner and does not define sequencing capabilities that utilize or are dependent on other actors, such as instructors, mentors, or peers. This Application Profile does not prohibit usage in contexts involving other actors; however, it does not define the roles of other actors or sequencing behaviors that result from participation of other actors. In addition, issues such as content look and feel and presentation style are not defined by this document.

The SCORM Version 1.3 Application Profile provides an external representation, via extensions to the SCORM Packaging Application Profile for Content Aggregation Packages, to exchange sequencing descriptions between different run-time components or LMSs. Thus, sequencing is based on the same content organization and tree structure as the Content Packaging Specification.

4.2. Sequencing Terminology

The following sections detail sequencing vocabulary essential in understanding the models introduced in this document. Each description is based on the IMS Simple Sequencing interpretation of the term.

4.2.1. Learning Activities/Activities

IMS Simple Sequencing Specification relies on the concept of *learning activities*. A learning activity (Figure 3.2.1a) may be loosely described as an instructional event or events embedded in a content resource or as an aggregation of activities that eventually resolve to discrete content resources with their contained instructional events. A learning activity may use a learning resource, or it may consist of several sub-activities.

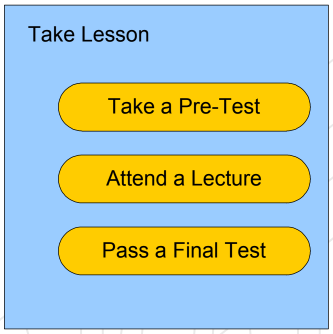


Figure 4.2.1a: Activity Illustration

In the illustration above the "Take Lesson" activity is composed of three sub-activities, "Take a Pre-Test", "Attend a Lecture" and "Pass a Final Test". The sub-activities are experienced in the context of the "Take Lesson" activity. This implies that there is a defined hierarchy of activities and that you cannot "Take a Pre-Test" without attempting the "Take Lesson" activity. The sub-activities may be made up of additional sub-activities or reference learning resources such as Sharable Content Objects (SCOs) or Sharable Content Assets (SCAs).

While participating in a learning experience, a learner will experience various learning activities. The LMS will deliver learning activities in a sequence determined at run-time, based on the progress made by the learner in previous learning activities, learner intention and the authored sequencing directions. Learning activities have the following characteristics:

- Learning activities have a discrete start and end
- Learning activities have well-defined completion and mastery conditions
- Learning activities can consist of sub-activities, nested to any depth
- Learning activities occur in context of their parent activity, if one exists

- Learning activities may or may not have associated learning resources (SCOs or SCAs)

SCORM components are experienced by a learner in context of a learning activity. When a learner attempts a learning activity, the learner will experience the SCORM component associated with that learning activity, if one exists. Associating SCORM components to learning activities is done through a SCORM content package.

- Both **SCAs** and **SCO**s can be associated with a learning activity.
- **Content Aggregations** consist of a set of learning activities

4.2.2. Tracking Activities

For each activity attempted by the learner, that activity shall have associated tracking status data. Learner interactions with an activity may affect its tracking status data. Tracking status data is used during sequencing to affect to sequencing process (See Section 7.1.1 for more details).

4.2.2.1 Data Persistence

Simple Sequencing does not specify how data (e.g., tracking data) is to be persisted across multiple instantiations of the overall sequencing process for a particular learner and activity tree or learning experience, e.g., across multiple login sessions. It is necessary to persist control, tracking and state data at least until the current attempt on the activity tree is terminated. Such an attempt may include one or more sessions. LMS policies govern whether to persist data beyond that time, such policies are beyond the scope of the application profile.

4.2.2.2 Objectives

Learning objectives are separate from learning activities in the IMS Simple Sequencing Specification. Learning objectives represent a set of locally and globally scoped data items, each with a satisfaction status and a satisfaction measure. Simple Sequencing makes no assumption as to how to interpret the objective (e.g., is it a competency, is it a mastery, or is it simply a shared value?). Activities may have more than one associated local objective and may reference multiple globally shared objectives. Multiple activities may reference the same global objective, thus sharing the data values.

The resolution of local and global objective IDs is not specified. An objective may be shared within a single activity tree or may be shared globally across multiple instantiations of tracking models. The lifetime of shared global objectives and the scope of sharing is not specified; it is determined by implementations. Activities may reference multiple objectives, thus providing a mechanism for activities to have subobjectives. However, the application profile makes no assumptions about the semantics or meanings of multiple objectives associated with an activity.

4.2.3. Active/Passive Content

Simple Sequencing relies on values within the tracking status model to control sequencing. Simple Sequencing differentiates between active and passive content and supports both active and passive content on an activity-by-activity basis. Active content is responsible for setting elements of the tracking status model via the SCORM Run-Time Environment Data Model (See Section 9 for details). For passive content, Simple Sequencing will automatically set certain values in the tracking model, based on defined default values.

4.2.4. Activity Tree

The SCORM Content Aggregation Model defines a structure that provides for the hierarchical organization of learning content. This is in the form of the content packaging organization that allows for the definition of a hierarchy of "items".

Each item in the structured hierarchy represents an instructionally relevant unit of information. Items can be nested to any arbitrary depth and can have learning taxonomy labels applied to them. For example, an item can represent a course, a module, a unit, a lesson, etc. In terms of sequencing, items correspond to discrete learning activities. Sequencing definition elements can be applied to activities by a content author to define a specific sequencing run-time behavior consistent with the desired learning experience. "Activity" is a generic term used in this specification to refer to a discreet portion of a learning experience.

This hierarchical structure has traditionally been specified in terms of a content organization for the purpose of interchanging content. This is the purpose of the concept of the content packaging manifest organization. Because the sequencing definition model as well as the run-time behaviors that result are relevant only in the context of a meaningful content structure, it is necessary for the specification to define a common, conceptual structure.

The SCORM Version 1.3 Application Profile and the IMS Simple Sequencing Specification refer to a hierarchical organization of activities as "Activity Tree". The Activity Tree is a (proprietary) internal representation of the content hierarchy. It does not have to be represented as a "tree". Its soul purpose is to provide a conceptual "anchoring" point for the behavior descriptions . The Activity Tree can be considered a generic, conceptual structure that represents a hierarchy of authored learning activities.

An Activity Tree is used in the following manner:

1. The Activity Tree represents the data structure that results from the content authoring and aggregation process. An authoring tool may for example implement an internal data structure that represents this hierarchy in a proprietary format. This structure results from whatever instructional design process or method the author or designer engaged in to create the intended learning experience.

2. The Activity Tree represents the data structure that an LMS implements that reflects the hierarchical, internal representation of the defined learning activities, including the tracking status information for each activity in the hierarchy on a per learner basis. In this context, the sequencing definition information that was defined by the content author determines the manner in which the LMS determines the relative sequence of learning activities, as well as the eligibility for learning activities to be attempted by a learner on a conditional basis.

It is anticipated, but not required, that systems that implement sequencing will have an internal, proprietary representation of the "Activity Tree".

The Activity Tree allows this application profile to describe informational and processing requirements, such as sequencing algorithms and behaviors in a non-implementation specific manner. Again, it is **not** the intention of the SCORM Version 1.3 Application Profile to mandate that authoring tools and LMSs implement Activity Trees, or that all instructional design methodologies be modified to focus on an Activity Tree as the final product of these methods. Rather, Activity Tree is the general term that represents hierarchical representations of learning activities and corresponding learning content.

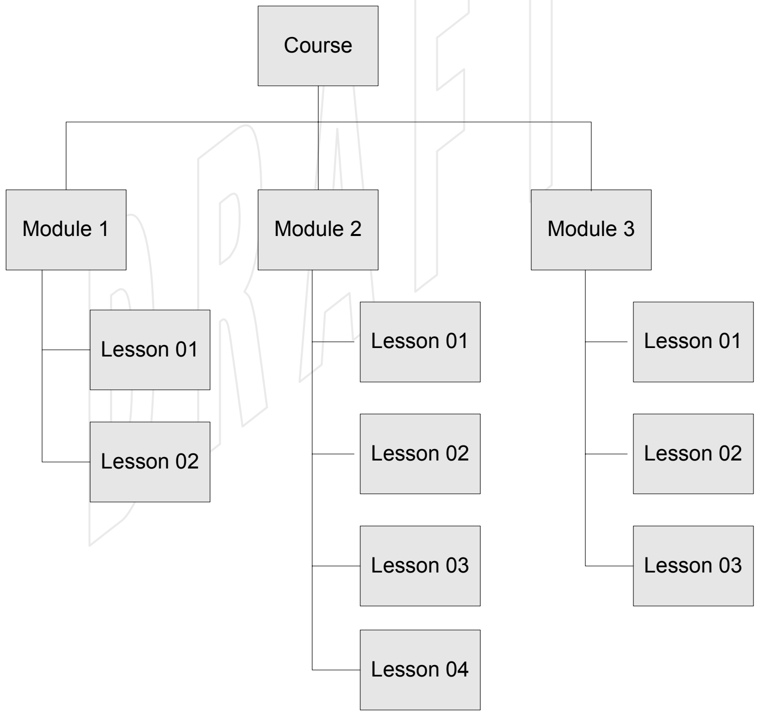


Figure 4.2.4a: A representation of an Activity Tree

4.2.5. Cluster

A cluster is an organized aggregation of activities consisting of a single parent activity and its first level children, but not the descendants of its children. The cluster is considered the basic sequencing building block. Sets of clusters are combined to form an activity tree. The parent activity of a cluster will contain the information about the sequencing strategy for the cluster. The parent activity in a cluster can be used to collect a group of child activities and the information about how the child activities are to be sequenced.

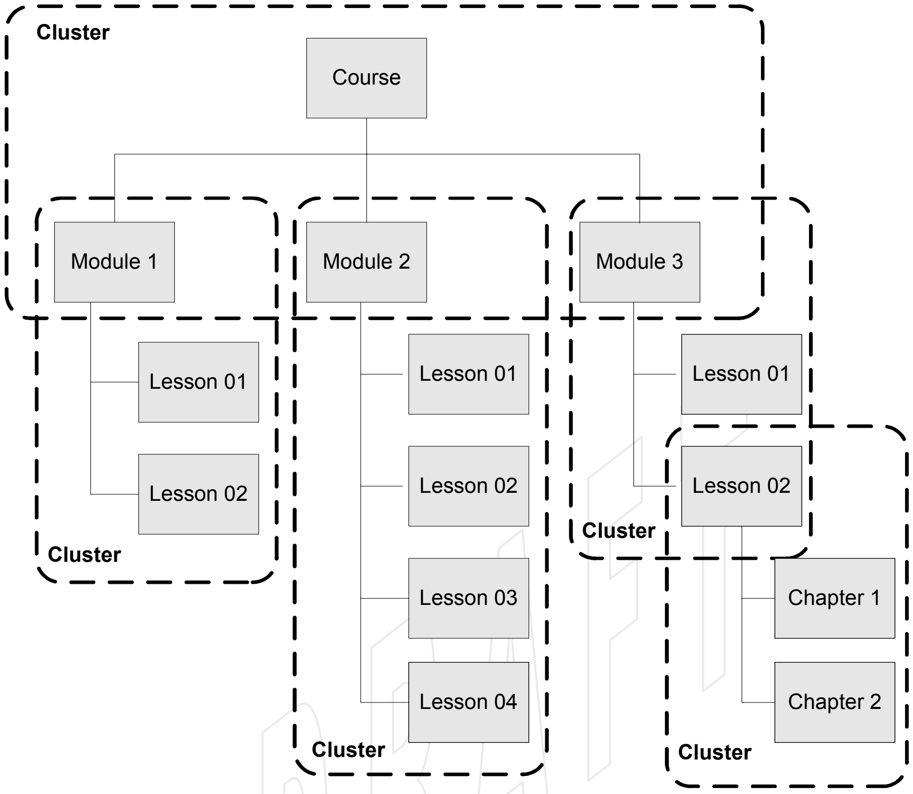


Figure 4.2.5a: What is a cluster?

Figure 3.2.3a represents a simple course, with three modules. The course is the parent of a cluster that contains all three of the modules, but not the lessons. Each module is also the parent of its own cluster consisting of itself and it's children. Module 1 is the parent of its cluster, consisting of Lesson 01 and 02.

4.2.6. Activity Duration Tracking

The learning experience temporal model, depicted below, captures a learning experience over time. A learning experience is presented to a learner in the form of attempts on a learning activity. An attempt on an activity can be accomplished across more than one session. These terms are used throughout the SCORM Version 1.3 Application Profile and are described in more detail below.

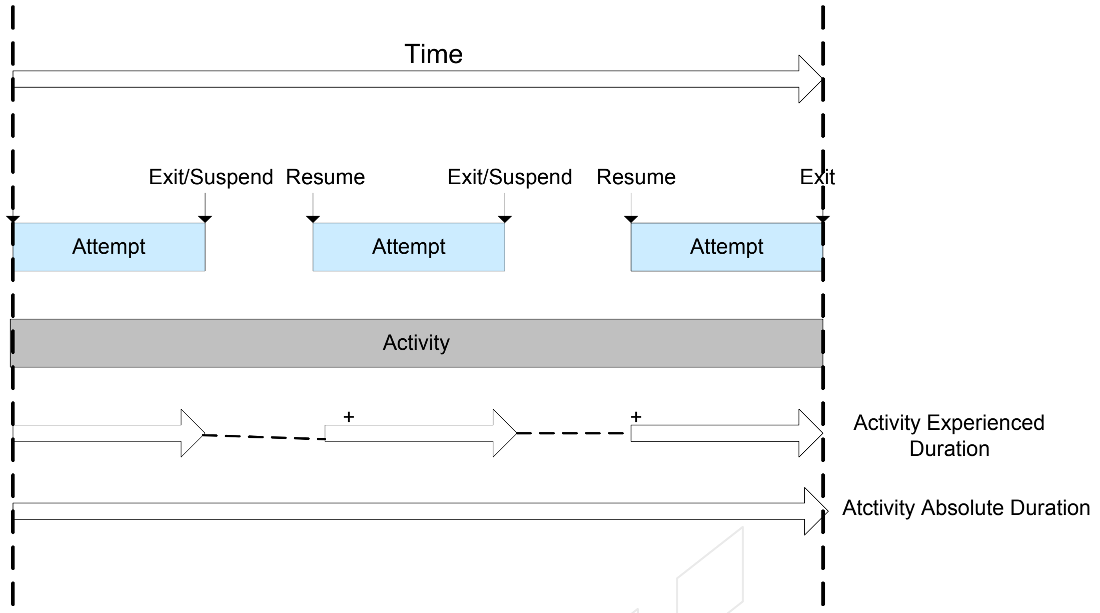


Figure 4.2.6a: Activity Duration Tracking

4.2.6.1 Attempt

An attempt is defined as an effort to complete a learning activity. Attempts on activities always occur within the context of an attempts on their parent activities. It is important to note that one and only one leaf activity can be attempted at any given time and all of the leaf activity's ancestors (through the root) are also being attempted when the leaf activity is being attempted. It may not always be possible to complete an attempt in a single session. There are many situations where a learner may have to suspend the learning activity and resume at a later time. In most cases, this should be a continuation of the current attempt, rather than a new attempt. As the result of a learner attempt on an activity, or some outside administrative action, the tracking status for an activity can change (See Section 7.1.1 for Tracking Status Information). When the tracking status of any activity changes, the effects of this change on its parent activity must be available before any subsequent sequencing requests that involve evaluation of that activity can be processed. Each time an attempt on an activity is terminated for any reason, the affects of any tracking status changes on the rest of the activity tree must be available before another sequencing request can be processed.

4.2.6.2 Session

A session is defined to be a continuous experience of a learning activity that is not interrupted by suspending the activity. An attempt to complete an activity may take several sessions. Note that in this context, "Session" is NOT analogous to a user login session.

4.2.6.3 Suspending and Resuming Activities

An attempt on an activity may be suspended and later resumed, resuming a suspended activity does not count as a new attempt on that activity. Other activities may be attempted while the activity is suspended. More than one activity may be suspended at any given time.

The current sequencing session may be suspended and later resumed at the last activity experienced by the learner.

4.2.6.4 Starting and Stopping the Sequencing Process

Simple Sequencing does not specify how to start the overall sequencing process or how to stop the process. Generally, the LMS will recognize some event, e.g., a course login, to start sequencing. Some other event, e.g., a logout, is mapped to the appropriate navigation, exit and sequencing requests, after which the LMS may terminate the overall sequencing process. Navigation events trigger navigation requests that are acted on by a sequencing engine.

4.2.7. Sequencing Definition Model

The Sequencing Definition Model defines a set of elements that can be used by content developers to define specific sequencing behavior. Conforming LMSs must support the behaviors that result from the definition of all of these elements. The definition model allows the association of specific elements with learning activities and their associated resources. Each information model element has a default value that is to be assumed in the absence of an explicit, content developer-defined value.

4.2.8. Content Types

The SCORM Sequencing Application Profile is somewhat independent of the types of learning content and learning objects and can be used to sequence all types of content. For example, content may include simple static web pages, Multipurpose Internet Mail Extensions (MIME) resources of any type (e.g., DOC, PDF files), services and proxies for services and dynamically created objects. In particular, content need not use the SCORM Communications API used by SCOs. Some tracking model values will coincide with SCORM Run-Time Data Model elements, but this is not a requirement in order to leverage sequencing. However, the SCORM Run-Time should be used to leverage the more complex models made possible by the SCORM Version 1.3 Application Profile.

SECTION 5
The Sequencing Definition Model

This page intentionally left blank.

5.1. SCORM Sequencing Definition Model

The SCORM Sequencing Definition Model is based directly on the IMS Simple Sequencing Definition Model. The SCORM Sequencing Definition Model prescribes additional application profile specific behaviors and restrictions beyond those currently defined by the IMS Simple Sequencing Specification.

The definition model defines a set of elements that may be used by content developers to define specific sequencing behavior. Conforming LMSs must support the behaviors that result from the definition of all of these elements. The definition model allows the association of specific elements with learning activities and their associated resources. Each information model element has a default value that is to be assumed by any sequencing system in the absence of an explicit, content developer-defined value.

The Sequencing Definition Model defines the following categories:

- Sequencing Control Modes
- Sequencing Rules
- Limit Conditions
- Auxiliary Resource
- Rollup Rules
- Objectives
- Objective Map
- Rollup Controls
- Selection Controls
- Randomization Controls
- Delivery Controls

5.1.1. Sequencing Control Modes

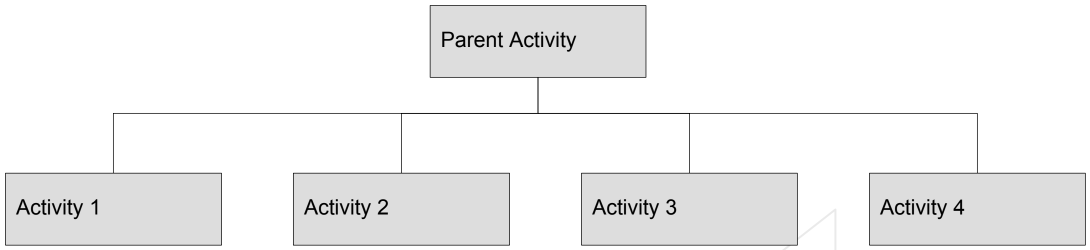
The Sequencing Control Modes allow for the content developer to affect various sequencing behaviors, as needed, to reflect the desired learning experience. The control modes are used in the following cases:

- During processing of a navigation request (Navigation Request Behavior) to determine whether or not the request will translate into a valid sequencing request,
- During various sequencing sub-processes to affect the resultant delivery request that will be returned by those sequencing sub-processes, and
- During various sequencing sub-processes to affect the various states of the tracking status data model elements (*Use Current Attempt Objective Progress Information* and *Use Current Attempt Progress Information*)

The Control Modes are defined to allow the content developer to control the sequencing behavior for a cluster. Multiple modes can be enabled simultaneously, for a cluster, to create combinations of sequencing control behaviors.

The following table describes the set of sequencing control modes defined by the IMS Simple Sequencing Specification.

5.1.1.1 Sequencing Control - Choice



The *Sequencing Control Choice* control mode indicates that the learner is free to choose any activity in any order. ***Any*** activity whose parent has *Sequencing Control Choice* defined to be True is a valid target for a "Choice" navigation request. When the learner chooses any activity, the activities associated learning resource should be delivered to the learner, unless another sequencing rule prevents the delivery of the chosen activity. When choice control mode is enabled, it is assumed that there is a means for the learner to trigger a "Choice" navigation event that results in a "Choice" sequencing request with a specified target activity included with the sequencing request. This implies a requirement of an LMS to provide a user interface control such as a "menu", "map" or "table of contents" to permit the "Choice" navigation event.

The *Sequencing Control Choice* control mode is always enabled unless explicitly disabled through a directive in the sequencing instructions (setting Choice to "false").

As mentioned above, any activity whose parent has *Sequencing Control Choice* defined to be True is a valid target for a "Choice" navigation request. If the learner chooses an activity that has sub-activities (the activity is a cluster which as defined in the Content Aggregation Model, has no associated learning resource), one of only two actions can be taken by the sequencing process:

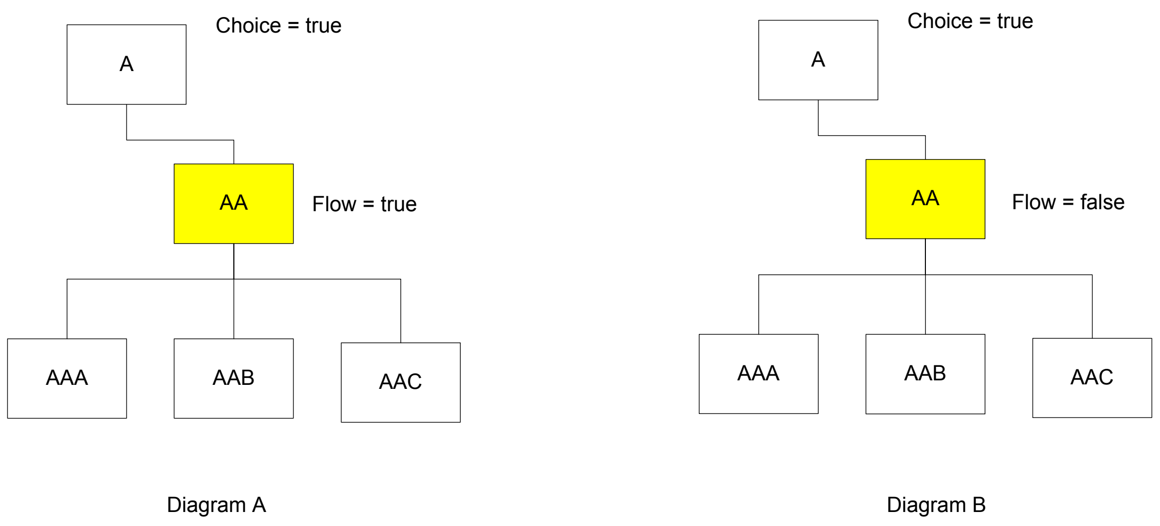


Figure 5.1.1.1a: Choice illustration

As depicted in Diagram A, the target of the choice navigation request has a *Sequencing Control Choice* defined to be True. This requires the set of sub-activities to be searched, in a pre-order traversal, until one is found that has a learning resource, and that learning resource is identified to be delivered.

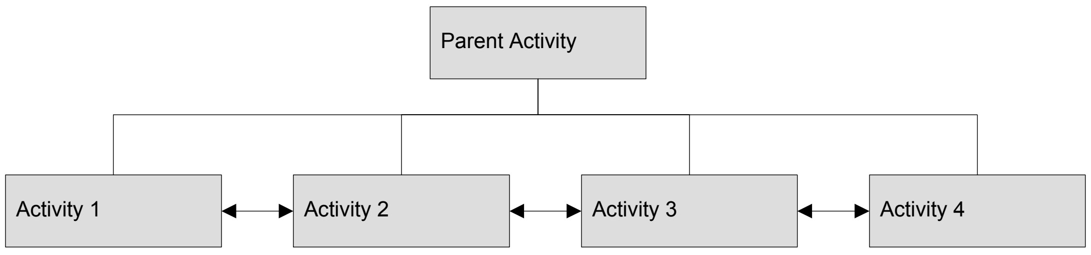
As depicted in Diagram B, the target of the choice navigation request has a Sequencing Control Choice defined as False. This requires the sequencing process to adhere to the interstitial process (*To Be Supplied*).

When an activity finishes, nothing happens until the learner chooses another activity or quits the main activity. If the learner chooses another activity while an activity is in progress, see Sequencing Control Choice Exit for more detail, and that chosen activity is available for delivery, the current activity's learning resource is terminated, and the chosen activity is attempted.

5.1.1.1.1 Sequencing Control – Choice Exit

The *Sequencing Control Choice Exit*, hereafter referred to as ChoiceExit, control mode should be used to indicate to a sequencing engine that a Choice sequencing request, triggered while a child of the activity is the current activity, is permitted. The sequencing request shall cause the current activity to *Exit* and the new chosen activity is attempted.

5.1.1.1.2 Sequencing Control - Flow

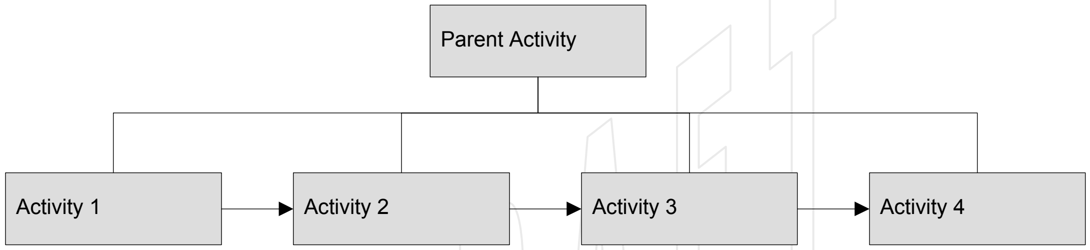


The *Sequencing Control Flow* control mode enables guided sequencing through the child activities of the activity for which it is defined. The Flow control mode requires that the *Continue* and *Previous* sequencing requests are enabled for an LMS to process. If the Flow control mode is enabled for a cluster of activities and the content author has not indicated that the content provides its own mechanism for issuing a navigation event, an LMS is required to provide a mechanism to permit the learner to "Continue" to the next activity or go back to a "Previous" activity.

If a Flow control mode is enabled for a cluster of activities:

1. If the learning activity does not indicate that the content is providing its own navigation buttons, the LMS (or delivery system), shall provide a user interface control (such as Continue and Previous buttons) to allow the learner to trigger a "Continue" or "Previous" navigation event.

5.1.1.1.3 Sequencing Control - Forward Only



The Sequencing Control Forward Only, hereafter referred to as ForwardOnly, element represents a rule that constrains sequencing among the children of the node for which the rule is defined. This element contains a true/false value. If ForwardOnly is "true" for a node, traversal of the children of that node is always in forward order. For example, when re-entering that node as the result of a Previous sequencing request from a subsequent node in traversal order of the tree, the first child node, rather than the last child node, will be traversed first.

If the rule is defined on a parent node (activity) the following requirements shall be adhered to:

If the *Sequencing Control Forward Only* is set to True:

1. If the *Sequencing Control Flow* is set to True, a backward direction is not allowed among the children of the node, where the children are those learning activities that are one-level down from the parent learning activity only.
2. *Previous* sequencing requests are ignored where the target node is a child of the node.
3. If the Sequencing Control Choice is set to True, sequencing requests where the target node is a child node in a backward direction from the currently delivered child node are ignored.

If the *Sequencing Control Forward Only* is set to False:

1. No restrictions are placed on *Previous* or *Choose* sequencing requests.

5.1.1.1.4 Use Current Attempt Objective Progress Information

This element indicates that the objective progress information for the children of the activity will be used (True or False) in rule evaluations and rollup if that information was recorded during the current attempt of the activity. The default value for this element is True.

5.1.1.1.5 Use Current Attempt Progress Information

This element indicates that the attempt progress information for the children of the activity will be used (True or False) in rule evaluations and rollup if that information was recorded during the current attempt of the activity. The default value for this element is True.

5.1.2. Sequencing Rules

The IMS Simple Sequencing specification employs a rule-based sequencing model. A set of zero or more rules can be defined for an activity and must be evaluated at various times in the sequencing and delivery process during run-time. Implementations are free to evaluate defined rules at any time for other purposes, but at a minimum the rules that are defined must be evaluated at certain times during the conceptual sequencing and delivery processes (See Sequencing Behaviors). The application of sequencing rules to an activity may alter the default sequencing path or may affect the availability of the activity and/or it's sub-activities for delivery.

Sequencing rules consist of a set of conditions and a corresponding action or behavior that is performed if the set of conditions evaluates to true (*if [condition_set] then [action/behavior]*). Sequencing rule conditions can be based on the Tracking Status Data Model of an activity including *Objective Progress Information*, *Activity/Attempt Progress Information*, or on the status of an activity with regard to defined *Limit Conditions*. The IMS Simple Sequencing specification defines a Tracking Status Data Model that defines the elements on which sequencing rule conditions can be based. Individual conditions can be combined to create sets of conditions for evaluation such that one individual condition must be true, or all conditions must be true in order for the resulting behavior to take effect.

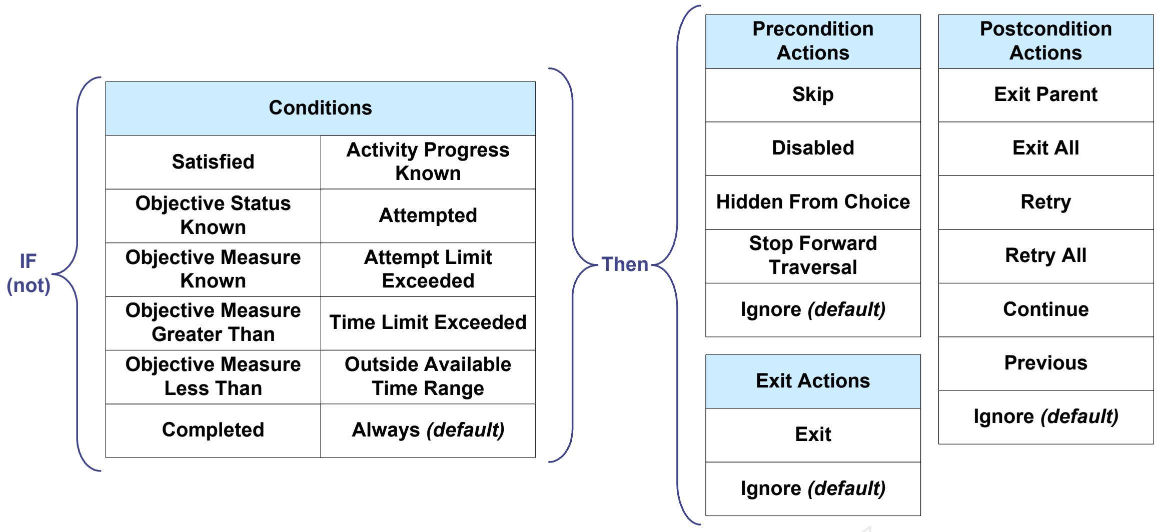


Figure 5.1.2a: Conditions and Actions/Behaviors

5.1.2.1 Sequencing Rule Combination

As mentioned above, sequencing rules can be combined to create sets of rules. These set of rules can be qualified to indicate which rules should be evaluated by an LMS. The IMS Simple Sequencing specification defines two types of qualifiers (condition combination). The default, if no qualifier is explicitly defined for the set of sequencing rules, is to evaluate all of the rules found in the set of sequencing rules. The table below explains the specifics of the Condition Combination information model element.

No.	Name	Description	Value Space	Default Value
1	*Condition Combination*	How rule conditions are combined in evaluating the rule. • *All* – The rule condition evaluates to True if and only if all of the individual rule conditions evaluate to True (logical *and*). • *Any* – The rule condition evaluations to True if any of the individual rule conditions evaluates to True (logical *or*).	Vocabulary	All

Table 5.1.2.1a: Condition Combination Description

5.1.2.2 Conditions

The Condition element contains a single condition that is evaluated in the context of the activity for which the sequencing rule set is defined. Individual Condition elements can be combined with other Condition elements to form a set of conditions. The content developer then has the ability to use the Condition Combination element to describe to the LMS how to evaluate the set of conditions. The Condition element consist of a set of restricted vocabulary items that are based on the IMS Simple Sequencing Tracking Status Model:

Condition	Description
Satisfied	The Condition evaluates to True if the Objective Progress value of the *Objective Satisfied Status* for the objective associated with the activity (indicated by the *Rule Condition Referenced Objective*) is True.
Objective Status Known	The Condition evaluates to True if the Objective Progress value of *Objective Progress Status* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective*) is True.
Objective Measure Known	The Condition evaluates to True if the Objective Progress value of *Objective Measure Status* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective)* is True.
Objective Measure Greater Than	The Condition evaluates to True if the Objective Progress value of *Objective Normalized Measure* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective)* is greater than the *Rule Condition Measure Threshold*.
Objective Measure Less Than	The Condition evaluates to True if the Objective Progress value of *Objective Normalized Measure* for the objective associated with the activity (indicated by *Rule Condition Referenced Objective)* is less than the *Rule Condition Measure Threshold*.
Completed	The Condition evaluates to True if the Attempt Progress value of *Activity Attempt Completion Status* for the activity is True.
Activity Progress Known	The Condition evaluates to True if the Attempt Progress value of *Activity Attempt Progress Status* for the activity is True.
Attempted	The Condition evaluates to True if the Activity Progress value of *Activity Attempt Count* for the activity is positive (i.e., the activity has been attempted).
Attempt Limit Exceeded	The Condition evaluates to True if the Activity Progress value of *Activity Attempt Count* for the activity is equal to or greater than the Limit Condition value of *Limit Condition Attempt Limit* for the activity.
Time Limit Exceeded	The Condition evaluates to True if any of the Activity or Attempt Progress duration values for the activity (*Activity Absolute Duration, Activity Experienced Duration, Activity Attempt Absolute Duration, Activity Attempt Experienced Duration*) exceed the corresponding duration Limit Condition values for the activity (*Activity Absolute Duration Limit, Activity Experienced Duration Limit, Activity Attempt Absolute Duration Limit, Activity Attempt Experienced Duration Limit*).
Outside Available Time Range	The Condition evaluates to True if the current time is before or after the corresponding time Limit Conditions for the activity. (*Begin Time Limit, End Time Limit*).
Always	The condition always evaluates to True. This is the default rule if no conditions are defined by the content author.

Table 5.1.2.2a: Descriptions of Rule Conditions

5.1.2.3 Rule Condition Referenced Objective

The Rule Condition Referenced Objective is used to identify which objective, out of the set of objectives defined for the activity, should be used during the evaluation of the rule condition that requires an objective information. These rule conditions are:

- Satisfied
- Objective Status Known
- Objective Measure Known
- Objective Measure Greater Than

- Objective Measure Less Than

If the activity has "1 and only 1" objective defined, then the Rule Condition Referenced Objective is considered optional and does not have to be specified. The default case is to use the "1 and only 1" objective for all rule condition evaluations that require the referenced objective. If the content developer does not define a unique identifier for the objective, then this attribute does not need to be used.

No.	Name	Description	Value Space	Default Value
2.2	*Rule Condition Referenced Objective*	The identifier of an objective associated with the activity used during the evaluation of the condition.	GUID	None

Table 5.1.2.3a: Rule Condition Referenced Objective

5.1.2.4 Rule Condition Measure Threshold

A rule condition can be defined that indicates the Measure Threshold for the activity. This threshold is used during the evaluation of the following rule conditions:

- Objective Measure Greater Than: [objective measure] > [measure threshold]
- Objective Measure Less Than : [objective measure] < [measure threshold]

This element is defined as a real number within the range of –1.0 to 1.0. The content developer should be aware that the comparisons performed with this threshold are greater than (>) and less than (>). There is no rule conditions defined that allow the greater than or equal to (>=) or the less than or equal to (<=) comparison operators.

<u>ADL Note.</u> With the introduction of this element, the MasteryScore element defined in the SCORM Version 1.2 is being deprecated. The MasteryScore element shall not be used in SCORM Version 1.3 manifests. Since this element replaces the MasteryScore, the current SCORM Version 1.2 mapping the element still persists. The Rule Condition Measure Threshold value should be used to initialize the cmi.student_data.mastery_score. If no Rule Condition Measure Threshold is defined the cmi.student_data.mastery_score should be considered 0 (default value for the Rule Condition Measure Threshold).

No.	Name	Description	Value Space	Default Value
2.3	*Rule Condition Measure Threshold*	The value used as a threshold during measure-based condition evaluations.	Real [-1..1] Precision of at least 4 significant decimal digits	0

Table 5.1.2.4a: Rule Condition Measure Threshold

5.1.2.5 Rule Condition Operator

The Rule Condition Operator element denotes the unary logical operators that can be applied to an individual condition. The IMS Simple Sequencing specification currently defines two valid values for this element:

- NO-OP: use rule condition as is (default value)
- Not: the rule condition should be negated prior to rule evaluation

No.	Name	Description	Value Space	Default Value
2.4	*Rule Condition Operator*	The unary logical operator to be applied to the condition. • *Not* – The corresponding condition is negated in rule evaluation. • *NO-OP* – The corresponding condition is used as is in rule evaluation.	Vocabulary	NO-OP

Table 5.1.2.5a: Rule Condition Operator

Note that defining a condition with the "not" operator in conjunction with the Always rule condition value means that the condition will unconditionally evaluate to false.

5.1.2.6 Actions

The Action element represents the intended action or behavior that the LMS is responsible for during the Sequencing Process. The set of actions are categorized in three unique evaluation order situations:

- Precondition Actions: apply when traversing the activity tree to select an activity for delivery.
- Postcondition Actions: apply when an activity attempt terminates.
- Exit Actions: apply after a descendant activity's attempt terminates.

No.	Name	Description	Value Space	Default Value
Precondition Actions				
3	*Rule Action*	The desired sequencing behavior if the rule evaluates to True. • *Skip* – The activity is not considered a candidate for delivery during a *Flow* sequencing request. • *Disabled* – The activity may not be the target of any sequencing or delivery request. • *Hidden from Choice* – The activity may not be the target of a "choice" sequencing request. • *Stop Forward Traversal* – Stop walking the activity tree, in a forward direction, at the activity while processing a sequencing	Vocabulary	Ignore

		request. • *Ignore* – No action.		
Postcondition Actions				
3	*Rule Action*	The desired sequencing behavior if the rule evaluates to True. • *Exit Parent* – Apply an *Exit Parent* exit request. • *Exit All* – Apply an *Exit All* exit request and return an *Exit* sequencing request. • *Retry* – Return a *Retry* sequencing request. • *Retry All* – Apply an *Exit All* exit request and return a *Start* sequencing request. • *Continue* – Return a *Continue* sequencing request. • *Previous* – Return a *Previous* sequencing request. • *Ignore* – No action.	Vocabulary	Ignore
Exit Actions				
3	*Rule Action*	The desired sequencing behavior if the rule evaluates to True. • *Exit* – Unconditionally terminate the activity. • *Ignore* – No action.	Vocabulary	Ignore

Table 5.1.2.6a: Description of Actions

5.1.3. Limit Conditions

A content developer can define limit conditions that control certain circumstances regarding if and when an activity can be delivered. Limit conditions can be associated with activities and are conditional based on the Tracking Model. When a limit condition is met or exceeded, the activity becomes unavailable for delivery.

Limit Conditions always override sequencing rules. For example, if a sequencing rule is defined for activity that specifies that the activity be retried, but a maximum attempts limit condition is also defined, and the maximum number of attempts has been met or exceeded, the activity is not retried because the limit condition takes precedence over the sequencing rule.

Similarly, limit conditions always supercede sequencing requests. For example, if a “Choose” sequencing request targets an activity that has a specified available time range, and the current clock time is outside the range, then the activity that was chosen is not available for delivery.

The following table defines the set of limit conditions. The Limit Conditions are defined as a pair values: a Control variable and the Limit. The control variable is used to indicate whether or not a particular Limit Condition has been defined by the content author. The Limit contains the specified limit condition. Certain sequencing processes are defined to act on whether or not limit conditions are defined (Control).

5.1.3.1 Attempts

There may be use cases where a content author wants to limit the number of attempts that a learner is permitted on a given learning activity. Sequencing rules can be built to limit the number of attempt for a specific learning activity. The *Limit Condition Attempt Limit* element contains a non-negative integer value that specifies the maximum number of attempts that may be taken on the associated learning activity for which the Limit Condition is defined. If the content author does not define a *Limit Condition Attempt Limit* value, then the is no constrain on the number of attempts that may be taken on the activity.

No.	Name	Description	Value Space	Default Value
1	*Limit Condition Attempt Control*	Indicates that a limit condition on the number of attempts for the activity has been established for the activity. If the value is False, there is no constraint on how many times the activity may be attempted.	Boolean	False
2	*Limit Condition Attempt Limit*	The maximum number of attempts for the activity. A zero value indicates the activity may not be accessed. The value is unreliable unless *Limit Condition Attempt Control* is True.	Non Negative Integer	0

Table 5.1.3.1a: Limit Condition (Attempt) Excerpt from IMS Simple Sequencing

According to the IMS Simple Sequencing Limit Condition table (Table 4.1.3.1a), if a *Limit Condition Attempt Limit* is defined then the *Limit Condition Attempt Control* is set to "True". If the *Limit Condition Attempt Limit* is not defined then the *Limit Condition Attempt Control* is set to the default of "False".

5.1.3.2 Activity Attempt Absolute Duration

There may be scenarios where a content author wants to limit the duration of time that can be spent in an attempt of a learning activity. Sequencing rules can be built to specify this duration limit. The *Activity Attempt Absolute Duration* element contains a value that specifies the maximum time duration that a learner is permitted to spend on a single attempt on the learning activity for which this value is specified. If the content author does not define an *Activity Attempt Absolute Duration* for a learning activity, then there is no constraint on the how long the learner can spend on the activity.

No.	Name	Description	Value	Default

			Space	Value
3	*Limit Condition Activity Attempt Absolute Duration Control*	Indicates that a limit condition on the maximum time duration that a learner is permitted to spend on any single attempt on the activity has been established for the activity. If the value is False, this limit does not constrain how long the learner may spend on the activity.	Boolean	False
4	*Limit Condition Activity Attempt Absolute Duration Limit*	The maximum time duration that a learner is permitted to spend on any single attempt on the activity. This limit applies to the time the activity is *active* – from the time the activity begins until the time the activity ends, including any time the activity was *suspended*. A zero value indicates the activity may not be accessed. The value is unreliable unless *Limit Condition Activity Attempt Absolute Duration Control* is True.	Duration – Accuracy 0.1 second	0.0

Table 5.1.3.2a: Activity Attempt Absolute Duration Limit

According to the IMS Simple Sequencing Limit Condition table (Table 4.1.3.2a), if a *Activity Attempt Absolute Duration Limit* is defined then the *Activity Attempt Absolute Duration Control* is set to “True”. If the *Limit Condition Attempt Limit* is not defined then the *Limit Condition Attempt Control* is set to the default of “False”.

5.1.3.3 Activity Attempt Experienced Duration

The IMS Simple Sequencing Specification defines two sets of durations, absolute and experienced. The experienced duration is the time that the learner is actually interacting with the activity. The duration does not include any time that the activity is suspended. The absolute duration (*Activity Attempt Absolute Duration Limit*), is the absolute duration from the time the attempt on the activity begins until the time the attempt on the activity ends, including any time the activity was suspended. If the content author does not define a *Limit Condition Activity Attempt Experienced Duration Limit* value, then there is no constraint on how long the learner may spend experiencing the attempt on the activity.

No.	Name	Description	Value Space	Default Value
5	*Limit Condition Activity Attempt Experienced Duration Control*	Indicates that a limit condition on the maximum time duration that a learner is permitted to spend on any single attempt on the activity has been established for the activity. If the value is False, this limit does not constrain how long the learner may spend on the activity.	Boolean	False
6	*Limit Condition Activity Attempt Experienced Duration Limit*	The maximum time duration that a learner is permitted to spend experiencing a single attempt on the activity. The limit applies to only the time the learner is actually interacting with the activity and does not apply when the activity is suspended	Duration – Accuracy 0.1 second	0.0

		(i.e., when the activity is not being experienced or is inactive). A zero value indicates the activity may not be accessed. The value is unreliable unless *Limit Condition Activity Attempt Experienced Duration Control* is True.		

Table 5.1.3.3a: Activity Attempt Absolute Duration Limit

5.1.3.4 Activity Absolute Duration

The previous two Limit Conditions (*Activity Attempt Absolute Duration Limit* and *Activity Attempt Experienced Duration Limit*) were focused on individual attempts on the activities. The IMS Simple Sequencing Specification also enables the tracking and limiting of time spent on the activity as a whole. The Activity Absolute Duration defines a limit condition on the maximum duration that a learner is permitted to spend on the activity. This duration is calculated by adding all of the absolute activity attempt durations (*Activity Attempt Absolute Duration*) for the given learner. Content authors can build sequencing rules that focus on the duration that a learner is permitted to spend on the activity as a whole.

No.	Name	Description	Value Space	Default Value
7	*Limit Condition Activity Absolute Duration Control*	Indicates that a limit condition on the maximum duration that a learner is permitted to spend on the activity (which includes all attempts) has been established for the activity. If the value is False, this limit does not constrain how long the learner may spend on the activity.	Boolean	False
8	*Limit Condition Activity Absolute Duration Limit*	The maximum time duration that a learner is permitted to spend on all attempts at the activity. A zero value indicates the activity may not be accessed. The value is unreliable unless *Limit Condition Activity Absolute Duration Control* is True.	Duration – Accuracy 0.1 second	0.0

Table 5.1.3.4a: Activity Attempt Absolute Duration Limit

5.1.3.5 Activity Experienced Duration

The Activity Experienced Duration defines the limit condition that a content author can define which limits the maximum duration that a learner is permitted to spend on the activity. As with Activity Attempt Experienced Duration Limit, this limit condition only applies to the time that the learner has spent actually experiencing the activity. This duration is calculated by adding all of the experienced activity attempt durations (*Activity Attempt Experienced Duration*) for the given learner.

No.	Name	Description	Value	Default

			Space	Value
9	*Limit Condition Activity Experienced Duration Control*	Indicates that a limit condition on the maximum duration that a learner is permitted to spend on the activity (which includes all attempts) has been established for the activity. If the value is False, this limit does not constrain how long the learner may spend on the activity.	Boolean	False
10	*Limit Condition Activity Experienced Duration Limit*	The maximum time duration that a learner is permitted to spend on all attempts of the activity. The limit applies to only the time the learner is actually experiencing the activity and does not apply when the activity is suspended (i.e., when the activity is not being experienced or is inactive). A zero value indicates the activity may not be accessed. The value is unreliable unless *Limit Condition Activity Experienced Duration Control* is True.	Duration – Accuracy 0.1 second	0.0

Table 5.1.3.5a: Activity Attempt Absolute Duration Limit

5.1.3.6 Begin Time Limit

There may be scenarios where a content author wants to limit when an activity is available to be delivered to the learner. The IMS Simple Sequencing Specification enables the content author to define such a rule. The *Begin Time Limit* defines the date and time before which the activity is not available to be delivered. If there is no *Begin Time Limit* defined for the activity, then there is no constraint on when the learner may access the activity.

No.	Name	Description	Value Space	Default Value
11	*Limit Condition Begin Time Limit Control*	Indicates that a limit condition on the availability of the activity has been established for the activity. If the value is False, there is no constraint on when the learner may access the activity.	Boolean	False
12	*Limit Condition Begin Time Limit*	The date and time before which the activity is not available. The value is unreliable unless *Limit Condition Begin Time Limit Control* is True.	Time Point – Accuracy 0.1 second	October, 15 1582 00:00:00.0

Table 5.1.3.6a: Activity Attempt Absolute Duration Limit

5.1.3.7 End Time Limit

There may be scenarios where a content author wants to limit when an activity becomes unavailable for delivery to the learner. The IMS Simple Sequencing Specification enables the content author to define such a rule. The *End Time Limit* defines the date and time after which the activity is not available to be delivered. If there is no *End Time*

Limit defined for the activity then there is no constraint on when the learner may access the activity

No.	Name	Description	Value Space	Default Value
13	*Limit Condition End Time Limit Control*	Indicates that a limit condition on the availability of the activity has been established for the activity. If the value is False, there is no constraint on when the learner may access the activity.	Boolean	False
14	*Limit Condition End Time Limit*	The date and time after which the activity is not available. The value is unreliable unless *Limit Condition End Time Limit Control* is True.	Timepoint – Accuracy 0.1 second	October, 15 1582 00:00:00.0

Table 5.1.3.7a: Activity Attempt Absolute Duration Limit

5.1.4. Auxiliary Resources

An activity may have auxiliary resources associated with it that provide the learner with additional services or resources. The IMS Simple Sequencing Specification does not define any semantics or meanings for these auxiliary resources. The IMS Specification does not define which resource may be made available, or how the resources are used. The only thing that the IMS Specification provides is a means for the auxiliary resources to be associated with an activity.

The IMS Specification provides for elements to associate the auxiliary resources to a given activity. The *Resource ID* (Value Space – GUID) and *Purpose* (Value Space – Open Vocabulary). The *Resource ID* is the identifier of the auxiliary resource. The *Purpose* indicates the purpose of the identified auxiliary resource.

The set of requirements defined by the IMS Specification:

1. When an activity is delivered to the learner, the auxiliary resources are also made available to the learner.
2. Auxiliary resources are optional and do not have to be defined for each activity.
3. If auxiliary resources are provided for an activity, then they are required to be uniquely identified and a purpose shall be provided.

The IMS Specification states that the *Purpose* is an open, unspecified vocabulary. The specification does not attempt to provide a set of predefined auxiliary resources and associated tokens (purpose). See (*To Be Supplied*) for a set of ADL defined auxiliary resources and defined behaviors.

5.1.5. Rollup Rule Descriptions

The IMS Simple Sequencing Specification defines the details of individual rule based rollup behavior for an activity. Rollup is defined as the process of evaluating the Objective and Attempt Progress data for a set of child activities to determine the Objective and Attempt Progress data for the parent. The Rollup Rules define a set of rollup control rules for describing this processes.

A set of zero or more rollup rules can be defined for an activity. Rollup rules are evaluated during the Rollup Process (see Rollup Behavior Section 7.1.5). The evaluation of rollup rules may alter the default sequencing path or may affect the availability of the activity and/or it's sub-activities for delivery.

Rollup rules consist of a set of child activity conditions and a corresponding action or behavior that is performed if the set of conditions evaluates to true (*if [child_activity_set condition_set] then [action/behavior]*). Rollup rules are optional and should only be defined when needed by the content author. Each rule has a defined default value. The default data is used if the data is not instantiated for a given rule.

5.1.5.1 Child Activity Set

The *Rollup Child Activity Set* defines the set of children that are used during the processing of Rollup Rules. There are several mechanisms defined by the IMS Simple Sequencing Specification to enable the content author to define the child activities to use for evaluation. The Child Activity Set is made up of a set of fixed vocabulary describing the set of child activities that participate in the evaluation process.

- All (*default value*). If the combination of all of the children of the parent activity has a rollup condition that evaluates to true, then perform the specified action
- Any. If any of the children of the parent activity has a rollup condition that evaluates to true, then perform the specified action
- None. If none of the children of the parent activity has a rollup condition that evaluates to true, then perform the specified action
- At Least Count. If at least the number of children indicated by the Rollup Minimum Count of the parent activity has a rollup condition that evaluates to true, then perform the specified action.
- At Least Percent. If at least the number of children indicated by the Rollup Minimum Count of the parent activity has a rollup condition that evaluates to true, then perform the specified action.

When the At Least Count vocabulary is used in describing the rollup rule condition, the value of the *Rollup Minimum Count* is required. The value is an integer value indicating the minimum number of children activities the must be evaluated in the rollup up process. The default value of the *Rollup Minimum Count* is 0, so if the value is left unspecified no children will be considered during the rollup evaluation.

When the At Least Percent vocabulary is used in describing the rollup rule condition, the value of the *Rollup Minimum Percent* is required. The value is a Real number between 0 and 1. The value specifies the minimum percentage of child activities that must be evaluated during the rollup process.

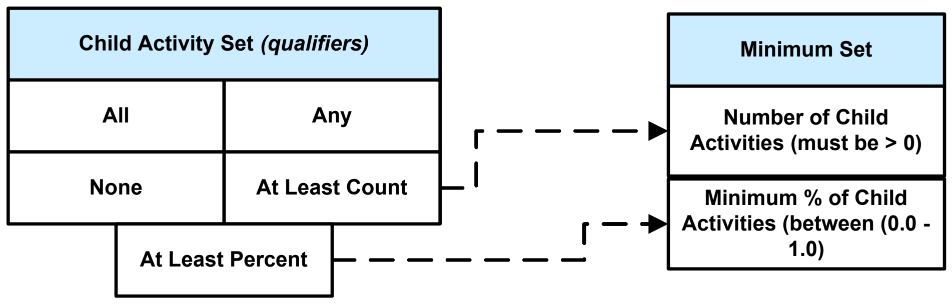

Child Activity Set *(qualifiers)*
All
Any
None
At Least Count
At Least Percent
Minimum Set
Number of Child Activities (must be > 0)
Minimum % of Child Activities (between (0.0 - 1.0)

Figure 5.1.5.1a: Minimum Set Relationship

There is flexibility built into the IMS Simple Sequencing specification to define various Rollup Rules for many different situations. The figure below show two simple cases on how to build Rollup Rules using the various elements described:

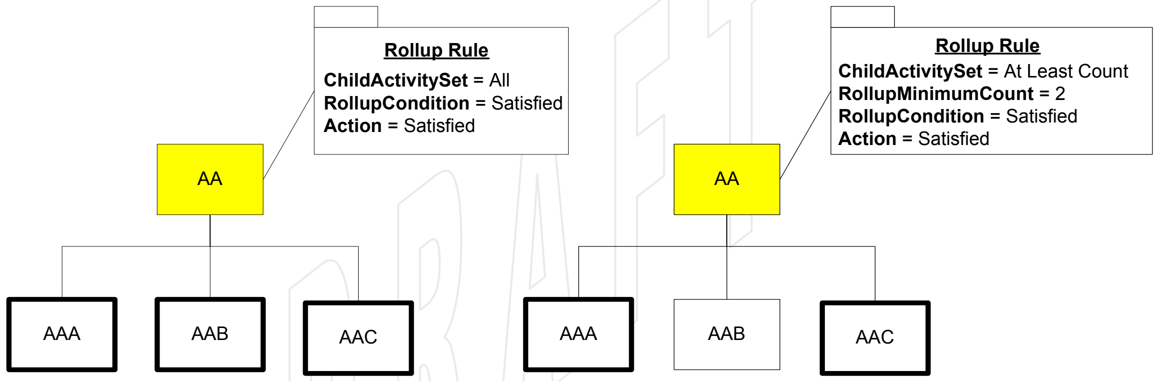


Figure 5.1.5.1b: Rollup Rule Illustration

In the first scenario, the rule defines a condition that states if “All” of AA’s children are “Satisfied” then AA should be considered “Satisfied”. The second scenario states that if at least 2 of Activity AA’s children are “Satisfied” then AA should be considered “Satisfied”. Based on these Rollup Rules and applied actions, a Sequencing Rule could then apply to activity AA that sates if AA is “Satisfied” then “Skip”.

5.1.5.2 Rollup Condition Combinations

Rollup rules can be combined to create sets of rules. These sets of rules can be qualified to indicate which rules should be evaluated by an LMS. The IMS Simple Sequencing Specification defines two types of qualifiers (*Condition Combination*).

- All. The rule condition evaluates to True if and only if all of the individual rollup conditions evaluate to True. Acts as a logical *and*.
- Any (*default value*). The rule condition evaluates to True if any of the individual rollup conditions evaluate to True. Acts as a logical *or*.

5.1.5.3 Rollup Conditions

The IMS Simple Sequencing Specification allows for the content author to define a set of Rollup Conditions. The Rollup Conditions element is a collection of individual Rollup Condition statements that contains a single condition that is evaluated in the context of the activity for which the rollup rule set is defined. The content developer then has the ability to use the *Condition Combination* element to describe to the LMS how to evaluate the set of conditions. The Condition element consist of a set of restricted vocabulary items that are based on the IMS Simple Sequencing Tracking Status Model:

Condition	**Description**
Satisfied	The Condition evaluates to True if the Objective value of *Objective Satisfied Status* for the rolled up objective associated with the child activity is True.
Objective Status Known	The Condition evaluates to True if the Objective value of *Objective Data Status* for the rolled up objective associated with the child activity is True.
Objective Measure Known	The Condition evaluates to True if the Objective value of *Objective Measure Status* for the rolled up objective associated with the activity is True.
Completed	The Condition evaluates to True if the Attempt Progress value of *Activity Attempt Completion Status* for the child activity is True.
Activity Progress Known	The Condition evaluates to True if the Activity Attempt Progress value of *Activity Attempt Progress Status* for the child activity is True.
Attempted	The Condition evaluates to True if the Activity Progress value of *Activity Attempt Count* for the child activity is positive (i.e., the child activity has been attempted).
Attempt Limit Exceeded	The Condition evaluates to True if the Activity Progress value of *Activity Attempt Count* for the child activity is equal to or greater than the Limit Condition value of *Limit Condition Attempt Limit* for the child activity.
Time Limit Exceeded	The Condition evaluates to True if any of the Activity Progress duration values for the child activity (*Activity Absolute Duration, Activity Experienced Duration, Activity Attempt Absolute Duration, Activity Attempt Experienced Duration*) exceed the corresponding duration Limit Condition values for the child activity (*Activity Absolute Duration Limit, Activity Experienced Duration Limit, Activity Attempt Absolute Duration Limit, Activity Attempt Experienced Duration Limit*).
Outside Available Time Range	The Condition evaluates to True if the current time is before or after the corresponding time Limit Conditions for the child activity (*Begin Time Limit, End Time Limit*).
Always	The condition always evaluates to False. This is the default rule if no conditions are defined by the content author.

Table 5.1.5.3a: Rollup Conditions

5.1.5.4 Rollup Condition Operator

The Rollup Condition Operator element denotes the unary logical operators that can be applied to an individual rollup conditions. The IMS Simple Sequencing specification currently defines two valid values for this element:

- NO-OP (*default value*): use rule condition as is
- Not: the rule condition should be negated prior to rule evaluation

5.1.5.5 Rollup Actions

The Rollup Action describes the desired action that should be applied to the activity that contains the Rollup Rule (if the rule or sets of rules evaluates to True). The rolled up behavior is applied to the Objective Progress Status or Activity Attempt Progress of the associated activity:

Rollup Action	**Description of Action**
Satisfied (*default value*)	Set the associated parent activity's objective (*Objective Satisfied Status*) to True. This action also sets the associated parent activity's objective status (*Objective Progress Status*) to True.
Not Satisfied	Set the associated parent activity's objective (*Objective Satisfied Status*) to False. This action also sets the associated parent activity's objective status (*Objective Progress Status*) to True.
Completed	Set the associated parent activity's completion status (*Activity Attempt Completion Status*) to True. This action also sets the associated parent activity's progress status (*Activity Attempt Progress Status*) to True.
Incomplete	Set the associated parent activity's completion status (*Activity Attempt Completion Status*) to False. This action also sets the associated parent activity's progress status (*Activity Attempt Progress Status*) to True.

Table 5.1.5.5a: Rollup Actions

5.1.5.6 Rollup Rule Example

The following figure (Figure 4.1.5.6a) illustrates several examples of the defining rollup up rules for learning activities.

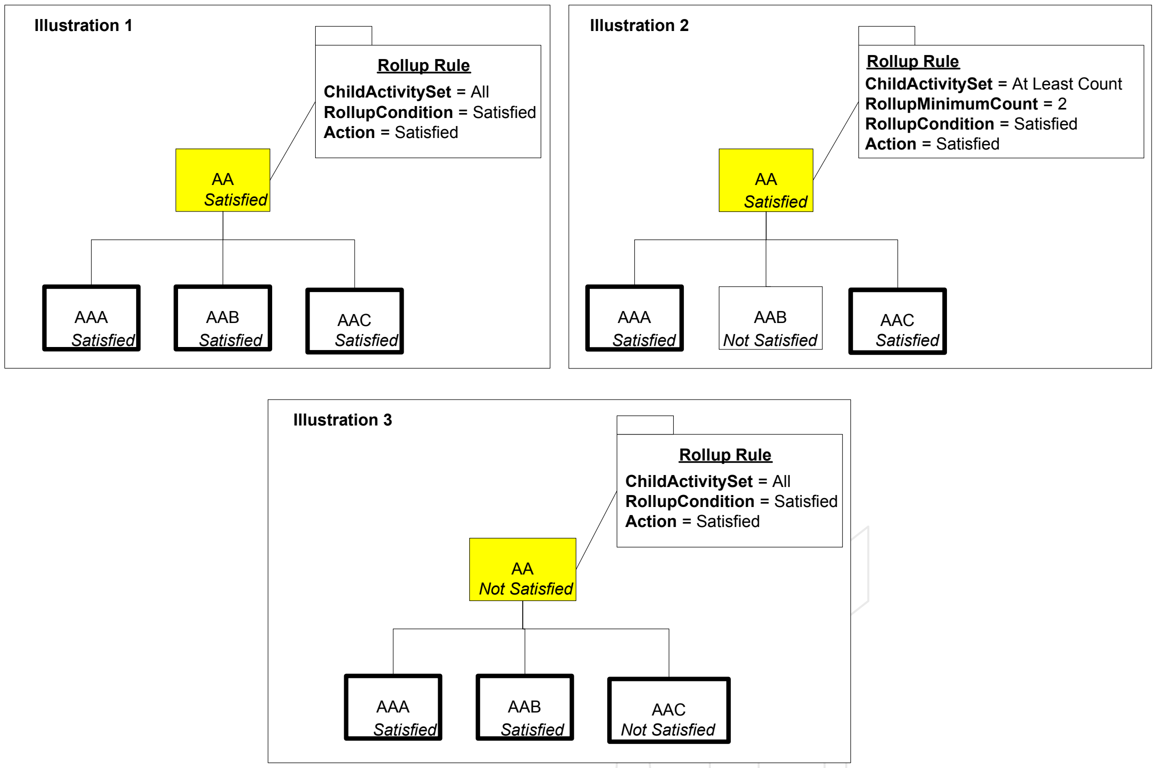


Figure 5.1.5.6a: Rollup Rule Condition Illustration

Illustration 1 depicts a rollup rule that states all of the parent's (AA) children (AAA, AAB and AAC) activities have to be considered "satisfied" in order to consider AA satisfied. Illustration 2 depicts a rollup rule that states only 2 of the children activities must have a status of "satisfied", in order for its parent (AA) to be considered satisfied. Illustration 3 depicts a rollup rule that is not satisfied. The Rollup Rule states that all of the children activities (AAA, AAB and AAC) must be satisfied in order for the parent activity (AA) to be satisfied. Child activity AAC status is Not Satisfied which cause the rollup rule to deem activity AA as not satisfied.

5.1.6. Controlling Rollup Process

The IMS Simple Sequencing Specification enables a content author to delineate what information is included for rollup. The rollup controls include descriptions of the types of rollup behaviors specified for an activity. There are currently three types of tracking status model information that are permitted to be involved in the rollup process:

- Objective Satisfaction (*Objective Satisfied Status*)
- Objective Measure (*Objective Normalized Measure*)
- Activity Completion Status (*Activity Attempt Completed Status*)

The first two values only apply to rollup if and only if the associated objective indicates that it contributes to rollup (*Objective Contributes to Rollup* set to True).

No.	Name	Description	Value Space	Default

				Value
1	*Rollup Objective Satisfied*	Indicates that the Objective value of *Objective Satisfied Status* for the objective (which has the *Objective Description* attribute value of *Objective Contributes to Rollup* equal to *True*) associated with the activity is included (True or False) in the rollup for the parent activity	Boolean	True
2	*Rollup Objective Measure Weight*	A weighting factor applied to the Objective value of *Objective Normalized Measure* for the objective (which has the *Objective Description* attribute value of *Objective Contributes to Rollup* equal to *True*) associated with the activity used during rollup for the parent activity.	Real [0..1] Precision of at least 4 significant decimal digits	1.0
3	*Rollup Progress Completion*	Indicates that the Attempt Progress information value of *Activity Attempt Completed Status* for the activity is included (True or False) in the rollup for the parent activity.	Boolean	True

Table 5.1.6a: Rollup Controls

5.1.7. Objectives

With the introduction of the IMS Simple Sequencing specification into the SCORM, there is a mechanism now in place to associate learning objective(s) with an activity. Each learning activity may have an unlimited number of learning objectives defined. The Tracking Status Model defines a set of data to be used to record satisfaction (e.g. satisfied, not satisfied) and measure (e.g. score) for each objective defined for the attempt on the activity. The mechanism enabled by the IMS Simple Sequencing specification is to associate a learning activity with an ID or set of IDs. The meaning of the learning objective is not defined.

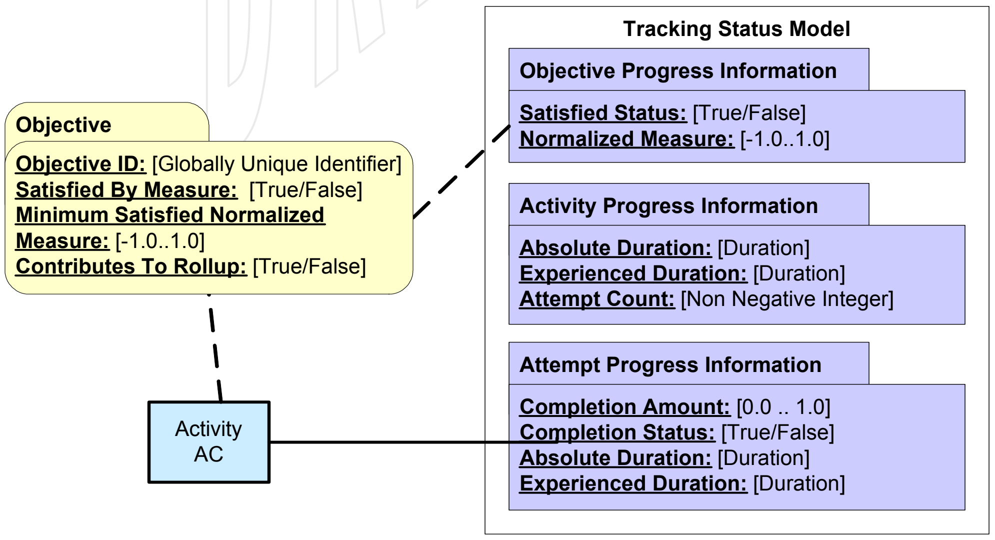

Tracking Status Model
Objective Progress Information
Satisfied Status: [True/False]
Normalized Measure: [-1.0..1.0]
Activity Progress Information
Absolute Duration: [Duration]
Experienced Duration: [Duration]
Attempt Count: [Non Negative Integer]
Attempt Progress Information
Completion Amount: [0.0 .. 1.0]
Completion Status: [True/False]
Absolute Duration: [Duration]
Experienced Duration: [Duration]
Objective
Objective ID: [Globally Unique Identifier]
Satisfied By Measure: [True/False]
Minimum Satisfied Normalized Measure: [-1.0..1.0]
Contributes To Rollup: [True/False]
Activity AC

Figure 5.1.7a: Objective Description and Objective Progress Information Relationship

Each objective contains a set of information that can be defined. The IMS Simple Sequencing Specification defines the following set of elements to describe each objective:

- Objective ID: A globally unique identifier for the objective associated with the activity.
- Objective Satisfied by Measure(*False – default value*): Whether or not the objective satisfaction is determined by a measure (e.g. score). This value indicates whether or not the objective's minimum satisfied normalized measure should be used in place of any other method to determine whether or not the activity is satisfied (Tracking Status Model – *Objective Satisfied Status*).
- Objective Minimum Satisfied Normalized Measure(*1.0 – default value*): The minimum satisfaction measure for the objective. This measure shall be normalized between –1.0 and 1.0. This value is unreliable unless the Objective Satisfied by Measure is set to True.
- Objective Contributes to Rollup (*False – default value*): Indicates whether or not the Tracking Status Model (*Objective Normalized Measure* and *Objective Satisfied Status*) is used during rollup evaluations.

If multiple objectives are defined for an activity, at least one primary objective for each activity shall be identified. The primary objective is the objective that is used during rollup.

5.1.7.1 Local Objective(s) vs. Global Objective(s)

Learning objectives are considered separate from learning activities. Learning objectives represent a set of locally and globally scoped data items, each with a satisfaction status and a satisfaction measure. Activities may have more than one associated local objective and may reference multiple globally shared objectives. Multiple activities may reference the same global objective, thus sharing the data values. This permits complex sequencing rules to be built to allow the status of one activity to affect the status of another activity.

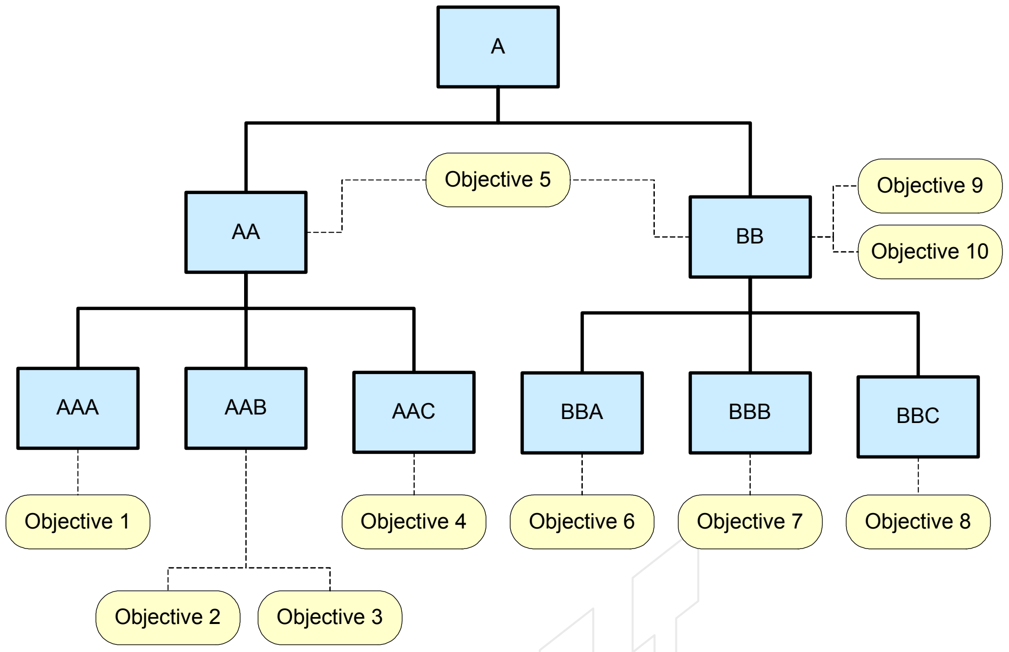


Figure 5.1.7.1a: Sharing Objectives

The resolution of local and global objective IDs is not specified. An objective may be shared within a single activity tree or may be shared globally across multiple activity trees. The lifetime of shared global objectives and the scope of sharing is not specified; it is determined by implementations. Activities may reference multiple objectives, thus providing a mechanism for activities to have sub-objectives.

5.1.7.2 Sharing Objectives - Objective Mapping

The *Objective Map* description defines a mapping of an activity's local objective information to and from a shared global objective. The objective mapping is the key enabler for sharing objective information between activities. There are several rules when applying objective maps:

- Each activity may have an unlimited number of objective maps.
- By default, no objective information is shared between activities. Each activity must define a set of *Objective Maps* to describe how local objective data is mapped to shared global objectives.
- The *Objective Map* data is evaluated whenever local objective information is processed, as described in the tracking model behaviors.
- For any given local objective, a 'read' mapping with at most one global objective may be defined.
- For any global objective, for any activity, a 'write' mapping with at most one local objective can be defined.

5.1.7.2.1 Reading/Writing Shared Objectives

There are scenarios, in the course of learning, where the content author may define rules that state that status of an objective for an activity would impact other activities. For example if a certain activity's objective has been met (satisfied) then a set of activities may not need to be presented to the learner (passed pre-test, no need to see material associated with certain objectives defined in the pre-test). There are other scenarios that can be created with this concept. In order to permit these types of scenarios the content author has to be able to define which data is shared with which activities. An *Objective Map* is the mechanism defined in the IMS Simple Sequencing Specification that enables this scenario.

There are two accessibility rules for this defined shared objective: "*reading from"* and "*writing to"*. Since, by default, each activity has at least one defined local objective, requirements have to be defined to indicate when to access the shared objective. A shared objective should be "*read from*" whenever objective status information is needed and the local objective information is undefined. The local objective information for each activity takes precedence over the shared objective. So if rules are being evaluated and the local objective information indicates that the objective is satisfied, then the global objective information should not be accessed. However, if the objective information for the local objective information is undefined and there is a objective map rule defined to read the satisfied status (True), then the shared global objective information should be read.

If the objective for the activity is defined to write to the global objective (*Write Objective Satisfied Status* – True), then the objective status, for the identified objective associated with the activity should be transferred to the global objective identified by the target objective id (*Target Objective ID*) upon exit of the activity.

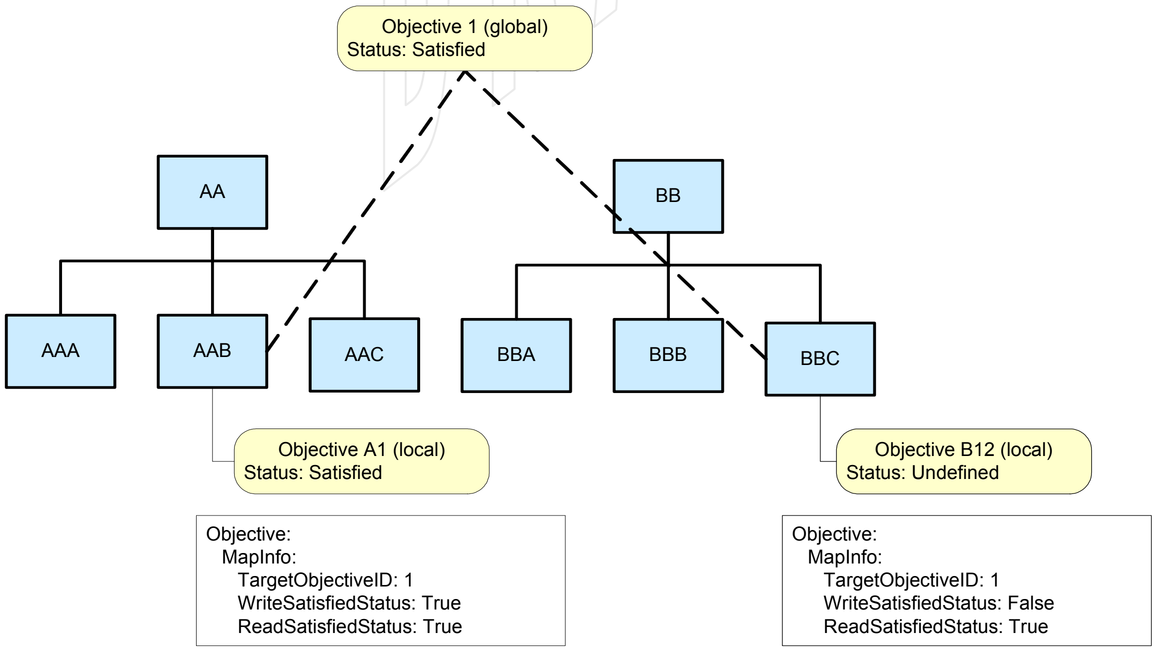


Figure 5.1.7.2.1a: Sharing Objectives

In the above diagram (Figure 5.1.7.2.1a), Objective 1 is identified as a global objective. The activities AAB and BBC both share the global objective (indicated by the TargetObjectiveID attribute). The Objective information for activity AAB indicates that the objective status should be read from, if needed, and written to the global objective (Objective 1). So when activity AAB terminates the objective status shall be transferred, by the LMS, to the global objective (Objective 1). If during the processing of sequencing rules to determine what to deliver next, the LMS may need to access activity BBCs objective status. Since the objective status for BBC is undefined the LMS must check the objective information defined for the activity. The objective information indicates that if the objective status is undefined, then the LMS shall read from the associated global objective. In this case, the LMS will read the global objective (Objective 1) that indicates that the objective is Satisfied. This then defines that activity BBC is also Satisfied and rules should be evaluated with this in mind.

No.	Name	Description	Value Space	Default Value
1	*Activity Objective ID*	The identifier of a local objective associated with the activity.	GUID	None Value is Required
2	*Target Objective ID*	The identifier of global shared objective targeted for the mapping.	GUID	None Value is Required
3	*Read Objective Satisfied Status*	Indicates that the local objective value of *Objective Satisfied Status*, for the identified objective associated with the activity (*Activity Objective ID*), should be retrieved (True or False) from the identified shared global objective (*Target Objective ID*), when the progress for the local objective is undefined (*Objective Progress Information* attribute value *Objective Progress Status* for the local objective is False). This operation does not change the Objective Information associated with the local objective.	Boolean	True
4	*Write Objective Satisfied Status*	Indicates that the local objective value of *Objective Satisfied Status*, for the identified objective associated with the activity (*Activity Objective ID*), should be transferred (True or False) to the identified global shared objective (*Target Objective ID*), upon exit of the activity.	Boolean	False
5	*Read Objective Normalized Measure*	Indicates that the local objective value of *Objective Normalized Measure*, for the identified objective associated with the activity (*Activity Objective ID*), should be retrieved (True or False) from the identified shared global objective (*Target Objective ID*), when the progress for the local object is undefined (*Objective Progress Information* attribute value *Objective Measure Status* for the local objective is False). This operation does not change the Objective Information associated with the local objective.	Boolean	True
6	*Write Objective*	Indicates that the local objective value of *Objective*	Boolean	False

	Normalized Measure	*Normalized Measure*, for the identified objective associated with the activity (*Activity Objective ID*), should be transferred (True or False) to the identified global shared objective (*Target Objective ID*), upon exit of the activity.		

Table 5.1.7.2.1a: Reading/Writing Global Objectives Controls

5.1.7.2.2 Lifetime of Global Objectives

The question arises on what is the life span of a global objective. At a minimum once a global objective (and its associated information) for an activity is created for a specific learner it shall live as along as the learner is associated with the given LMS. This allows for the global objective to be shared with other activities not in the context of the original activity (e.g. from course to course in an LMS).

5.1.8. Controlling Selection of Activities

Content authors have the ability to define sequencing rules that indicate when to select certain activities and limit the number of activities to be chosen. This enables a rule that can be written to state to a sequencer, pick 4 of the 6 activities on the first attempt of an activity.

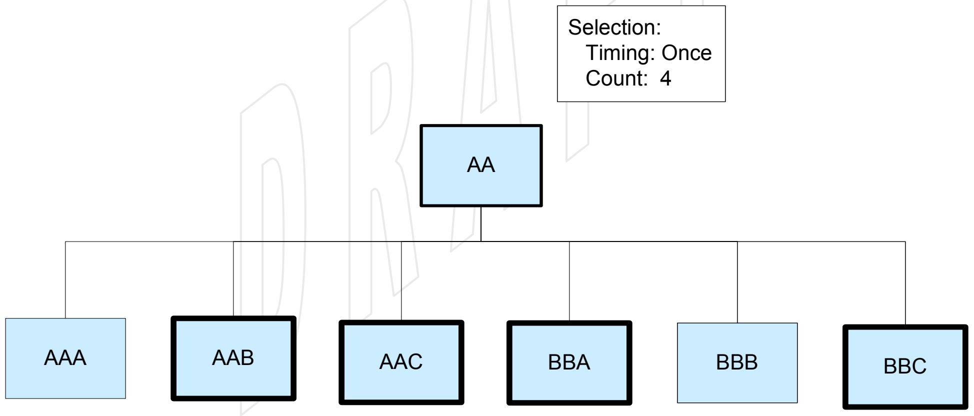


Illustration 1

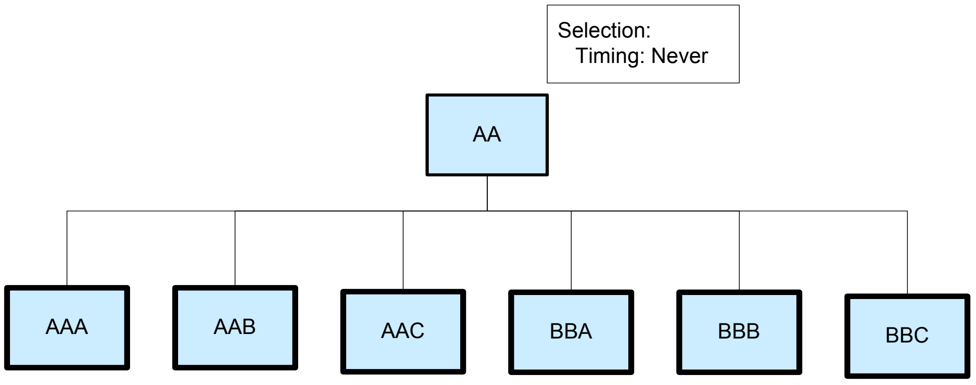


Illustration 2 (default case)

Figure 5.1.8a: Selection examples

Illustration 1 indicates that on the first attempt of activity AA, the sequencer shall pick 4 of the 6 children to deliver. This information must be persisted if the learner decides to suspend the attempt on activity AA. If the selection count is larger than the number of children then all activities should be selected as candidates to be delivered.

Illustration 2 indicates that the sequencer shall select all of the children as candidates to be delivered to the learner. This is the default behavior if no Selection rules are indicated.

Selection controls include descriptions of how the children of an activity should be selected during the sequencing process. Although the selection of child activities 'on each new attempt' is a desired feature, when to perform the operation is not well defined. The IMS Simple Sequencing specification does not specify the behavior of this attribute at this time. The *Selection Timing* of "On Each New Attempt" shall not be used at this time.

No.	Name	Description	Value Space	Default Value
1	*Selection Timing*	Indicates when selection should occur. • *Never* – Selection is never applied; all of the children of the activity are selected by default. • *Once* – Selection is applied before the first attempt on an activity. • *On Each New Attempt* – Selection is applied before each new attempt on an activity. The *On Each New Attempt* option and its associated behavior is not specified in this version of the Simple Sequencing Specification	Vocabulary	Never
2	*Selection Count Status*	Indicates the selection count data is (True or False) meaningful for the activity.	Boolean	False
3	*Selection Count*	Indicates the number of child activities that must be selected from the set of child activities associated with the activity. If *Selection Count* is larger than the number of child activities, all child activities are selected. This value is unreliable unless *Selection Count Status* is *True*. If *Selection Count Status* is *False* all child activities are selected.	Non Negative Integer	0

Table 5.1.8a: Selection Controls

5.1.9. Randomization Controls

Content authors have the ability to define sequencing rules that indicate whether or not a sequencer shall randomly select activities for delivery. This enables rules that govern whether and how child activities in a cluster of activities should be randomly sequenced when a new attempt is initiated for the parent activity of the cluster. The timing of the randomization can occur in the following:

- Never (*default value*): Randomization should never be applied to the children activities
- Once: Randomization shall be applied before the first attempt on the activity
- On Each New Attempt: Randomization shall be applied before each new attempt on the activity

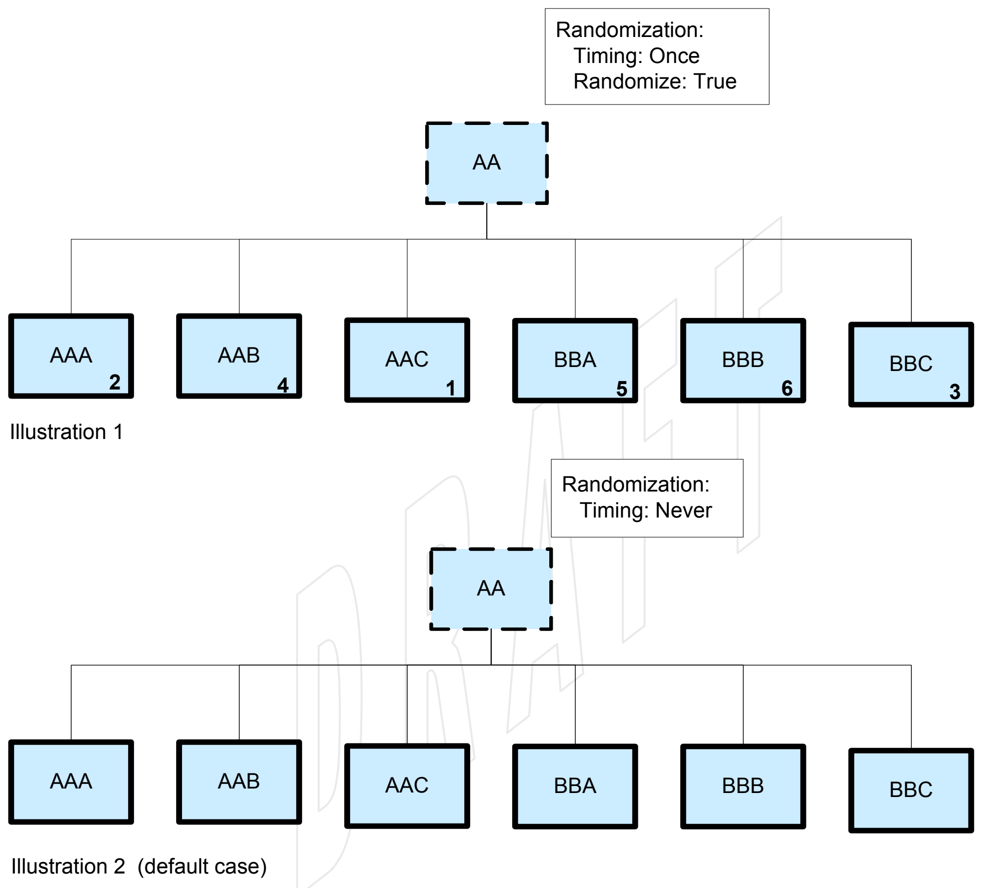


Figure 5.1.9a: Randomization Examples

The above figure illustrates two examples of randomization rules. Illustration 1 has a rule defined that indicates to the LMS that randomization shall be applied when activity AA is first attempted. The LMS is responsible for determining the random order (this is indicated by the numbers assigned to each activity). This information must be retained by the LMS in the case where the learner decides to suspend the attempt on activity AA. Illustration 2 (default case) has a rule defined that indicates to the LMS never to randomize the children activities.

No.	**Name**	**Description**	**Value Space**	**Default Value**
1	*Randomization Timing*	Indicates when the ordering of the children of the activity should occur. • *Never* – Randomization is never applied.	Vocabulary	Never

		• *Once* – Randomization is applied before the first attempt on the activity. • *On Each New Attempt* – Randomization is applied before each new attempt on the activity.		
2	*Randomize Children*	Indicates that the order of the child activities is randomized (True or False).	Boolean	False

Table 5.1.9a: Randomization Controls

5.1.10. Delivery Controls

Delivery controls describe actions and controls used when an activity is delivered. These controls indicate whether or not objective, activity and attempt progress data are recorded when the activity is delivered. Content authors can define these rules to limit which activity information is recorded for any activity.

No.	Name	Description	Value Space	Default Value
1	*Tracked*	Indicates that Objective and Attempt Progress information for the activity attempt should be recorded (True or False) and the data will contribute to the rollup for the parent activity. How the data is tracked and recorded is not specified.	Boolean	True
2	*Completion Set by Content*	Indicates that the Activity Attempt Completion Status information for the activity in the Tracking Model will be set by the content object (True or False). A False value indicates that default rules will be used to set progress data. For a True value, how, if or when the completion data is set is not specified.	Boolean	False
3	*Objective Set by Content*	Indicates that the Objective Status information for the associated objective that has the attribute value *Objective Contributes to Rollup* value of True with the Tracking Model will be set by the content object (True or False). A False value indicates that default rules will be used to set objective data. For a True value, how, if or when the objective data is set is not specified.	Boolean	False

Table 5.1.10a: Delivery Controls

The delivery controls shall be used by LMSs to aid in the tracking of data associated with activities. The controls indicate whether or not the LMS can expect the content (SCO), associated to the activity, to set particular tracking data (completion and objective).

5.1.10.1 Tracked

The Tracked element indicates that the Objective and Attempt Progress information for the activity should be recorded and tracked by the LMS. The default value, if not supplied by the content author is to track the information for the activity. If a content author does not want the information tracked by the LMS, the content author needs to specify this requirement by setting the Tracked element to “False”.

5.1.10.2 Completion Set by Content

The Completion Set by Content indicates that the content is responsible for indicating whether or not the activity is complete. The content (SCO) does not have direct access into the Objective and Attempt Progress information that is controlled by the LMS. The SCO can impact this information by setting various SCORM Run-Time Environment Data Model elements. The mapping of these specific elements to the Objective and Attempt Progress information is described in the Tracking Data Model section of this document. If this element is set to “True” and the SCO does not indicate the completion status then default values shall be used by the LMS in determining the Attempt Progress information.

5.1.10.3 Objective Set by Content

The Objective Set by Content indicates that the content is responsible for indicating whether or not the objective is satisfied. The content (SCO) does not have direct access into the Objective and Attempt Progress information that is controlled by the LMS. The SCO can impact this information by setting various SCORM Run-Time Environment Data Model elements. The mapping of these specific elements to the Objective and Attempt Progress information is described in the Tracking Data Model section of this document. If this element is set to “True” and the SCO does not indicate the objective status then default values shall be used by the LMS in determining the Attempt Progress information.

This page intentionally left blank.

SECTION 6
The Sequencing XML Binding

This page intentionally left blank.

XML Binding of the Sequencing Definition Model

The XML Binding is derived from the IMS Simple Sequencing XML Schema Definition (XSD). The XML Binding defines how the IMS Simple Sequencing Definition model is interpreted and bound into XML. The binding describes how the Definition Model items are represented in XML by indicating which items are XML elements versus XML attributes, the names of the elements/attributes, etc.

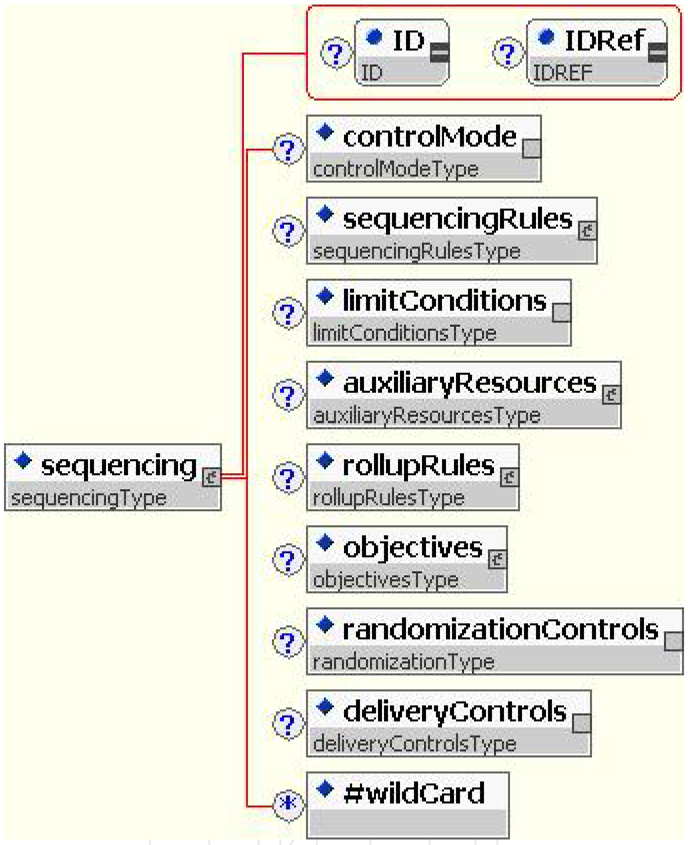
The following table shows symbols that relate directly to the "Multiplicity" specified for each element. This describes the number of times the element can occur within its parent element, or in the case of the top most element in the document, how many times the element occurs in the XML document.

Symbol	Meaning
(no symbol)	When no multiplicity symbol is present, this denotes that the element may exist **one and only one** time.
+	The plus sign denotes that the element may occur **one or more** times within its parent element.
?	The question mark denotes that the element may occur **zero or one** time within its parent element.
*	The asterisk denotes that the element may occur **zero to many** times within its parent element.

Table 6.1a: XML Binding Symbols

The Data Types for elements and attributes are defined in terms of the W3C Recommendation 02 May 2001, XML Schema: Datatypes (www.w3c.org). These data types are identified with the following prefix – xs:. For more information on the details of the data type please reference the W3C Recommendation for XML Schema.

6.1.1. <sequencing> Element



Description: The outermost root element for encapsulating the sequencing rules that apply to a learning activity.

Multiplicity: The <sequencing> element can appear 0 or 1 time as an extension to the <item> and <organization> elements of a content package manifest.

Attributes:

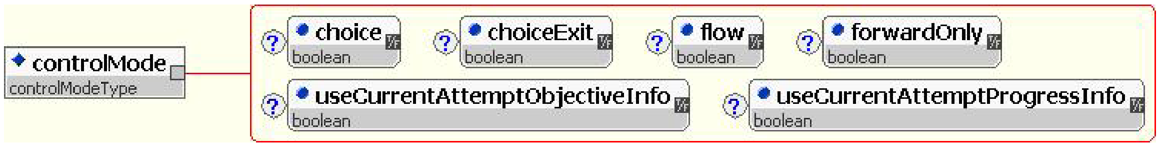
- ID (0 or 1, optional): An identifier that uniquely identifies the set of sequencing rules. Data Type: xs:ID
- IDRef (0 or 1, optional): A reference to a <resource> identifier (within the same package or a submanifest) that is used to resolve the ultimate location of the sequencing element. This allows a set of sequencing rules to be defined an re-used throughout the manifest. Data Type: xs:IDRef

Elements:

- controlMode
- sequencingRules
- limitConditions
- auxiliaryResources
- rollupRules
- objectives

- randomizationControls
- deliveryControls

6.1.1.1 <controlMode> Element



Description: Indicates the control mode specified by the content developer for the learning activity.

Multiplicity: The <controlMode> element shall occur 0 or 1 time within the parent element <sequencing>.

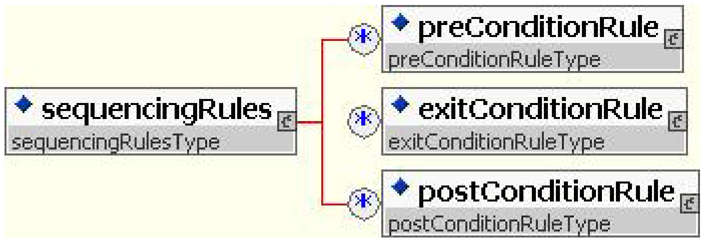
Attributes:

- choice (0 or 1, optional): Defines that the learning activity should be experienced in a *Choice* control mode. The attribute defaults to true if not present in the <controlMode> element. Data Type: xs:boolean.
- choiceExit (0 or 1, optional): Defines that the learning activity should be experienced in a *ChoiceExit* control mode. The attribute defaults to true if not present in the <controlMode> element. Data Type: xs:boolean.
- flow (0 or 1, optional): Defines that the learning activity should be experienced in a *Flow* control mode. The attribute defaults to false if not present in the <controlMode> element. Data Type: xs:boolean.
- forwardOnly (0 or 1, optional): Defines that the learning activity should be experienced in a *forwardOnly* control mode. The attributes defaults to false if not present in the <controlMode> element. Data Type: xs:boolean.
- useCurrentAttemptObjectiveInfo (0 or 1, optional): Indicates that the objective progress information for the children of the activity will be used in rule evaluations and rollup if that information was recorded during the current attempt of the activity. The attribute defaults to true if not present in the <controlMode> element. Data Type: xs:boolean.
- useCurrentAttemptProgressInfo (0 or 1, optional): Indicates that the attempt progress information for the children of the activity will be used in rule evaluations and rollup if that information was recorded during the current attempt of the activity. The attribute defaults to true if not present in the <controlMode> element. Data Type: xs:boolean.

Elements:

- None

6.1.1.2 <sequencingRules> Element



Description: This element contains a set of sequencing rules to apply to a learning activity.

Multiplicity: The <sequencingRules> element shall occur 0 or 1 time within the parent element <sequencing>.

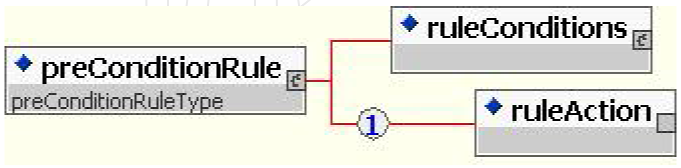
Attribute:

- None

Element:

- preConditionRule – Rules/Action that control sequencing decisions and delivery of a specific activity. Rules that include such actions are used to determine if the activity will be delivered
- exitConditionRule – Rules/Actions that terminate an activity. Rules that include such actions are applied when a descendent of an activity exits.
- PostConditionRule – Rules/Actions that control sequencing flow by issuing sequencing requests. Rules that include such actions are applied when an activity exits.

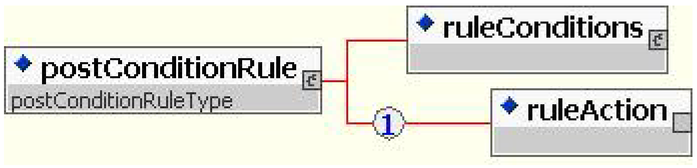
6.1.1.2.1 <preConditionRule> Element



Description: The <preConditionRule> element contains a single sequencing rule that belongs to the set of sequencing rules that apply to a learning activity.

Multiplicity: The <preConditionRule> element shall occur 0 or more times within the <sequencingRules> element.

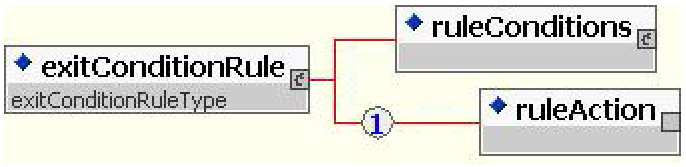
6.1.1.2.2 <postConditionRule> Element



Description: The <postConditionRule> element contains a single sequencing rule that belongs to the set of sequencing rules that apply to a learning activity.

Multiplicity: The <postConditionRule> element shall occur 0 or more time within the <sequencingRules> element.

6.1.1.2.3 <exitConditionRule> Element



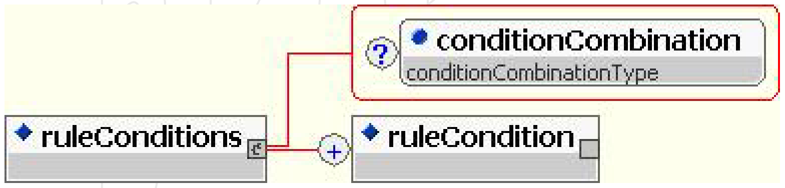
Description: The <exitConditionRule> element contains a single sequencing rule that belongs to the set of sequencing rules that apply to a learning activity.

Multiplicity: The <exitConditionRule> element shall occur 0 or more times within the <sequencingRules> element.

6.1.1.2.4 Rule Conditions

Each rule condition (preConditionRule, postConditionRule and exitConditionRule) is all the same content model. Each condition rule is made up of 1 and only 1 set of rule conditions and 1 and only 1 action. The next several sections define the content model for the rule conditions for each of the three conditions described above.

6.1.1.2.4.1 <ruleConditions> Element



Description: The <ruleConditions> element is an unordered collection of conditions for a sequencing rule. A rule may include multiple rule conditions.

Multiplicity: The <ruleConditions> element shall occur 1 and only 1 time within the <preConditionRule>, <postConditionRule> and <exitConditionRule> elements.

Attribute:

- conditionCombination (0 or 1, optional): How rule conditions are combined in evaluating the rule. Data Type: xs:token

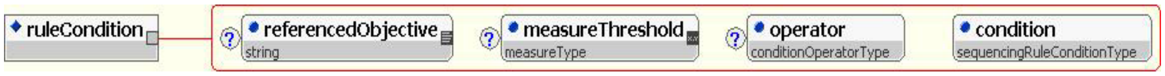
  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - all (*default value*)
  - any

Element:

- ruleCondition

6.1.1-2.4.1.1. <ruleCondition> Element



Description: The <ruleCondition> element contains a single sequencing rule that denotes actions that control sequencing decisions and delivery of a specific activity. Rules that include such actions are used to determine if the activity will be delivered.

Multiplicity: The <ruleCondition> element shall occur 1 or more times within the <ruleConditions> element.

Attribute:

- referencedObjective (0 or 1, optional): The identifier of an objective associated with the activity used during the evaluation of the condition. This identifier should be globally unique. Data Type: xs:string.
- measureThreshold (0 or 1, optional): The value used as a threshold during measure based condition evaluations. The attribute defaults to 0.0 if not present in the <ruleCondition> element. Data Type: xs:decimal [-1..1] (Precision of at least 4 significant decimal digits).
- operator (0 or 1, optional): The unary logical operator to be applied to the condition. Data Type: xs:token.

  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - not
  - noOp (*default value*)
- condition (0 or 1, optional): A condition element for the rule. Data Type: xs:token.

  The attribute's value shall be represented as a member of a set of restricted vocabularies:

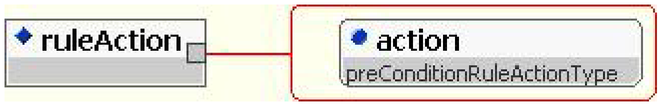
  - satisfied
  - objectiveStatusKnown
  - objectiveMeasure
  - objectiveMeasureGreaterThan
  - objectiveMeasureLessThan
  - completed
  - activityProgressKnown
  - attempted
  - attemptLimitExceeded
  - timeLimitExceeded
  - outsideAvailableTimeRange

   - always (*default value*)

Element:

- None

6.1.1.2.4.2 <ruleAction> Element



Description: The <ruleAction> element defines the desired sequencing behavior (action) if the rule evaluates to True.

Multiplicity: The <ruleAction> element shall occur 1 and only 1 time within the <preConditionRule>, <postConditionRule> and <exitConditionRule> element.

Attribute:

- action (1 and only 1, mandatory): The desired sequencing behavior if the rule evaluates to True. This action applies when traversing the activity tree to select an activity for delivery. Data Type: xs:token.

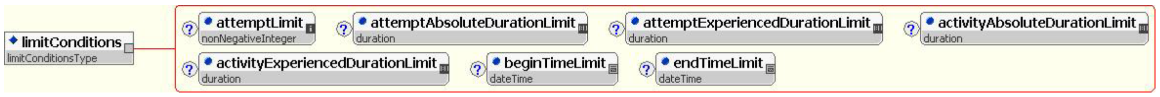
  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - skip
  - disabled
  - hidden
  - stop
  - ignore (*default value*)

Element:

- None

6.1.1.3 <limitConditions> Element



Description: The <limitConditions> element contains a set of factors constraining sequencing and delivery behavior associated with a learning activity in the activity tree.

Multiplicity: The <limitConditions> element shall occur 0 or 1 time within the <sequencing> element.

Attribute:

- attemptLimit (0 or 1, optional): This attribute contains an integer value that specifies the maximum number of attempts that may be taken on the learning activity. The default value for this attribute, if not defined, is unspecified and does not affect tracking of the number of attempts. Data Type: xs:integer (non-negative).

- attemptAbsoluteDurationLimit (0 or 1, optional): This attribute contains a value that specifies the maximum time duration that a learner is permitted to spend on a single attempt on the learning activity. The default value of this attribute, if not defined, is unspecified. Data Type: xs:time
- attemptExperiencedDurationLimit (0 or 1, optional): This attribute contains a value that specifies clock time duration that a learner is permitted to spend experiencing the learning activity. The default value of this attribute, if not defined, is unspecified. Data Type: xs:time

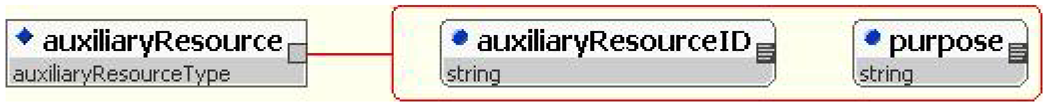
  ADL Note: This element replaces the <adlcp:maxtimeallowed> element defined in the SCORM Version 1.2. This element should be used to initialize the cmi.student_data.max_time_allowed data model element.

- activityAbsoluteDurationLimit (0 or 1, optional): This attribute contains a value that specifies the maximum actual time duration that a learner is permitted to spend experiencing the learning activity. The default value of this attribute, if not defined, is unspecified. Data Type: xs:time
- activityExperiencedDurationLimit (0 or 1, optional): The maximum time duration that a learner is permitted to spend on all attempts of the activity. The default value of this attribute, if not defined, is unspecified. Data Type: xs:time
- beginTimeLimit (0 or 1, optional): This attribute contains a date and time, or just time of day when the learning activity becomes available to be delivered. The default value of this attribute, if not defined, is 0 hours (beginning of the day). Data Type: xs:dateTime
- endTimeLimit (0 or 1, optional): The date and time, or just time of data when the availability of the learning activity ends. The default value of this attribute, if not defined, is 2400 hours (end of the day) Data Type: xs:dateTime.

Element:

- None.

6.1.1.4 <auxiliaryResource> Element



Description: The <auxiliaryResource> element describes the auxiliary resource associated with the activity.

Multiplicity: The <auxiliaryResource> element shall occur 0 or More times within the <auxiliaryResources> element.

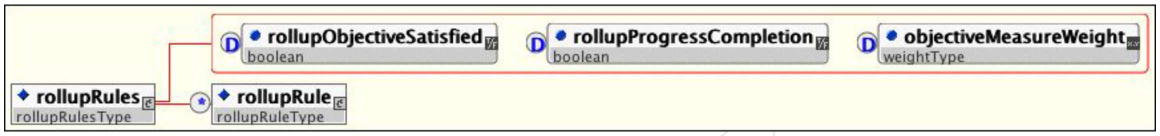
Attribute:

- auxiliaryResourceID (1 and only 1, required if auxiliary resource is defined): A unique identifier for the auxiliary resource. No default value is defined, this attribute is required if an auxiliary resource is defined. Data Type: xs:string.
- purpose (1 and only 1, required if auxiliary resource is defined): Indicates the purpose of the auxiliary resource. No default value is defined, this attribute is required if an auxiliary resource is defined. Data Type: xs:string.

Element:

- None

6.1.1.5 <rollupRules> Element



Description: The <rollupRules> element describes the set of information required to be supplied to aid in determining a parent nodes “rolled up” mastery/completion status.

Multiplicity: The <rollupRules> element shall occur 0 or 1 time within the <sequencing> element.

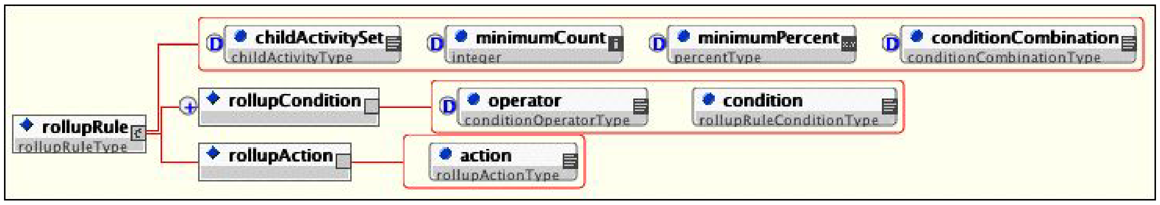
Attribute:

- rollupObjectiveSatisfied (0 or 1, optional): Indicates that the Objective value of *Objective Satisfied Status* for the objective associated with the activity is included in the rollup for the parent activity. This attribute is used when the *Objective Description* attribute value of *Objective Contributes to Rollup* equals True)The default value, if not defined, is “true”. Data Type: xs:boolean.
- rollupProgressCompletion (0 or 1, optional): This attribute specifies a Boolean (true/false) value that is used to determine whether the node for which it is defined contributes to the “rolled up” *ProgressStatus Complete* value of its parent node. The default value, if not defined, is “true”. Data Type: xs:boolean.
- objectiveMeasureWeight (0 or 1, optional): This attribute specifies a floating point value between 0.0 and 1.0 that is used to determine the weight of a learning activity’s score relative to its sibling activities’ scores when calculating the parent activity’s rolled up *MasteryStatus NormalizedScore* value. The default value, if not defined, is 1.0. Data Type: xs:float.

Element:

- rollupRule

6.1.1.5.1 <rollupRule> Element



Description: The <rollupRule> element is a container element that collects specific rollup rules for mastery and progress status evaluation for a parent node, based on the status of its child activities. The rollup rules is a set of four rules that define the conditions under which an activity is considered to be passed, failed, completed or incomplete based on the status of its child activities.

Multiplicity: The <rollupRule> element shall occur 0 or 1 time within the <rollupRules> element.

Attribute:

- childActivitySet (0 or 1, optional): The set of children of the activity whose data values are used to evaluate the rollup condition. Data Type: xs:token.

  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - all (default value)
  - any
  - none
  - atLeastCount
  - atLeastPercent

- minimumCount (0 or 1, optional): The number of children activities associated with a "childActivitySet" attribute of "AtLeastCount". The default value, if not defined, is 0. Data Type: xs:integer (Non-Negative).
- minimumPercent (0 or 1, optional): The percentage of children activities associated with a "childActivitySet" attribute value of "AtLeastPercent" The default value, if not defined, is 0.0. Data Type: xs:float (Real[0..1] Precision of at least 4 digits).
- conditionCombination (0 or 1, optional): How rollup conditions are combined in evaluating the rule. Data Type: xs:token

  The attribute's value shall be represented as a member of a set of restricted vocabularies:

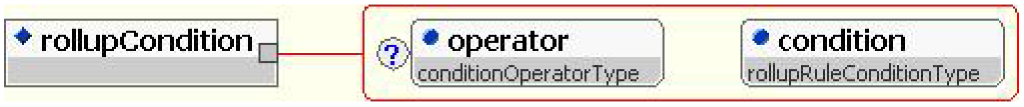
  - all
  - any (default value)

Element:

- rollupCondition

- rollupAction

6.1.1.5.1.1 <rollupCondition> Element



Description: The <rollupCondition> element is the rollup condition to be evaluated by the sequencing engine.

Multiplicity: The <rollupCondition> element shall occur 0 or 1 time within the <rollupRule> element.

Attribute:

- operator (0 or 1, optional): The unary logical operator to be applied to the condition. Data Type: xs:token.

  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - not
  - noOp

  The default value for this attribute, if not defined, is "noOp".

- condition (0 or 1, optional): A condition element for the rule. Data Type: xs:token.

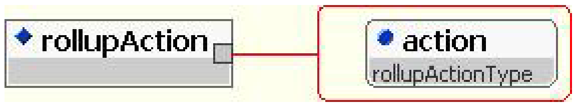
  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - satisfied
  - objectiveStatusKnown
  - objectiveMeasure
  - objectiveMeasureGreaterThan
  - objectiveMeasureLessThan
  - completed
  - activityProgressKnown
  - attempted
  - attemptLimitExceeded
  - timeLimitExceeded
  - outsideAvailableTimeRange
  - always (default value)

Element:

- None

6.1.1.5.2 <rollupAction> Element



Description: The <rollupAction> element contains a single sequencing rule action that belongs to the set of sequencing rules that apply to a learning activity.

Multiplicity: The <rollupAction> element shall occur 0 or 1 time within the <rollupRule> element.

Attribute:

- action (1 and only 1, mandatory): The desired rollup behavior if the rule evaluates to "true". Data Type: xs:token.

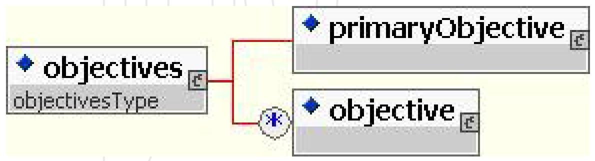
  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - satisfied (default value)
  - notSatisfied
  - completed
  - incomplete

Element:

- None

6.1.1.6 <objectives> Element



Description: The <objectives> element encapsulates all of the objectives.

Multiplicity: The <objectives> element shall occur 0 or 1 time within the <sequencing> element.

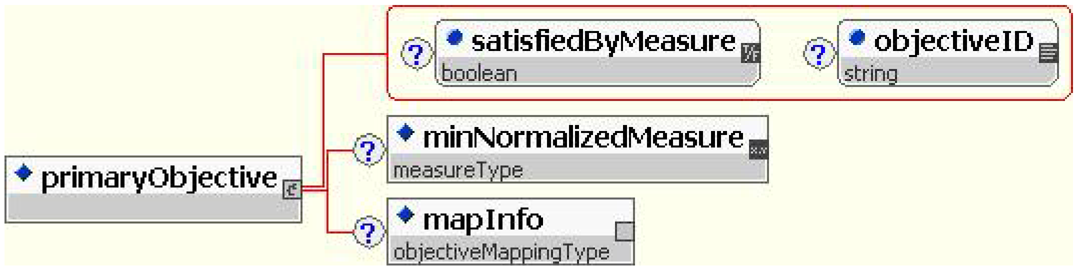
Attribute:

- None

Element:

- primaryObjective
- objective

6.1.1.6.1 <primaryObjective> Element



Description: The <primaryObjective> element describes the primary objective information for this item or organization.

Multiplicity: The <primaryObjective> element shall occur 0 or 1 time within the <objectives> element.

Attribute:

- satisfiedByMeasure (0 or 1, optional): Indicates that the <minNormalizedMeasure> element is to be used in place of any other method to determine if the objective associated with the activity has been satisfied. The default value, if not defined, is “false”. Data Type: xs:boolean.
- objectiveID (0 or 1, optional): The identifier of an objective associated with the activity. The ID is a link to the corresponding objective information. Data Type: xs:string.

Element:

- minNormalizedMeasure
- mapInfo

6.1.1.6.1.1 <minNormalizedMeasure> Element

Description: The <minNormalizedMeasure> element is the minimum satisfaction measure for the objective, normalized between –1..1 (inclusive). If the “objective” value of Objective Normalized Measure exceeds this value, the objective value of Objective Data Status is set to “true”. The value is unreliable unless Objective Satisfied by Measure is “true”. The default value, if not defined, is “1.0”. Data Type: xs:float (Real[-1..1] Precision of at least 4 significant decimal digits).

Multiplicity: The < minNormalizedMeasure > element shall occur 0 or 1 time within the <primaryObjective> element.

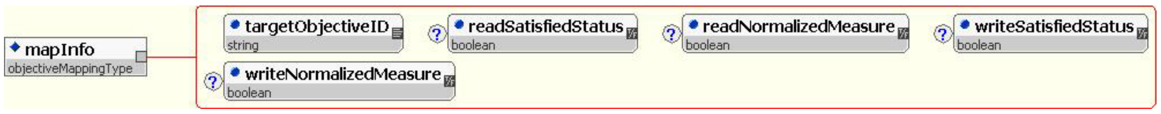
Attribute:

- None

Element:

- None

6.1.1.6.1.2 <mapInfo> Element



Description: The <mapInfo> element defines a mapping of an activity's local objective information to and from a shared global objective.

Multiplicity: The <mapInfo> element shall occur 0 or 1 time within the < primaryObjective > element.

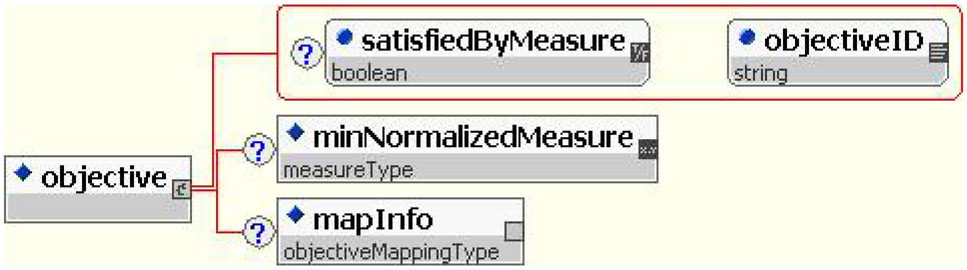
Attribute:

- targetObjectiveID (1 and only 1, mandatory): The identifier of global shared objective targeted for the mapping. Data Type: xs:string.
- readSatisfiedStatus (0 or 1, optional): Indicates whether or not the local objective value of *Objective Satisfied Status*, for the identified objective associated with the activity should be retrieved from the identified shared global objective (*targetObjectiveID*) when the progress for the local objective is undefined. The default value, if not defined, is "false". Data Type: xs:boolean.
- readNormalizedMeasure (0 or 1, optional): Indicates whether or not the local objective value of *Objective Normalized Measure*, for the identified objective associated with the activity should be retrieved from the identified shared global objective (*targetObjectiveID*) when the progress for the local objective is undefined. The default value, if not defined, is "true". Data Type: xs:boolean.
- writeSatisfiedStatus (0 or 1, optional): Indicates whether or not the local objective value of *Objective Satisfied Status*, for the identified objective associated with the activity should be transferred to the identified global shared objective (*targetObjectiveID*) upon exit of the activity. The default value, if not defined, is "false". Data Type: xs:boolean.
- writeNormalizedMeasure (0 or 1, optional): Indicates whether or not the local objective value of *Objective Normalized Measure*, for the identified objective associated with the activity should be transferred to the identified global shared objective (*targetObjectiveID*) upon exit of the activity. The default value, if not defined, is "false". Data Type: xs:boolean.

Element:

- None

6.1.1.6.2 <objective> Element



Description: The <objective> element describes a single objectives information for this item or organization.

Multiplicity: The <objective> element shall occur 0 or 1 time within the <objectives> element.

Attribute:

- satisfiedByMeasure (0 or 1, optional): Indicates that the <minNormalizedMeasure> element is to be used in place of any other method to determine if the objective associated with the activity has been satisfied. The default value, if not defined, is "false". Data Type: xs:boolean.
- objectiveID (0 or 1, optional): The identifier of an objective associated with the activity. The ID is a link to the corresponding objective information. Data Type: xs:string.

Element:

- minNormalizedMeasure
- mapInfo

6.1.1.6.2.1 <minNormalizedMeasure> Element

Description: The <minNormalizedMeasure> element is the minimum satisfaction measure for the objective, normalized between –1..1 (inclusive). If the "objective" value of Objective Normalized Measure exceeds this value, the objective value of Objective Data Status is set to "true". The value is unreliable unless Objective Satisfied by Measure is "true". The default value, if not defined, is "1.0". Data Type: xs:float (Real[-1..1] Precision of at least 4 significant decimal digits).

Multiplicity: The < minNormalizedMeasure > element shall occur 0 or 1 time within the < objective > element.

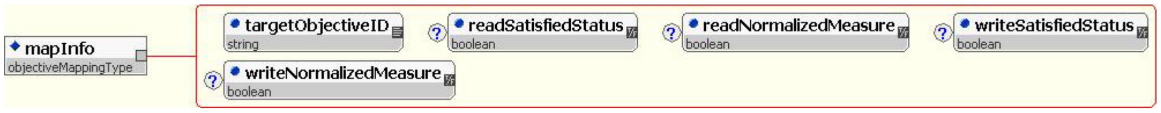
Attribute:

- None

Element:

- None

6.1.1.6.2.2 <mapInfo> Element



Description: The <mapInfo> element defines a mapping of an activity's local objective information to and from a shared global objective.

Multiplicity: The < mapInfo > element shall occur 0 or 1 time within the < objective > element.

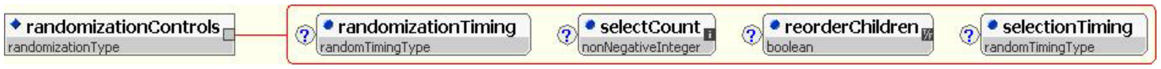
Attribute:

- targetObjectiveID (1 and only 1, mandatory): The identifier of global shared objective targeted for the mapping. Data Type: xs:string.
- readSatisfiedStatus (0 or 1, optional): Indicates whether or not the local objective value of *Objective Satisfied Status*, for the identified objective associated with the activity should be retrieved from the identified shared global objective (*targetObjectiveID*) when the progress for the local objective is undefined. The default value, if not defined, is "false". Data Type: xs:boolean.
- readNormalizedMeasure (0 or 1, optional): Indicates whether or not the local objective value of *Objective Normalized Measure*, for the identified objective associated with the activity should be retrieved from the identified shared global objective (*targetObjectiveID*) when the progress for the local objective is undefined. The default value, if not defined, is "true". Data Type: xs:boolean.
- writeSatisfiedStatus (0 or 1, optional) Indicates whether or not the local objective value of *Objective Satisfied Status*, for the identified objective associated with the activity should be transferred to the identified global shared objective (*targetObjectiveID*) upon exit of the activity. The default value, if not defined, is "false". Data Type: xs:boolean.
- writeNormalizedMeasure (0 or 1, optional): Indicates whether or not the local objective value of *Objective Normalized Measure*, for the identified objective associated with the activity should be transferred to the identified global shared objective (*targetObjectiveID*) upon exit of the activity. The default value, if not defined, is "false". Data Type: xs:boolean.

Element:

- None

6.1.1.7 <randomizationControls> Element



Description: The <randomizationControls> element describe how the children of an activity should be ordered during the sequencing process.

Multiplicity: The <randomizationControls> element shall occur 0 or 1 time within the <sequencing> element.

Attribute:

- randomizationTiming (0 or 1, optional): Indicates when the ordering of the children of the activity should occur. Data Type: xs:token.

  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - never (default value)
  - once
  - onEachNewAttempt
- selectCount (0 or 1, optional): Indicates the number of child activities that must be selected from the set of child activities associated with the activity. If this value is larger than the number of child activities, all child activities are selected. This value is unreliable unless Selection Count Status is "true". If Selection Count Status is "false" all child activities are selected. The default value, if not defined, is 0. Data Type: xs:integer (Non-Negative).
- reorderChildren (0 or 1, optional): Indicates the selection count data is meaningful for the activity.. The attribute defaults to false if not present in the <controlMode> element. Data Type: xs:boolean.
- selectionTiming (0 or 1, optional): Indicates when selection should occur. Data Type: xs:token.

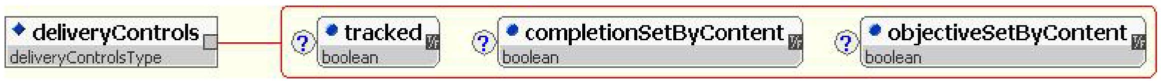
  The attribute's value shall be represented as a member of a set of restricted vocabularies:

  - never (default value)
  - once
  - onEachNewAttempt

Element:

- None

6.1.1.8 <deliveryControls> Element



Description: The <deliveryControls> element describes actions and controls used when an activity is delivered, i.e., Objective, Activity, and Attempt Progress Data are recorded when the activity is delivered.

Multiplicity: The <deliveryControls> element shall occur 0 or 1 time within the <sequencing> element.

Attribute:

- tracked (0 or 1, optional): Indicates that Objective and Attempt Progress information for the activity attempt should be recorded and the data will contribute to the rollup for the parent activity. How the data is tracked and recorded is not specified. The attribute defaults to true if not present in the < deliveryControls > element. Data Type: xs:boolean.
- completionSetByContent (0 or 1, optional): Indicates that the Activity Attempt Completion Status information for the activity in the Tracking Model will be set by the content object. A "false" value indicates that default rules will be used to set progress data. For a "true" value, how, if or when the completion data is set is not specified. The attribute defaults to false if not present in the < deliveryControls > element. Data Type: xs:boolean.
- objectiveSetByContent (0 or 1, optional): Indicates that the Objective Status information for the associated objective that has the attribute value objective contributes to Rollup value of "true" with the Tracking Model will be set by the content object. A "false" value indicates that default rules will be used to set objective data. For a "true" value, how, if or when the objective data is set is not specified. The attribute defaults to false if not present in the < deliveryControls > element. Data Type: xs:boolean.

Element:

- None

6.1.2. Relationship to Content Packaging

The IMS Content Packaging Specification provides a structure for relating a learning activity to a content resource – the *item* element and its relationship to a *resource* element. Furthermore, *item* elements can be clustered into collections, with such collections contained in a parent *organization* element, as learning activities may be clustered together in a parent activity or activities. Therefore, IMS Simple Sequencing maps the concept of a learning activity to an *item* element, a collection of *item* elements within an *organization* element and to an *organization* element itself as defined by the Content Packaging Specification. The Content Packaging XML Binding is extended by this specification to define how sequencing information is associated with packaged content.

The process of defining a specific sequence of learning activities begins with the creation of an aggregation of content to be interchanged using a SCORM Content Aggregation Application Profile of the IMS Content Package specification. As shown in the figure below, the Content Packaging *organization* element and each *item* element within it can have defined sequencing behaviors through the association of sequencing information:

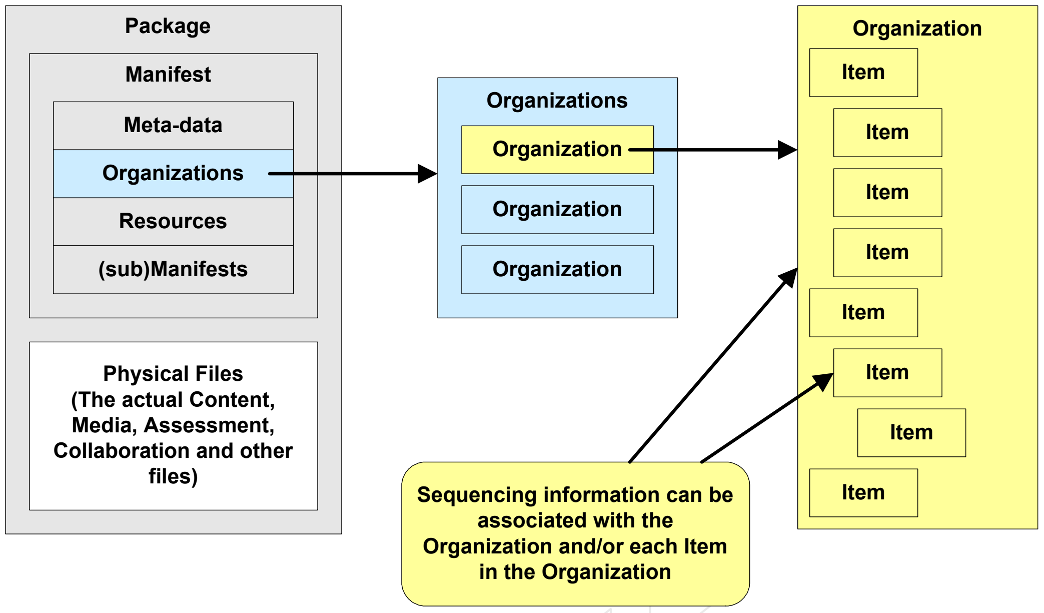

Package
Manifest
Meta-data
Organizations
Resources
(sub)Manifests
Physical Files (The actual Content, Media, Assessment, Collaboration and other files)
Organizations
Organization
Organization
Organization
Organization
Item
Item
Item
Item
Item
Item
Item
Item
Sequencing information can be associated with the Organization and/or each Item in the Organization

Figure 6.1.2a: Content Packaging Structure

Sequencing information can also be associated at a resource level (<resourceSequencing> element), as depicted in the following diagram.

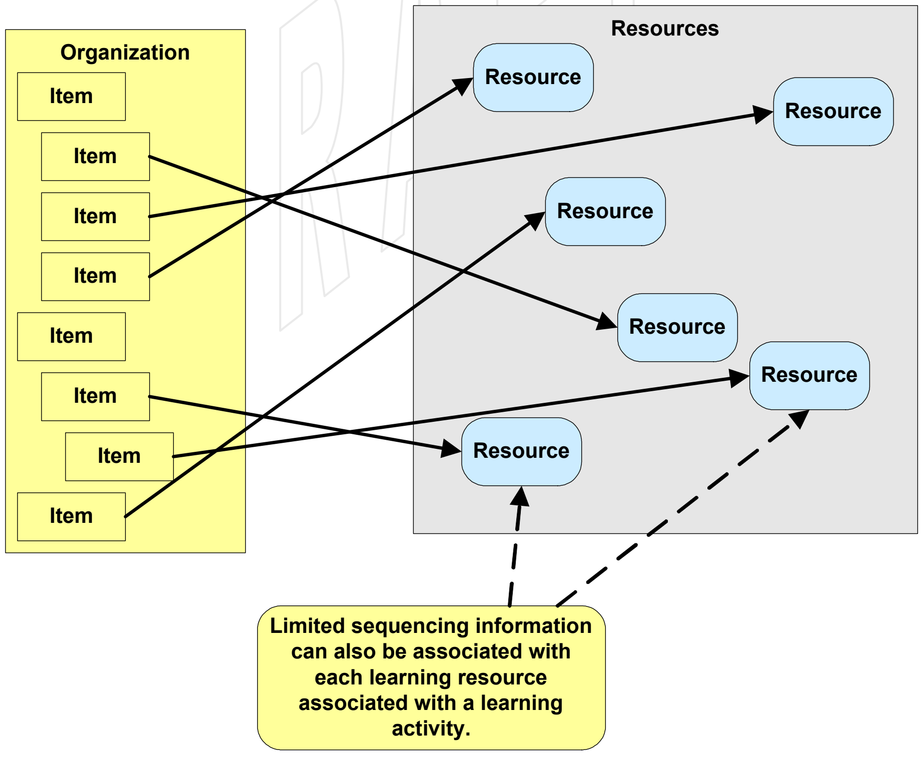

Organization
Item
Item
Item
Item
Item
Item
Item
Item
Resources
Resource
Resource
Resource
Resource
Resource
Resource
Limited sequencing information can also be associated with each learning resource associated with a learning activity.

Figure 6.1.2b: Item to Resource Relationship

All SCORM conformant Content Aggregation packages include, by default, sequencing information. If a SCORM Content Package does not include any sequencing rules, the implied default behavior is to allow a learner to freely choose any activity with no guidance or constraints.

6.1.2.1 Changes to the SCORM 1.2 Content Packaging Profiles

The SCORM Content Packaging Application Profile includes several elements required by LMSs to process SCORM Content Packages. These elements are included in the *adlcp* namespace and where applied as an extension to the IMS Content Packaging specification.

The IMS Simple Sequencing specification defines several elements that serve the same function as adlcp namespace elements. Where appropriate, adlcp namespace elements have been replaced with their corresponding IMS Simple Sequencing elements.

The *adlcp* namespace elements and their corresponding IMS SS listed below.

- adlcp:location: This element remains in the ADL Namespace. Its function is not replaced by any IMS Simple Sequencing Specification element.
- adlcp:datafromlms: This element remains in the ADL Namespace. Its function is not replaced by any IMS Simple Sequencing Specification element.
- adlcp:prerequisites: This element has been deprecated from the ADL namespace. Its function conflicts with the explicit sequencing elements defined by IMS Simple Sequencing specification. The sequencing rules defined in the IMS Simple Sequencing Specification supercedes this element. Since this element is deprecated, LMSs are not guaranteed to support this element.
- adlcp:maxtimeallowed: This element has been deprecated from the ADL Namespace. It is replaced by the IMS Simple Sequencing *Activity Attempt Experienced Duration Limit* element. Since this element is deprecated and has been replaced by a specific IMS Sequencing element, LMSs are not guaranteed to support this element.
- adlcp:timelimitaction: This element remains in the ADL Namespace. Its function is not replaced by any IMS Simple Sequencing Specification element.
- adlcp:masteryscore: This element has been deprecated from the ADL Namespace. It is replaced by using the IMS Simple Sequencing *Objective Minimum Satisfied Normalized Measure* element. Since this element is deprecated and has been replaced by a specific IMS Sequencing element, LMSs are not guaranteed to support this element.
- adlcp:scormtype: This element remains in the ADL Namespace. Its function is not replaced by any IMS Simple Sequencing Specification element. The only change was the addition of the "sca" to the list of fixed vocabularies.

SECTION 7
Sequencing Behaviors

This page intentionally left blank.

7.1. Sequencing Behaviors

This section describes the behaviors associated with the sequencing processes described in the IMS SS Specification. The descriptions that follow are not intended to replace the detailed behavioral descriptions (pseudo code) included in the SS Specification; instead, they are intended to help distill the primary features and characteristics of those processes.

The IMS SS Specification includes two data models that apply to each activity in the activity tree and a set of sequencing behaviors. The data models and their relation to the activities can be summarized:

- **Tracking Model** – Captures information gathered from the learner's interaction with learning resources associated with activities. This is a dynamic run-time (while the learner is interacting with an LMS) data model.
- **Sequencing Definition Model** – Describes how the various sequencing processes utilize and interpret the Tracking Model information to 'sequence' activities. This is a static (defined in the Content Package) data model describing authored sequencing intentions for a give content aggregation. Section 5 describes the Sequencing Definition Model in detail.

For each activity, the set of all data model elements (from both data models) that apply to that activity can be considered the activity's state. The various sequencing behaviors are described as independent processes. Each process utilizes pieces of the two data models to exhibit well-defined behavior, but do not rely on any of the other sequencing processes. The Overall Sequencing Process defines how all of the sequencing processes relate to one another within the context of a 'sequencing session'.

7.1.1. Tracking Status Model

7.1.1.1 Introduction

In previous version of the SCORM, the only data model that was defined was the SCORM Run-Time Environment Data Model. This information was used to track the learner's interaction with a SCO. There was always some confusion on how or what information was tracked about the activity that is associated with the resource. With the addition of the IMS Simple Sequencing Specification, an additional data model is more specifically defined for an LMS to manage – The Tracking Status Model. The Tracking Status Model is a collection of dynamic, sequencing state information associated with each node of the activity tree. This set of state data is associated with each node in the activity tree for each learner. The initial values for these data items are defined in the sequencing rules contained in SCORM content package. During the learning experience this state information is updated as the learner interacts with the activities and learning resources (if applicable). The state information is implementation specific based on the defined sequencing rules and state data requirements defined in this application profile.

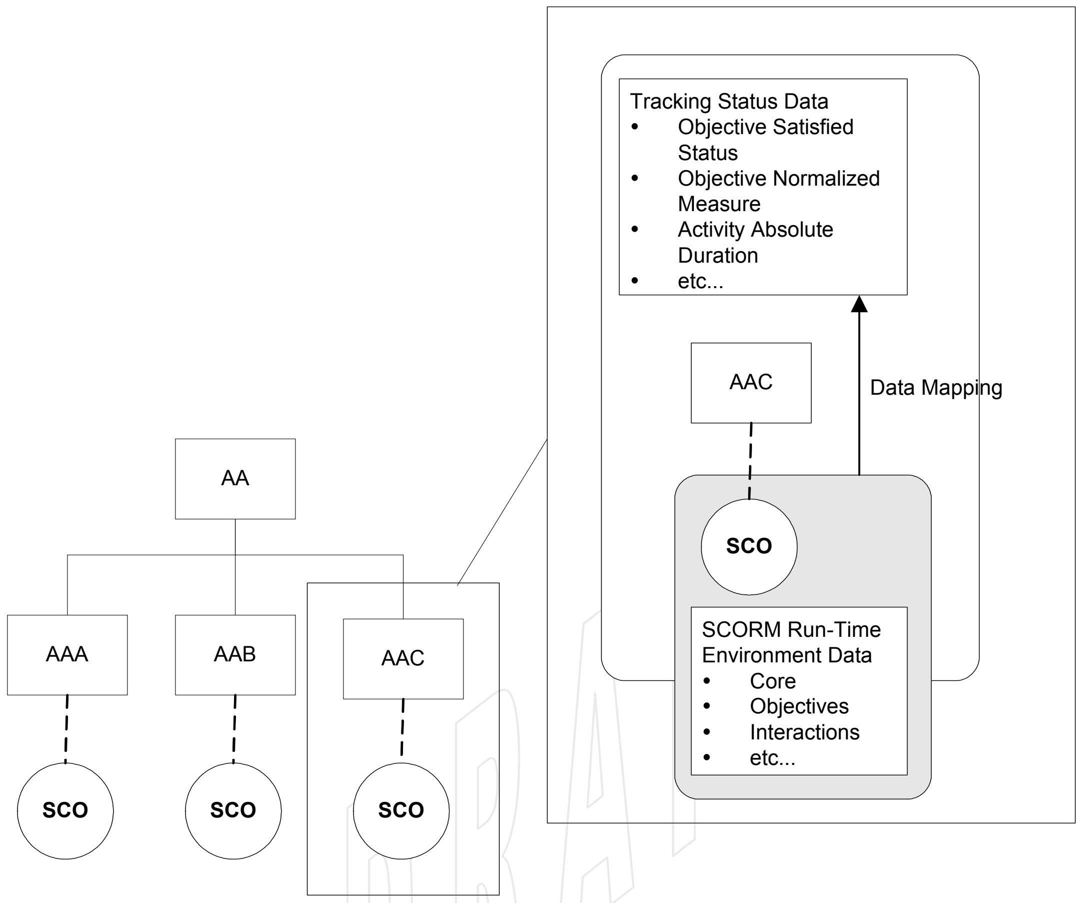

Tracking Status Data
• Objective Satisfied Status
• Objective Normalized Measure
• Activity Absolute Duration
• etc...
AAC
Data Mapping
AA
SCO
AAA
AAB
AAC
SCORM Run-Time Environment Data
• Core
• Objectives
• Interactions
• etc...
SCO
SCO
SCO

Figure7.1.1.1a: Conceptual illustration of the Run-Time State Model

The SCORM Run-Time Environment Data Model, defined in the SCORM Version 1.2, is used by learning resources (SCOs) to define information being communicated (e.g. status, scores). The LMS must maintain the run-time state of the required activities that are associated with these learning resources. The LMS must maintain this state across sessions and the learning resources must utilize only these predefined data elements if reuse across multiple LMSs is to occur. Certain SCORM Run-Time Environment Data Model elements have a major impact on the elements defined in the Tracking Status Model. These relationships and behaviors are explained in the following sections.

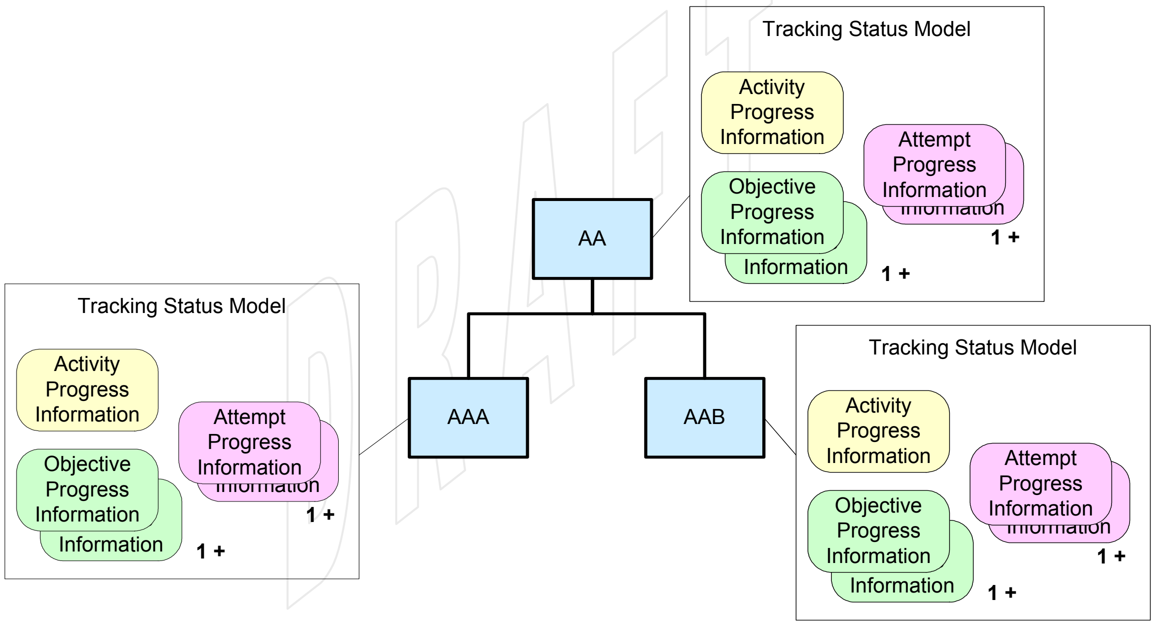
To enable conditional sequencing of activities, information about a learner's interactions with the learning resources associated with activities must be maintained and managed. The IMS SS Specification describes tracking information that must be maintained for each activity in the activity tree – the set of data model elements that describe the tracking information is called the Tracking Model.

SCORM does not impose any implementation requirements on how the tracking model is represented or managed. Furthermore, there is no requirement that only one set of tracking information exist for each activity. Implementations are free to manage multiple sets of tracking information, which can be used to perform 'what-if' evaluations of activity tree state, enabling 'intelligent' run-time behaviors. However, when the various sequencing processes are applied within the context of the Overall Sequencing Process, the processes must all utilize the same set of valid tracking information as described in the IMS SS Specification.

The sections below describe the Tracking Model elements and the behaviors for managing tracking information.

7.1.1.2 Tracking Status Model

The IMS Simple Sequencing Specification enables sequencing behavior to be conditional, based on tracking status, including progress and mastery information associated with learning activities. The IMS Simple Sequencing Specification defines a common vocabulary of status information on which run-time sequencing decisions can be based.



AA
AAA
AAB
Tracking Status Model
Activity Progress Information
Objective Progress Information
Attempt Progress Information
1 +
1 +

Figure 7.1.1.2a: Tracking Status Model

Activities have associated Tracking Status information that is specific to each learner that experiences the activity. It is assumed that an LMS updates tracking information at run-time as a result of learner interaction with delivered learning resources that support activities. For those activities that have associated learning resources that communicate (SCOs), the LMS shall manage the tracking status model based on communication from the learning resource. This management process determines the state of all of the learning activities defined, which in turn, determines the sequence in which the learning activities should be delivered. For those learning resources that do not communicate there are some defined requirements in the following sections to help LMSs manage the tracking status model.

Changes in the state of tracking status elements for an activity affect the state of the activity's parent. The process of evaluating the effects of change in the state of an activity on its parent is referred to in the IMS Sequencing Specification as "rollup". The "rollup" of information needs to be handled by an LMS.

Tracking status information can be used by an LMS to control sequencing behavior through the association of sequencing rules to activities by a content author. Certain rules described in the IMS Simple Sequencing Specification are dependent upon the values of the elements of the tracking status model at run-time. This dependency is specified to enable dynamic sequencing behavior based on learner progress and/or mastery of activities, and the state of an activity itself. These rules are evaluated at run-time to determine which activity in an organization of activities is delivered in response to a sequencing request.

Tracking status information elements must be maintained for each activity in the activity tree. Activities may be aggregated together with other activities to form higher-level activities. Activities may be nested to any desired depth. A sub-activity is always experienced in the context of its parent activity, if one exists.

The Tracking Status Model describes the information that must be maintained by a system that delivers sequenced activities. An LMS must be able to receive and maintain the tracking status information for each activity defined. The tracking status model defines the following set of tracking status information:

- **Objective Progress Information:** Results of the learner's interactions related to an objective.
- **Activity/Attempt Progress Information:** Describes a learner's progress on an activity. This information describes the cumulative progress information for an individual activity.

7.1.1.3 Objective Progress Information

An activity may have any number of objectives associated with it. The characteristics and properties of each objective is described by the Sequencing Definition elements *Objective Description* and *Objective Map*.

For each attempt on an activity, a learner gets one set of Objective Progress Information (Table 7.1.1.3a) for each objective associated with that activity.

No.	Name	Description	Value Space	Default Value
1	*Objective Progress Status*	Indicates the objective has a satisfaction value (True or False).	Boolean	False
2	*Objective Satisfied Status*	Indicates the objective is satisfied (True or False). The determination or meaning of satisfied or not satisfied is not defined in this model. The value is unreliable unless *Objective Progress Status* is True.	Boolean	False
3	*Objective Measure Status*	Indicates the objective has a measure value (True or False).	Boolean	False
4	*Objective*	The measure (e.g., score) for the objective,	Real [-1..1]	0.0

	Normalized Measure	normalized between -1..1 (inclusive). The mechanism to normalize a measure is not defined in this model. The value is unreliable unless *Objective Measure Status* are True.	Precision of at least 4 significant decimal digits	

Table 7.1.1.3a: Objective Tracking Information

The detailed sequencing behaviors, described in the IMS SS pseudo code, utilize data model pairs – one element describes tracked information and the other describes if that that tracked information is valid. For example, *Objective Satisfied Status* describes if the objective has been satisfied or not, and *Objective Progress Status* describes if the value of *Objective Satisfied Status* is valid. Because tracking information is a run-time data model, implementers are free to represent these values as they please to optimize their system; however, their systems must exhibit the behaviors described in the IMS SS pseudo code.

To make this document more readable, only one element will be used to describe each pair, and an 'unknown' value will be added to its vocabulary. For example, *Objective Satisfied Status* will be described using the following vocabulary:

- **satisfied** – *Objective Progress Status* == true; *Objective Satisfied Status* == true
- **not satisfied** – *Objective Progress Status* == true; *Objective Satisfied Status* == false
- **unknown** – *Objective Progress Status* == false

Similarly, in addition to its normal floating point range, *Objective Normalized Measure* will be described as having the value unknown to represent *Objective Measure Status* == false.

7.1.1.4 Activity Progress Information

Each activity has one set of information that spans all attempts on that activity; this information is the Activity Progress Information (Table 7.1.1.4a).

No.	Name	Description	Value Space	Default Value
1	*Activity Progress Status*	Indicates the activity progress information is (True or False) meaningful for the activity.	Boolean	False
2	*Activity Absolute Duration*	The cumulative duration of all attempts on the activity, i.e., the time from the initial start of the activity to the end of the activity. The mechanism for determining the duration is not defined in this model. The value is unreliable unless *Activity Progress Status* is True.	Duration Accuracy 0.1 second	0.0
3	*Activity Experienced Duration*	The cumulative *experienced* duration of all attempts on the activity, i.e., the time from the initial start of the activity to the end of the activity, not including any time elapsed while the activity is suspended (i.e., when the activity is not being experienced or is inactive). The mechanism for determining the duration of the suspend time is not defined in this model.	Duration Accuracy 0.1 second	0.0

		The value is unreliable unless *Activity Progress Status* is True and the activity is a leaf.		
4	*Activity Attempt Count*	The number of attempts on the activity. The count includes the current attempt, i.e., 0 means the activity was not attempted and 1 or greater means it either is in progress or completed. The mechanism and timing of when the *Activity Attempt Count* is incremented is not defined in this model. The value is unreliable unless *Activity Progress Status* is True.	Non Negative Integer	0

Table 7.1.1.4a: Activity Progress Information

The Activity Progress data model elements are managed as follows:

- *Activity Progress Status* is set to true when the first attempt on the activity begins.
- *Activity Absolute Duration* is the total absolute duration of all attempts on the activity.
- *Activity Experienced Duration* is the cumulative *experienced* duration of all attempts on the activity
- *Activity Attempt Count* is incremented when each new attempt begins.

7.1.1.5 Attempt Progress Information

For each attempt on an activity, a learner gets one set of Attempt Progress Information (Table 7.1.1.5a).

No.	Name	Description	Value Space	Default Value
1	*Activity Attempt Progress Status*	Indicates the attempt progress information (True or False) is meaningful for the activity attempt. The value is unreliable unless *Activity Attempt Count* is greater than (>) 0.	Boolean	False
2	*Activity Attempt Completion Amount*	The measure of the completion of the attempt on the activity, normalized between 0..1 (inclusive) where 1 means the activity attempt is complete and any lesser value means the activity attempt is not complete. The mechanism to define the completion amount is not defined in this model. The value is unreliable unless *Activity Attempt Progress Status* is True.	Real [0..1] Precision of at least 4 significant decimal digits	0.0
3	*Activity Attempt Completion Status*	Indicates the activity attempt is completed (True or False). The determination or meaning of completed or incomplete is not defined in this model. The value is unreliable unless *Activity Attempt Progress Status* is True.	Boolean	False
4	*Activity Attempt Absolute Duration*	The duration of the attempt on the activity, i.e., time from the start of the attempt to the end of the attempt. The mechanism for determining the duration is not defined in this model. The value is unreliable unless *Activity Attempt Progress Status* is True.	Duration Accuracy 0.1 second	0.0

5	*Activity Attempt Experienced Duration*	The *experienced* duration of the attempt on the activity, i.e., the time from the start of the attempt to the end of the attempt, not including elapsed time while the activity attempt is suspended (i.e., when the activity attempt is not being experienced or is inactive). The mechanism for determining the duration or the suspend time is not defined in this model. The value is unreliable unless *Activity Attempt Progress Status* is True.	Duration Accuracy 0.1 second	0.0

Table 7.1.1.5a: - Attempt Progress Information

The detailed sequencing behaviors, described in the IMS Simple Sequencing Specification pseudo code, utilize data model pairs – one element describes tracked information and the other describes if that that tracked information is valid. For example, *Activity Attempt Completion Status* describes if the attempt has been completed or not, and *Activity Attempt Progress Status* describes if the value of *Activity Attempt Completion Status* is valid. Because tracking information is a run-time data model, implementers are free to represent these values as they please to optimize their system; however, their systems must exhibit the behaviors described in the IMS SS pseudo code.

To make this document more readable, only one element will be used to describe each pair, and an 'unknown' value will be added to its vocabulary. For example, *Activity Attempt Completion Status* will be described using the following vocabulary:

- **completed** – *Activity Attempt Progress Status* == true; *Activity Attempt Completion Status* == true
- **incomplete** – *Activity Attempt Progress Status* == true; *Activity Attempt Completion Status* == false
- **unknown** – *Activity Attempt Progress Status* == false

Similarly, in addition to their defined duration type, *Activity Attempt Absolute Duration* and *Activity Attempt Experienced Duration* will be described as having the value unknown to represent *Activity Attempt Progress Status* == false.

The current version of the IMS SS Specification does not utilize the *Activity Attempt Completion Amount* value, other than to set it. SCORM does not define any additional behavior for this element.

7.1.1.6 Activity State Information

The sequencing engine maintains additional state information for each activity in the activity tree. This information is called Activity State Information (Table 7.1.1.6a); it is utilized and managed by the various sequencing process during the Overall Sequencing Process.

No.	Name	Description	Value Space	Default Value
1	*Activity is Active*	Indicates that an attempt is currently in progress for the activity, i.e. the activity has been delivered to the learner and has not been exited, or is an	Boolean	False

		ancestor of the current activity (True or False).		
2	*Activity is Suspended*	Indicates the activity is currently suspended (True or False).	Boolean	False
3	*Available Children*	A list indicating the ordering of the available child activities for the activity.	Ordered List of Activities	All children

Table 7.1.1.6a: Activity State Information

The Activity State data model elements are managed as follows:

- *Activity is Active* is set to true when an attempt on the activity begins, and is set to false when the attempt on the activity ends. This element has several characteristics and effect in the context of the various sequencing processes:
    - There can only be one 'active path' in the activity tree at any one time – only the activities along the 'active path' can have *Activity is Active* == true. The 'active path' begins at the root of the tree and ends at the *Current Activity* (see Global State Information in the next section).
    - Only one (or no) leaf activity may have *Activity is Active* == true. If a leaf activity has *Activity is Active* == true, that activity must be the *Current Activity*.
    - The *Current Activity* will only have *Activity is Active* == true, if it has not already exited.
- *Activity is Suspended* may be set to true when the current attempt on the activity ends. This is done in one of two ways depending on the type of activity.
    - If the activity is a leaf, the activity's associated learning resource or the LMS may indicate that the activity's learning resource exited with suspend.
    - If the activity is a cluster, the sequencing engine will set the cluster to suspended if any of its children are suspended.
- *Available Children* maintains the ordered list of the activity's children; this element only applies to activity clusters. This element is implicitly utilized during Navigation Behavior, Sequencing Behavior, and Delivery Behavior as the set of children to be sequenced. It is set by the Selection and Random Behavior.

7.1.1.7 Global State Information

The sequencing engine maintains additional state information for the activity tree. This information is called Global State Information (Table 7.1.1.7a); it is utilized and managed by the various sequencing process during the Overall Sequencing Process

No.	**Name**	**Description**	**Value Space**	**Default Value**
1	*Current Activity*	Indicates the current activity. If an activity is being experienced by the learner, the current activity is the activity delivered by the most recently completed *Content Delivery Environment Process* (see Delivery	Activity	None

		Behavior). If an activity is not being experienced by the learner, the current activity is the activity identified to exit by the most recently completed exit process (see Exit Behavior)		
2	*Suspended Activity*	Indicates the activity from which a *Suspend All* navigation request was triggered.	Activity	None

Table 7.1.1.7a: Global State Information

- *Current Activity* indicates the unique activity in the activity tree being tracked by the sequencing engine. All of this activity's ancestors must have *Activity is Active* == true.
- *Suspended Activity* indicates the unique leaf activity that was the *Current Activity* in the previous sequencing session; a sequencing session that ended due to a SuspendAll exit request.

Figure 7.1.1.8a depicts the state model of the *Current Activity*; it summarizes the effects of the various sequencing processes on the *Current Activity* and the current activity's state.

1. Starting at the red star, a learning resource has been delivered to the learner; the resource's associated activity is the *Current Activity* – it must be leaf activity and it is now active. Only the ancestors (along the 'active path') of the current activity are active.
2. The state of the activity tree remains like this until some navigation request triggers the Overall Sequencing Process. If the navigation request is valid, it will result in an exit request to 'end' the attempt on the current activity – the current activity exits and its *Activity is Active* == false. The *Current Activity* remains the leaf activity.
3. When a leaf activity exits, the system may indicate the condition of the exit was to suspend – this is done in the Completion Subprocess. If the system indicates this, the leaf activity has its *Activity is Suspended* == true.
4. During the Exit Behavior Process, Exit Action Rules are evaluated on all of the ancestors of the leaf activity – this is done in the Sequencing Exit Action Rule Subprocess. The result of this subprocess will be that either the leaf activity remains the *Current Activity*, or an ancestor of the leaf activity becomes the *Current Activity*.
5. During the Exit Behavior Process, post condition rules are evaluated on the *Current Activity*.
6. The Sequencing Behavior processes any pending sequencing request, which may result in a delivery request.
7. The Delivery Behavior process validates any pending delivery request. If the delivery request validates, the target activity becomes the Current Activity and an attempt is started on it – *Activity is Active* == true; *Activity is Suspended* == false.
8. Go to Step 1.

7.1.1.8 Tracking Behavior

The information in this section is intended to supplement, not replace, the Tracking Model Behavior section of the IMS SS Specification. Please refer to the IMS SS Specification for more details.

By default, the set of objective information for each objective only applies to the activity that defined that objective – these are called 'local' objectives. Other activities cannot directly reference the objective information associated with another activity. To enable multiple activities to reference the same objective information, an *Objective Map* must be defined for both activities. Each *Objective Map* defines a relationship between the 'local' objective and some global shared objective. There are two kinds of maps:

- A 'write' *Objective Map* will set the corresponding global shared objective field(s) to the value(s) of the 'local' objective. This operation is done each time the 'local' objective information changes.
- A 'read' *Objective Map* will read the corresponding global shared objective field(s) whenever the 'local' objective value is required, but is unknown (*Objective Measure Status* == false or *Objective Progress Status* == false). This operation does not alter the 'local' objective information.

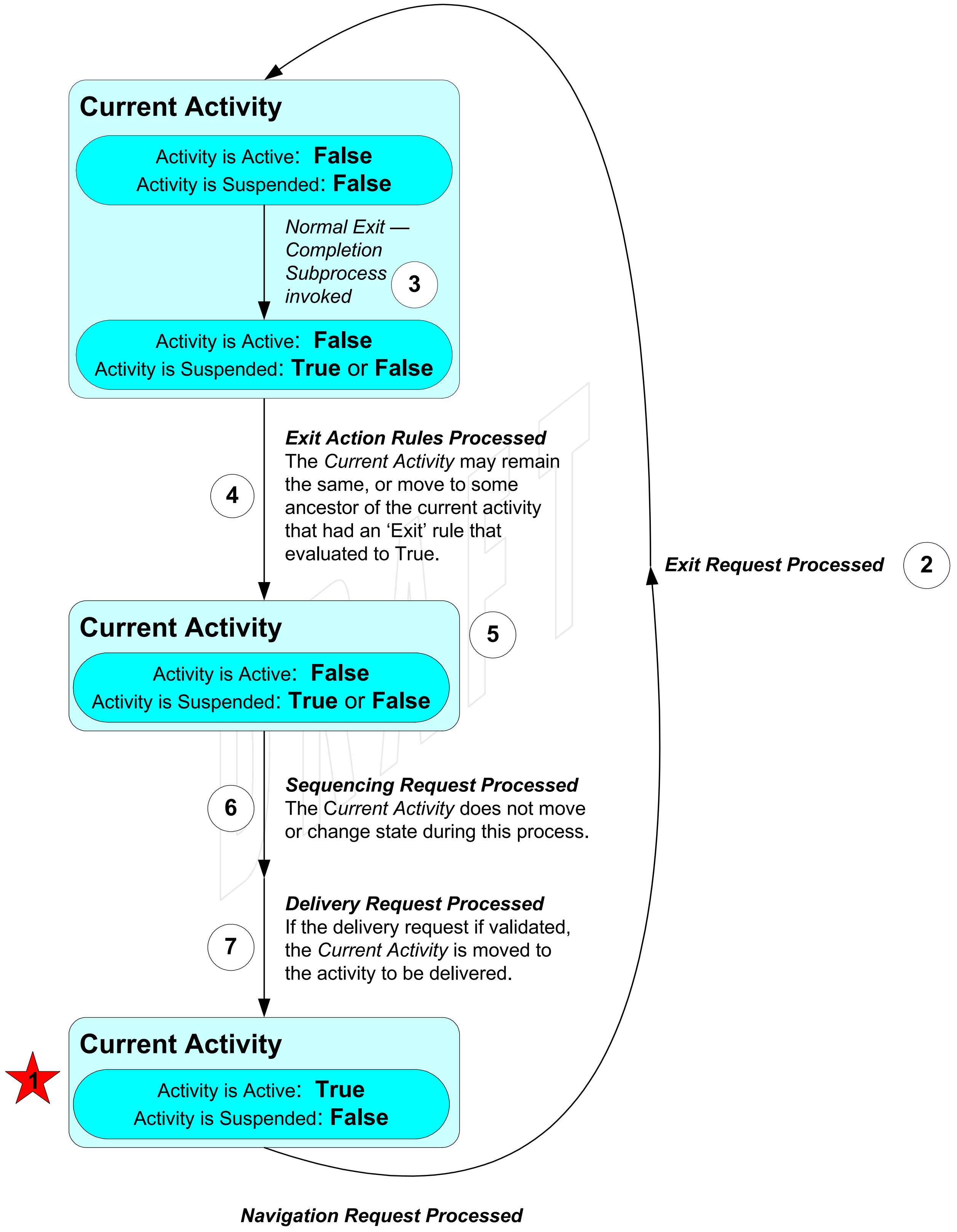

Current Activity
Activity is Active: False
Activity is Suspended: False
Normal Exit — Completion Subprocess invoked
3
Activity is Active: False
Activity is Suspended: True or False
Exit Action Rules Processed
The Current Activity may remain the same, or move to some ancestor of the current activity that had an 'Exit' rule that evaluated to True.
4
Exit Request Processed
2
Current Activity
5
Activity is Active: False
Activity is Suspended: True or False
Sequencing Request Processed
The Current Activity does not move or change state during this process.
6
Delivery Request Processed
If the delivery request if validated, the Current Activity is moved to the activity to be delivered.
7
Current Activity
1
Activity is Active: True
Activity is Suspended: False
Navigation Request Processed

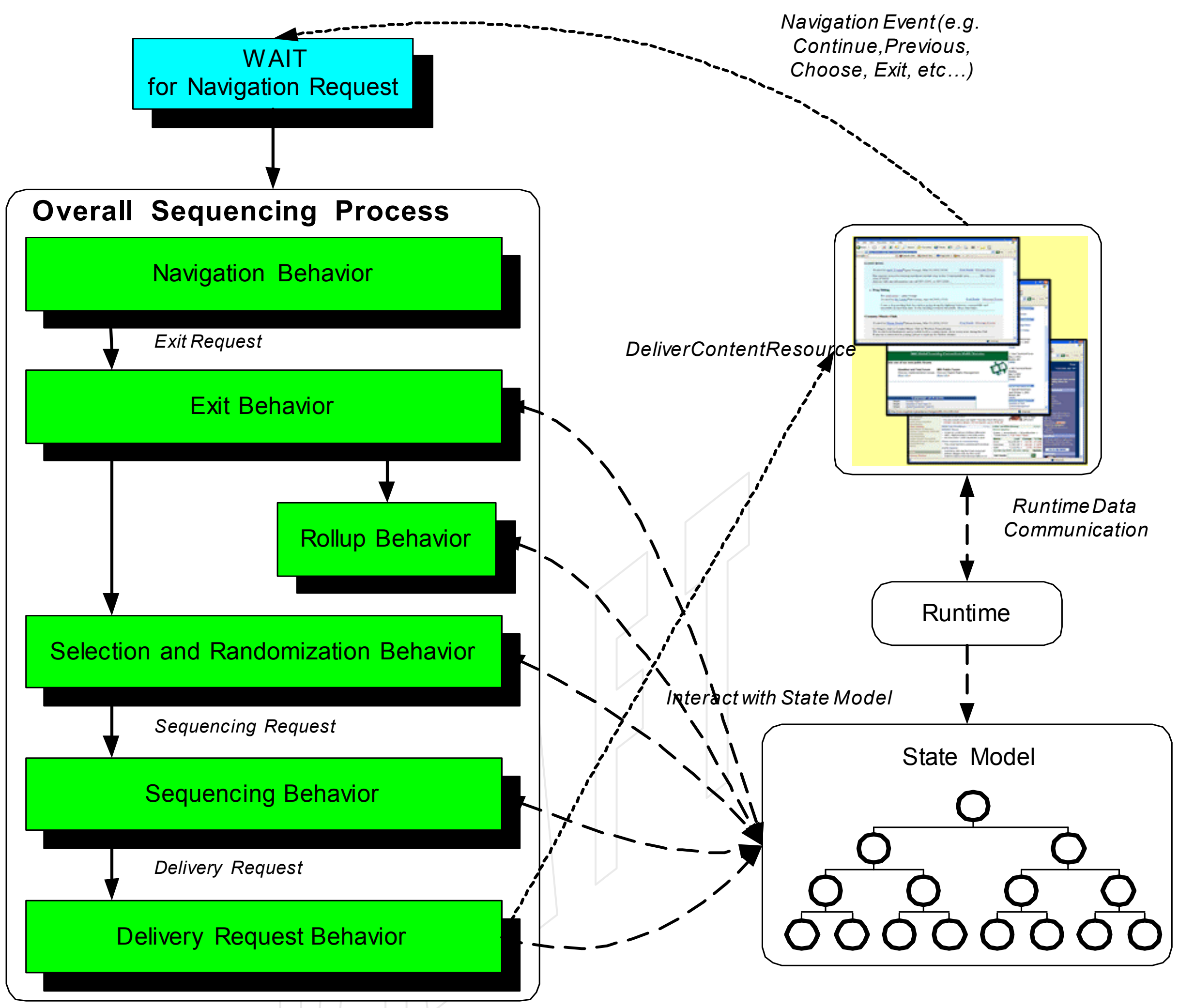
Figure 7.1.1.8a: Current Activity State Model

7.1.2. Overall Sequencing Process

The information in this section is intended to supplement, not replace, the Overall Process Behavior section of the IMS SS Specification. Please refer to the IMS SS Specification for more details.

The Overall Sequencing Process provides the overarching control process for the sequencing engine. It describes how the various sequencing behaviors are applied within the context of a sequencing session. The Overall Sequencing Process encapsulates the following sequencing behaviors:

- Navigation Behavior – Describes how a navigation request (generated by the LMS) is validated and translated into exit and sequencing requests.
- Exit Behavior – Describes how the current attempt on an activity ends, how the state of the activity tree is updated, and how if some action should be performed due to the attempt ending.
- Rollup Behavior – Describes how tracking information for cluster activities is derived from the cluster's children's tracking information.
- Selection and Randomization Behavior – Describes how the activities in a cluster should be considered during processing a sequencing request.
- Sequencing Behavior – Describes how a sequencing request is processed on an activity tree in attempt to identify the 'next' activity to deliver.
- Delivery Behavior – Describes how an activity identified for delivery is validated for delivery, and how LMS should handle delivery of the activity.



Navigation Event (e.g. Continue, Previous, Choose, Exit, etc…)
WAIT for Navigation Request
Overall Sequencing Process
Navigation Behavior
Exit Request
Exit Behavior
Rollup Behavior
Selection and Randomization Behavior
Sequencing Request
Sequencing Behavior
Delivery Request
Delivery Request Behavior
DeliverContentResource
Runtime Data Communication
Runtime
Interact with State Model
State Model

Figure 7.1.2a - Conceptual Model of the Overall Sequencing Process

The Conceptual Model of the Overall Sequencing Process (Figure 7.1.2a) graphically depicts the relation of the various sequencing behaviors to one another and the activity tree (the collection of state models). The entry point for the Overall Sequencing Behavior is a navigation request issued by the LMS. Typically navigation requests will be generated in relation to some learner triggered navigation control (see Navigation Behavior Section 7.1.3). The exit point for the Overall Sequencing Behavior is either the identification of the 'next' activity to deliver, or an interstitial state; the result of the Overall Sequencing Behavior will be something delivered to the learner.

7.1.2.1 Sequencing Loop

The following sequencing loop describes how the various behavior processes interact during sequencing and delivery. The description assumes the activity tree exists and has been initialized.

1. The learner initiates access to the content delivery environment or LMS (e.g., accesses the system, logs in) and establishes a context within a particular unit of instruction (e.g., selects a course or content aggregation).
2. The LMS initiates a sequencing process by issuing a "start" (if it is the first time in the content) or "resume all" (if the content is been experienced before and has been suspended) navigation request.
3. The navigation behavior process translates the "Start" or "Resume All" navigation request into a sequencing request to start sequencing or resume sequencing at a particular node in the activity tree. (This may be the first item in the sequence, that is, the root node in the activity tree, or it may be the node at which the learner suspended activity during a previous session.)
4. Using the information in the tracking model and the sequencing request (e.g., start or activity X, the sequencing behavior traverses the activity tree to locate the appropriate activity (node in the tree) to deliver to the learner.
5. The delivery behavior determines if the selected activity can be delivered, and if so, prepares to deliver the activities content resources to the learner. If the selected activity cannot be delivered, then the overall sequencing process stops and waits for another navigation request.
6. The learner interacts with the learning resource. The sequencing processes are idle and waiting for requests while the learner interacts with the learning resource
7. The activity may report values that update the various tracking model elements during the learner's interactions with the learning resource.
8. The learner, delivery system, or activity invokes a navigation event, such as Continue, Previous, Choose activity X, Abandon, or Exit.
9. The LMS informs the sequencing process of the navigation event by issuing a navigation request.
10. The navigation behavior translates the navigation request into an exit request and a sequencing request. If the navigation request indicates that the learner wants to exit the session, the sequencing process is exited. (Exit behavior and persistence of the activity state model is unspecified and left to the implementation).
11. If the activity triggered the navigation request by terminating or exiting, it may report additional values that update the tracking model. The activity then exits. The rollup process is invoked to determine the effects of any state changes that occurred as a result of delivering the content resource for the activity. The process updates the tracking model for the activity and for any of its parent activities within the activity tree.
12. The process repeats beginning at step 4.

Several variations exist on the order described above, including:

- Selection and randomization of activities may occur prior to the processing of a sequencing request.
- An activity may report state and status information and exit without triggering a specific navigation request.
- Only certain classes of resources and activities may report information to the tracking model during delivery of content and learner interaction with the content.

- The sequencing process may not be able to identify an activity to deliver, or the activity may not be delivered due to failure to meet the defined delivery conditions.

Although the Overall Sequencing Behavior shows how the various sequencing processes are related, implementations are free to invoke the individual sequencing processes at any time, outside of the context of the Overall Sequencing Process. If they do so, they must provide sufficient state management to appear as if they only exhibit the behavior of the Overall Sequencing Process. In addition, implementations are free to invoke the Overall Sequencing Process on ‘dirty’ or tentative data (by managing multiple sets of tracking information), to evaluate “what if” scenarios. Neither the Overall Sequencing Process or any of the sequencing processes ‘know’ that tracking information they are using is the ‘clean’. A common scenario for performing a tentative Overall Sequencing Process evaluation is to provide an ‘intelligent’ user interface. For example, an implementation may provide some navigation control that allows the learner to trigger a ‘continue’ navigation event. If the learner triggers that control, the LMS is free to process a tentative continue navigation request to see what would happen. If the tentative request results in an error, the LMS could notify the learner and throw away the ‘dirty’ state data. If the tentative request succeeds, the LMS could deliver the resulting learning resource and use the ‘dirty’ state data.

The only requirement for a SCORM 1.3 compliant LMS is that it exhibit the behavior described in the Overall Sequencing Process.

7.1.3. Navigation Behavior

Navigation behavior is the primary entry point into the Overall Sequencing Process. It provides the means for learner and system intentions to be communicated to the sequencing engine. The external events that indicate navigation intention are called Navigation Events. The means to trigger these events are called Navigation Controls. The LMS is responsible for processing Navigation Events and invoking the sequencing engine with a corresponding navigation request.

7.1.3.1 Navigation Events

Navigation Events are events external to the LMS that indicate the learner’s or system’s intention to navigate through content in some manner. These events are typically triggered by the learner through user interface controls, however an LMS is free to trigger navigation events. SCORM does not place any restrictions on how navigation events are triggered or processed.

When a navigation event is detected, the LMS may respond in two ways:

1. Ignore the event – The LMS may ignore navigation events that would result in an undesirable system state (experience) for the learner. For example, if the learner is currently experiencing the learning resource associated with the last leaf of the

activity tree, the desire to continue to the next item would result in nothing being delivered; the LMS may ignore a 'continue' navigation event.

2. Issue a navigation request – The LMS may translate the navigation event into a valid navigation request and invoke the Overall Sequencing Process.

7.1.3.2 Navigation Controls

Navigation controls are user interface artifacts that provide the means for a leaner to indicate the desire to navigate away from the *Current Activity* in a particular manner. SCORM does not place any requirements on the LMS or content as to what navigation controls are visible, how they are rendered, how they are triggered, or what navigation events they trigger.

SCORM compliant LMSs or content must provide, at a minimum, navigation controls whose corresponding navigation events would result in valid navigation requests. For example, if there is an activity that is a valid target for user choice (based on the current known state of the activity tree), the LMS must provides some means for the learner to choose that activity. SCORM compliant LMSs and content are not required to render navigation controls whose corresponding events would result in either an invalid navigation request or no content to deliver. For example, if the *Current Activity* is the root of the activity tree, the LMS does not have to provide some means to trigger a previous navigation request. If an LMS or content does render navigation controls whose corresponding events would result in either an invalid navigation request or no content to deliver, the LMS must respond with the appropriate interstitial page.

7.1.3.3 Navigation Requests

The Overall Sequencing process begins when a navigation request is issued to the sequencing engine. Once the navigation request is issued, all behaviors described in the IMS Simple Sequencing Specification process must be applied. A SCORM conformant LMS must exhibit the appropriate behaviors based on the Sequencing Definition information and current tracking information applied to the activities in the activity tree.

SCORM LMSs must accept the following navigation requests (Table 7.1.3.3a).

Navigation Request	**Action**
Start	Issue a *Start* sequencing request.
Resume All	Issue a *Resume All* sequencing request.
Continue	May issue an Exit exit request Issue a *Continue* sequencing request.
Previous	May issue an Exit exit request Issue a *Previous* sequencing request.
Choice	May issue an Exit exit request Issue a *Choice* sequencing request. The request is accompanied by the identification of the target activity.
Exit	Issue an *Exit* exit request. Issue an *Exit* sequencing request. The current attempt on the currently delivered activity is terminated normally; the

	attempt is over. The termination and exit of the activity were not the result of any other external navigation event (e.g., *Continue, Previous, Choice*).
Exit All	Issue an *Exit All* exit request. Issue an *Exit* sequencing request.
Suspend All	Issue a *Suspend All* exit request. Issue an *Exit* sequencing request. The current attempt on the currently delivered activity and all of its parents are terminated normally; the attempts are not over, and the activities are not complete. The activities may be resumed at some time in the future (resumption is not a new attempt). A Simple Sequencing processor must record sufficient state and tracking information so that the activities may be resumed in the future.
Abandon	Issue an *Abandon* exit request. Issue an *Exit* sequencing request. The current attempt on the currently delivered activity is terminated abnormally and the activity is not complete. The activity attempt may not be resumed. There is no rollback of any tracking data.
Abandon All	Issue an *Abandon All* exit request. Issue an *Exit* sequencing request. The current attempt on the currently delivered activity and all of its parents are terminated abnormally and the activities are not complete. The activity attempts may not be resumed. There is no rollback of any tracking data.

Table 7.1.3.3a: SCORM 1.3 Navigation Requests

7.1.3.4 Navigation Behavior Process

The information in this section is intended to supplement, not replace, the Navigation Behavior section of the IMS SS Specification. Please refer to the IMS SS Specification for more details.

The Navigation Behavior Process is triggered by the Overall Sequencing Process. It accepts a navigation request and may return: invalid navigation request, a sequencing request, or an exit request and a sequencing request.

An invalid navigation request is returned when:

- An undefined (not in Table 7.1.3.3a) navigation request is issued.
- A *Start* navigation request is issued, but the sequencing session has already begun.
- A *Resume All* navigation request is issued, but the sequencing session has already begun.
- A *Continue* navigation request is issued and the parent of the *Current Activity* does not have the *Sequencing Control Mode Flow* == true.
- A *Previous* navigation request is issued and the parent of the *Current Activity* does not have the *Sequencing Control Mode Flow* == true, or the parent of the *Current Activity* does not have the *Sequencing Control Mode Forward Only* == false.
- A *Choice* navigation request is issued and:

  - The target of the *Choice* navigation request does not exist in the activity tree. This could be the case if the target activity is not in the tree at all, or if the target activity is not a member of the Available Children of the target's parent.
  - The parent of the target of the *Choice* navigation request is either the root of the activity tree or it has the *Sequencing Control Mode Choice* == true.
  - If processing the *Choice* sequencing request for the target of the *Choice* navigation request would require an active activity (along the 'active path') to be exited and that activity has the *Sequencing Control Mode Choice Exit* == false.

A valid *Continue*, *Previous*, or *Choice* navigation request will result in the corresponding sequencing request. In addition, if a valid *Continue*, *Previous*, or *Choice* navigation request was issued and the *Current Activity* is active, an *Exit* exit request will be issued to end the current attempt on the *Current Activity*.

The *Exit*, *Exit All*, *Suspend, Abandon*, *Abandon All*, navigation requests all result in the corresponding exit request and an *Exit* sequencing request. These navigation requests will cause the *Current Activity* to exit and possibly end the sequencing session.

7.1.4. Exit Behavior

Exit Behavior has two intensions: to end the current attempt on the 'exiting' activity, and to ensure the state of the activity tree is in the most current valid state. Exit Behavior acts on an exit request. It may move the *Current Activity* and may return a sequencing request.

It is important to distinguish between an activity exiting and the activity's associated learning resource exiting. How and when an activity's associated learning resource exits is out of scope of SCORM; SCORM only requires that SCOs end communication (by calling LMSFinish) prior to exiting. An activity exits as part of the internal sequencing behaviors; it is not affected by the exit of any learning resource, however it may (depending on the LMS implementation), require its associated learning resource to exit.

More specifically, the *Current Activity* exits in response to an exit request, if the *Current Activity* is active. The sequencing engine must ensure that the *Current Activity* has exited so that the activity tree is the most current valid state prior to processing any sequencing requests. LMS implementations may require that the learning resource associated with the exiting activity also exit to ensure the most current valid state information is available to the sequencing engine.

7.1.4.1 Exit Requests

Exit requests may be relayed through the Overall Sequencing Process after a valid navigation request has been processed.

In general, an exit request indicates that the attempt on the *Current Activity* must end – the *Current Activity* will become inactive. The IMS SS Specification defines several types of exit requests, each of which results in a different behavior; a SCORM conformant LMS will exhibit these behaviors (Table 7.1.4.1a).

Exit Request	Action
Exit	The current attempt on the *Current Activity* is terminated normally; the attempt is over.
`Exit All`	The current attempts on the active activities (from the root to the *Current Activity,* inclusive) are terminated normally; the attempts are over.
Suspend All	The current attempts on the active activities (from the root to the *Current Activity*, inclusive) are suspended.
Abandon	The current attempt on the Current Activity is terminated abnormally and the activity is not complete. The attempt may not be resumed. There is no rollback of any tracking data.
Abandon All	The current attempts on the active activities (from the root to the *Current Activity*, inclusive) are terminated abnormally and the activities are not complete. Attempts on any activity may not be resumed. There is no rollback of any tracking data.

Table 7.1.4.1a: SCORM 1.3 Exit Requests

7.1.4.2 Evaluating Post Condition and Exit Action Rules

Activities may have one or more post condition an exit action Sequencing Rules associated with them. Sequencing Rules have the structure:

If *condition set* **Then** *action*

The *condition set* defines a set of conditions, which are evaluated against the tracking status information for the activity. If the *condition set* is true, the *action* is applied.

Sequencing rule actions are partitioned into three sets that correspond generally to the timing of their evaluation, which sequencing processes they apply to, and to their affect on those processes. Exit action Sequencing Rules are only evaluated during the Sequencing Exit Action Rule Subprocess of the Exit Behavior. Post Condition Sequencing Rules are only evaluated during the Sequencing Post Condition Rules Subprocess.

For example:

- **If not satisfied Then retry** – Process a Retry sequencing request on the activity, if the activity's objective status is equal to not satisfied.
- **If outside available time range Then exit** – Exit this activity during the Sequencing Exit Action Sequencing Process request on the activity, if the current time is outside of the activity's available time range.
- **If attempted Then exit parent** – Exit the parent of this activity, if the activity has been attempted.
- **If attempted Then exit all –** Exit the activity tree and end the current sequencing session, if the activity's objective status is equal to satisfied.

The examples above represent only a small portion of the types of sequencing rules that may be defined.

7.1.4.3 Exit Behavior Process

The information in this section is intended to supplement, not replace, the Exit Behavior section of the IMS SS Specification. Please refer to the IMS SS Specification for more details.

7.1.4.3.1 Completion Subprocess

The Completion Subprocess is invoked when an activity exits normally. This subprocess ensures that state of the 'exiting' activity is up to date. This subprocess does not change which activity is the *Current Activity*. The following actions occur during the Completion Subprocess:

- The 'exiting' activity becomes inactive.
- If the 'exiting' activity is a leaf, the activity's associated learning resource (SCO) may indicate that the activity exited by setting the SCORM Run-Time Environment Data Model element – cmi.core.exit- to "suspend".
- If the 'exiting' activity is a leaf and its associated learning resource is a SCO, the defined SCORM data mapping occurs immediately after step 1.2 of the Completion Subprocess.
- If the 'exiting' activity is a leaf, its rolled-up objective may be set to satisfied and its completion progress may be set to completed, if the learning resource did not set these values. This is done automatically by the sequencing engine based on the values of the activity's *Delivery Control* values.

- If the activity is a cluster, it is 'suspended' if any of its children are 'suspended'.

7.1.4.3.2 Sequencing Exit Action Rules Subprocess

To Be Supplied.

7.1.4.3.3 Sequencing Post Condition Rule Subprocess

To Be Supplied.

7.1.5. Rollup Behavior

Each activity in the activity tree may track a learner's interactions with that activity. The set of tracking information associated with each activity is defined by the Tracking Model (Section 7.1.1). Leaf activities may provide learning resources that are directly experienced by the learner. Learning resources may communicate status information, which is used to set the corresponding activity's tracking information. When learning resources do not communicate status information (identified by either *Objective Set by Content* == false or *Completion Set by Content* == false), the sequencer will directly set the corresponding activity's tracking information. However, cluster activities cannot provide learning resources and do not directly set their status information. The status of a cluster activity is based on the status of its children; the process of evaluating a cluster's status information is called 'rollup'. Figure 7.1.5a shows a cluster with three children;

this figure will be used throughout this section to illustrate the various rollup processes. The status information of the cluster (green box) is determined, through rollup, from the status information of the cluster's children (yellow boxes).

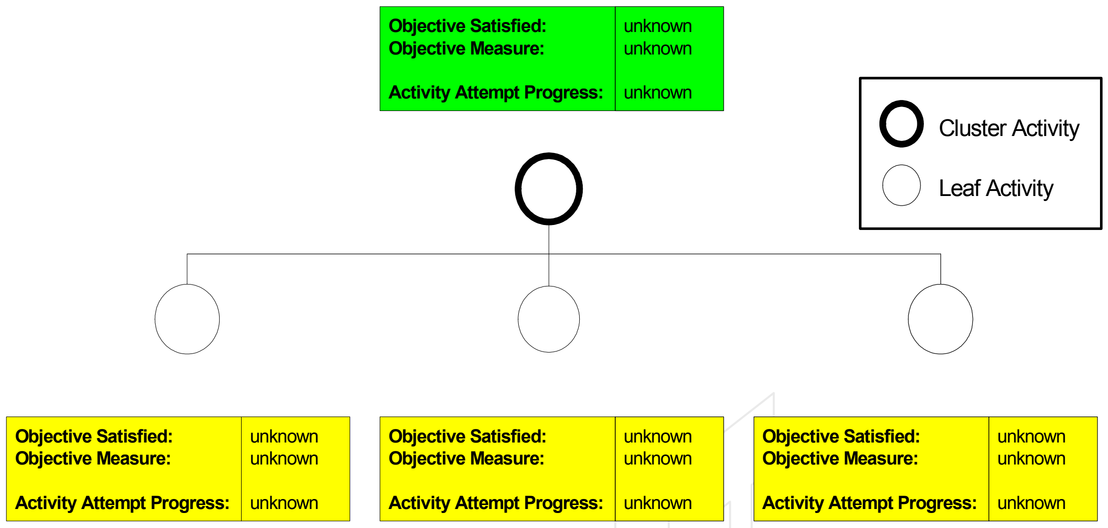

Objective Satisfied: unknown
Objective Measure: unknown
Activity Attempt Progress: unknown
Cluster Activity
Leaf Activity
Objective Satisfied: unknown
Objective Measure: unknown
Activity Attempt Progress: unknown
Objective Satisfied: unknown
Objective Measure: unknown
Activity Attempt Progress: unknown
Objective Satisfied: unknown
Objective Measure: unknown
Activity Attempt Progress: unknown

Figure 7.1.5a: Activity Status Information Used During Rollup

The term 'Rollup' is used throughout this section to mean: "The process of determining a cluster activity's status information based on its children's status."; this is synonymous with "Apply the Overall Rollup Process".

7.1.5.1 Overall Rollup Process

The information in this section is intended to supplement, not replace, the Rollup Behavior section of the IMS SS Specification. Please refer to the IMS SS Specification for more details.

The Overall Rollup Process describes how rollup is applied to the activity tree. It is the controlling process that ensures that all rollup behaviors are applied appropriately. The Overall Rollup Process is only triggered by the sequencer during the Exit Behavior Process, but it may the LMS is free to invoke it any time. If an LMS invokes the Overall Rollup Process at any time other than the Exit Behavior Process, the LMS must ensure that the tracking status states of all activities in the activity tree (and any associated shared global objectives) are consistent with the defined sequencing behavior. One way to ensure that LMS triggered, 'tentative', rollups do not adversely affect the tracking status state of the activity tree is to use a second set of tracking status information, during these rollups.

Within the context of other sequencing information associated with activities and the other sequencing behaviors:

- Rollup only includes *tracked* children.
- Rollup only includes children that contribute to rollup as defined by their *Rollup Controls*.

- Rollup can only be evaluated if all *tracked* and contributing children have know status information.
- Rollup is performed starting at the parent of the leaf activity that triggered rollup (due to a status change) to the root of the activity tree.
- Measure rollup is always performed first, followed by Objective and Activity Progress rollup, in any order.
- The Measure and Objective Rollup Processes only include objective tracking information for each child's unique objective that 'contributes to rollup'.
- The result the Objective Rollup Process only affects the clusters unique objective that 'contributes to rollup'.
- The Overall Rollup Process may be stopped when the status of a cluster activity does not change.
- Rollup rules define how rollup is evaluated for a cluster activity.
- Rollup rules have no effect if defined on a leaf activity – there is nothing to 'rollup'.
- Rollup only affects tracking status values of cluster activities; it does not trigger any sequencing rule evaluations or cause any side-effect actions.

7.1.5.2 Evaluating Rollup Rules

Cluster activities may have one or more Rollup Rules associated with them. Rollup Rules have the structure:

If *child-activity set*, *condition set* **Then** *action*

The *child-activity set* describes the number of *tracked* children activities, which also contribute to rollup. The *condition set* defines a set of conditions, which are evaluated against the tracking status information for each child activity in the *child-activity set*. If the *condition set* is true, the Rollup Rule results in the cluster's tracking status being set according to the *action*.

For example:

- **If any not satisfied Then not satisfied** – The cluster's objective status is set to not satisfied, if any of its tracked, contributing, children has its objective status equal to not satisfied.
- **If 3 satisfied Then satisfied** – The cluster's objective status is set to satisfied, if any three of its tracked, contributing, children has its objective status equal to satisfied.
- **If all satisfied or completed Then completed** – The cluster's activity attempt progress status is set to completed, if all of its tracked, contributing, children have their objective status equal to satisfied or their activity attempt progress status equal to completed.
- **If all satisfied and attempted Then satisfied** – The cluster's objective status is set to satisfied, if all of its tracked, contributing, children have their objective status equal to satisfied and they have been attempted.

- **If 50% attempted Then incomplete** – The cluster's activity attempt progress status is set to incomplete, if 50% or more of its tracked, contributing, children have been attempted.

The examples above represent only a small portion of the types of rollup rules that may be defined.

7.1.5.3 Measure Rollup Process

The Measure Rollup Process sets the cluster's measure to the average weighted measure of the cluster's children. It only includes children that are *tracked* and that *Rollup Objective Satisfied*. The Measure Rollup Process does not affect the cluster's objective or progress status. If any *tracked* child that contributes to rollup does not have a known measure, the measure of the cluster is unknown. An activity's measure may be omitted from during measure rollup by setting its *Rollup Objective Measure Weight* to 0.0. Figure 7.1.5.3a shows an example of the Measure Rollup Process. The information in the light blue boxes comes from the sequencing information associated with the activity.

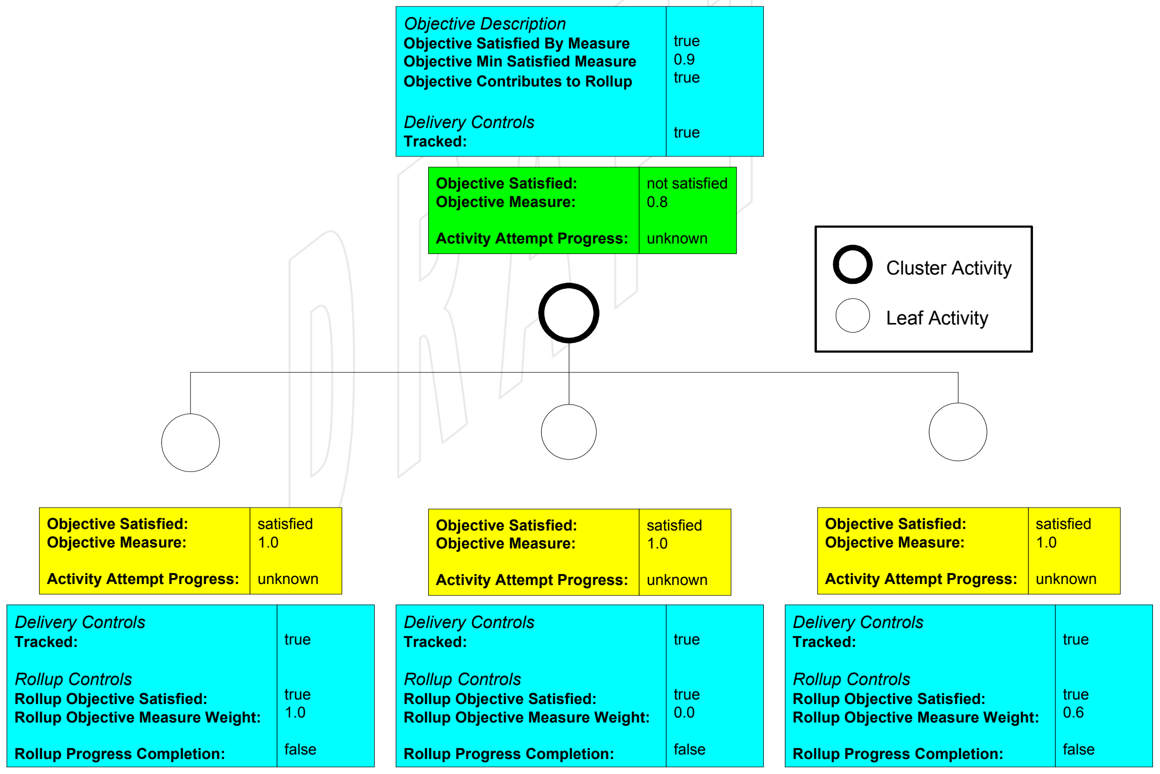


Figure 7.1.5.3a: Example Of the Measure Rollup Process

7.1.5.4 Objective Rollup Process

The Objective Rollup Process sets the cluster's rolled-up objective (the objective with *Objective Contributes to Rollup* == true) status to unknown, satisfied, or not satisfied. It only includes children that are *tracked* and that *Rollup Objective Satisfied*. There are three methods of rolling up objective information. The first method that applies is the only one used to evaluate the objective status of the cluster.

1. Using Measure – If the rolled-up objective has *Objective Satisfied by Measure*, the rolled-up measure is compared against the *Objective Minimum Satisfied Measure*.

    - If the rolled-up measure is unknown, the objective status is unknown.
    - If the rolled-up measure equals or exceeds the *Objective Minimum Satisfied Measure*, the objective status is satisfied.
    - If the rolled-up measure is less than the *Objective Minimum Satisfied Measure*, the objective status is not satisfied.

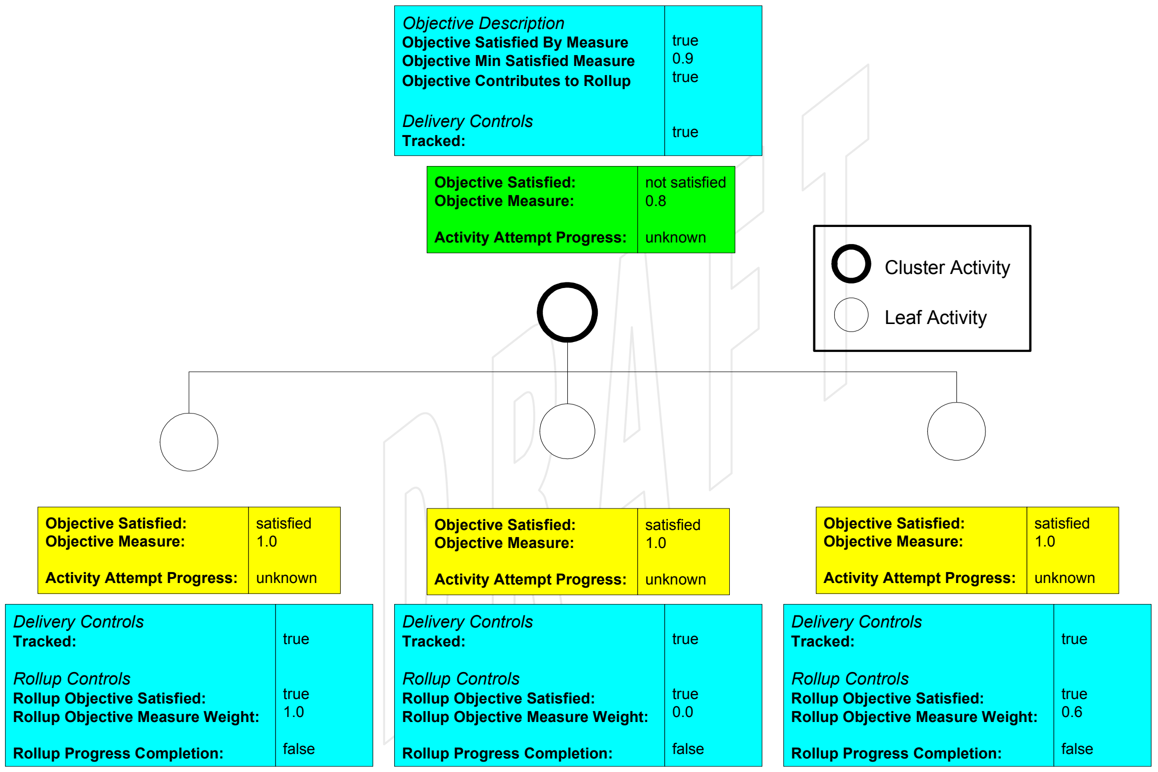


Figure 7.1.5.4a: Objective Rollup Using Measure

2. Using Rules – If any Rollup Rules are defined on the activity that have the actions *satisfied* or *not satisfied*, those rules are evaluated to determine the cluster's objective status. *Not satisfied* rules are evaluated first.

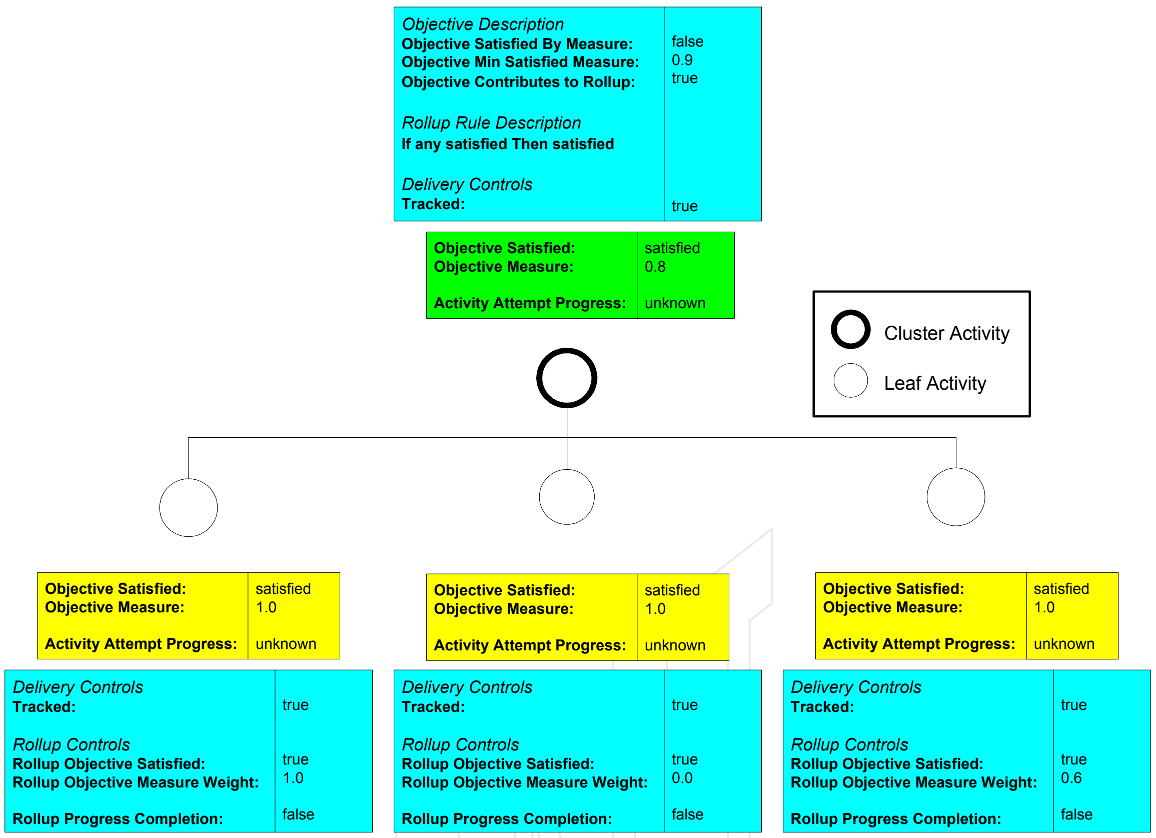

Objective Description
Objective Satisfied By Measure: false
Objective Min Satisfied Measure: 0.9
Objective Contributes to Rollup: true
Rollup Rule Description
If any satisfied Then satisfied
Delivery Controls
Tracked: true
Objective Satisfied: satisfied
Objective Measure: 0.8
Activity Attempt Progress: unknown
Cluster Activity
Leaf Activity
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: unknown
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 1.0
Rollup Progress Completion: false
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: unknown
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.0
Rollup Progress Completion: false
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: unknown
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.6
Rollup Progress Completion: false

Figure7.1.5.4b: Objective Rollup Using Rules

3. Default Rules – If no Rollup Rules are defined on the activity that have the actions *satisfied* or *not satisfied*, the default rollup rule: "If all satisfied Then satisfied", is evaluated.

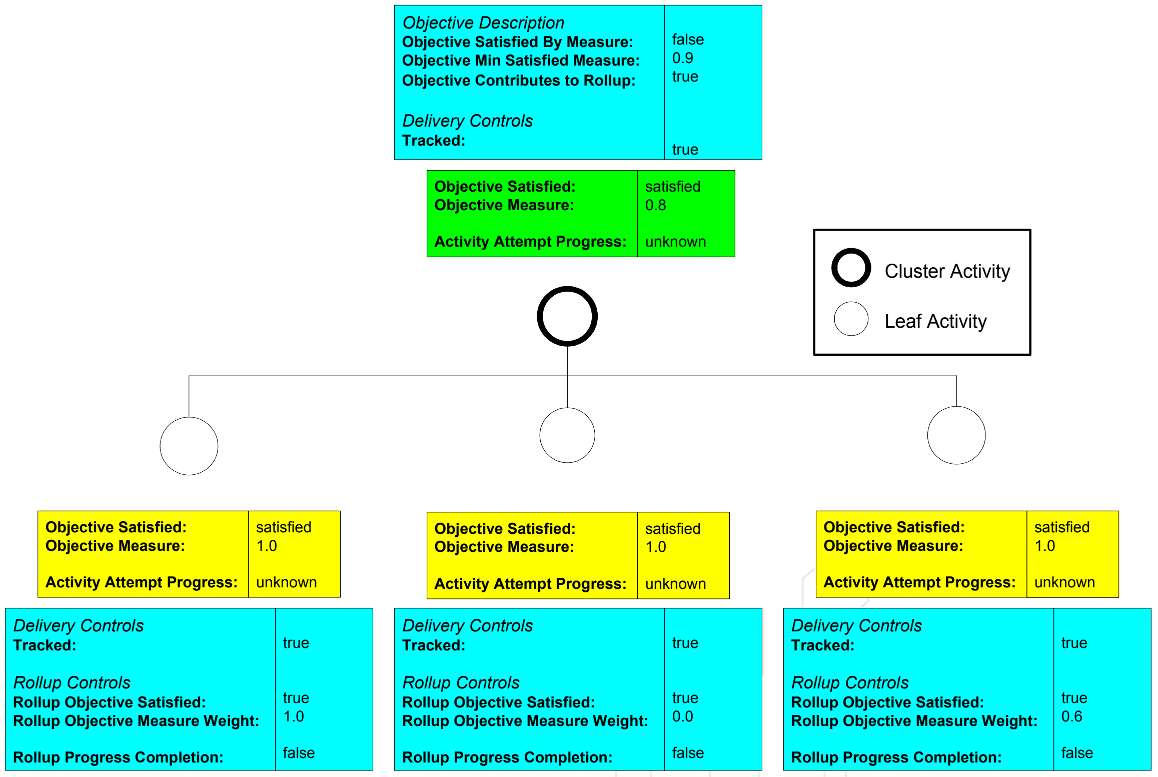
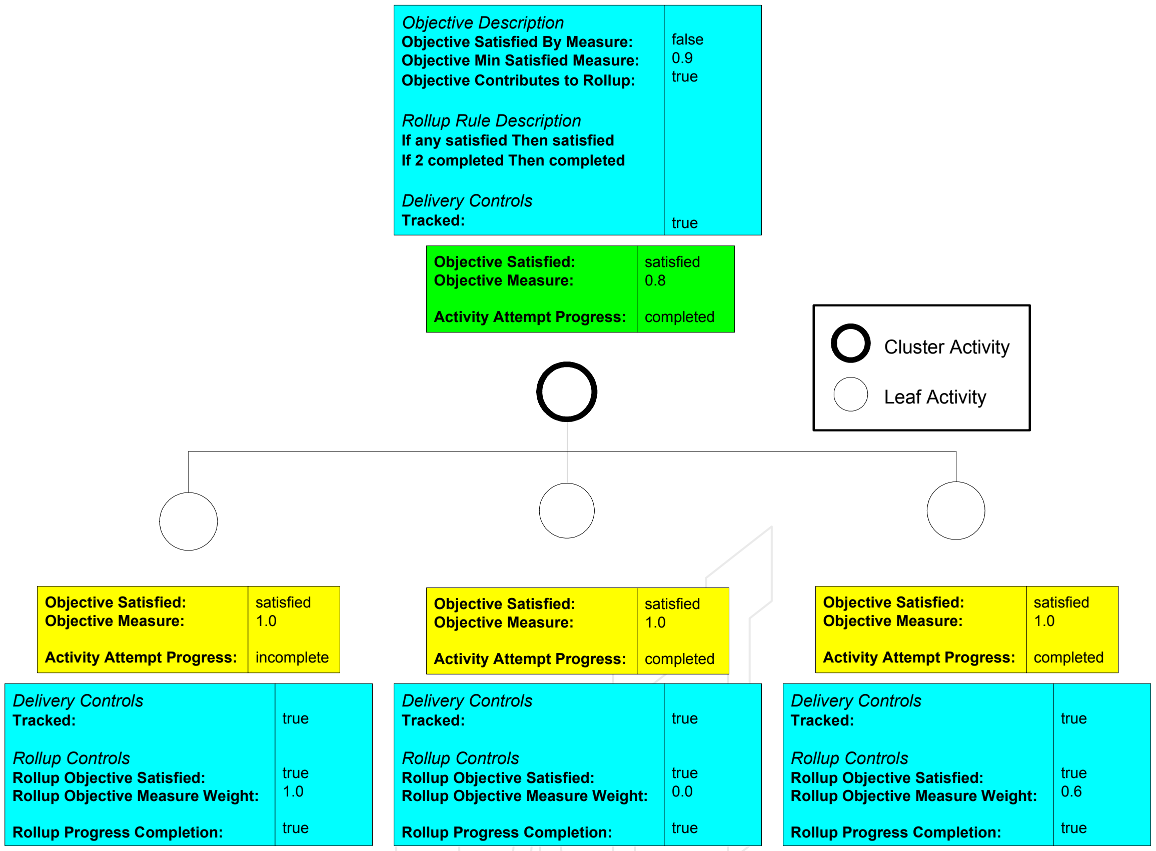

Objective Description
Objective Satisfied By Measure: false
Objective Min Satisfied Measure: 0.9
Objective Contributes to Rollup: true
Delivery Controls
Tracked: true
Objective Satisfied: satisfied
Objective Measure: 0.8
Activity Attempt Progress: unknown
Cluster Activity
Leaf Activity
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: unknown
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 1.0
Rollup Progress Completion: false
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: unknown
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.0
Rollup Progress Completion: false
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: unknown
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.6
Rollup Progress Completion: false

Figure 7.1.5.4c: Objective Rollup Using Default Rule

7.1.5.5 Activity Progress Rollup Process

The Activity Progress Rollup Process sets the cluster's activity attempt progress status to unknown, complete, or incomplete. It only includes children that are *tracked* and that *Rollup Progress Completion*. There are two methods of rolling up progress information. The first method that applies is the only one used to evaluate the progress status of the cluster.

1. Using Rules – If any Rollup Rules are defined on the activity that have the actions *complete* or *incomplete*, those rules are evaluated to determine the cluster's progress status. *Incomplete* rules are evaluated first.



Objective Description
Objective Satisfied By Measure: false
Objective Min Satisfied Measure: 0.9
Objective Contributes to Rollup: true
Rollup Rule Description
If any satisfied Then satisfied
If 2 completed Then completed
Delivery Controls
Tracked: true
Objective Satisfied: satisfied
Objective Measure: 0.8
Activity Attempt Progress: completed
Cluster Activity
Leaf Activity
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: incomplete
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 1.0
Rollup Progress Completion: true
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: completed
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.0
Rollup Progress Completion: true
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: completed
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.6
Rollup Progress Completion: true

Figure 7.1.5.5a: Activity Progress Status Rollup Using Rules

2. Default Rule – If no Rollup Rules are defined on the activity that have the actions *complete* or *incomplete*, the default rollup rule: "If all completed Then completed", is evaluated.

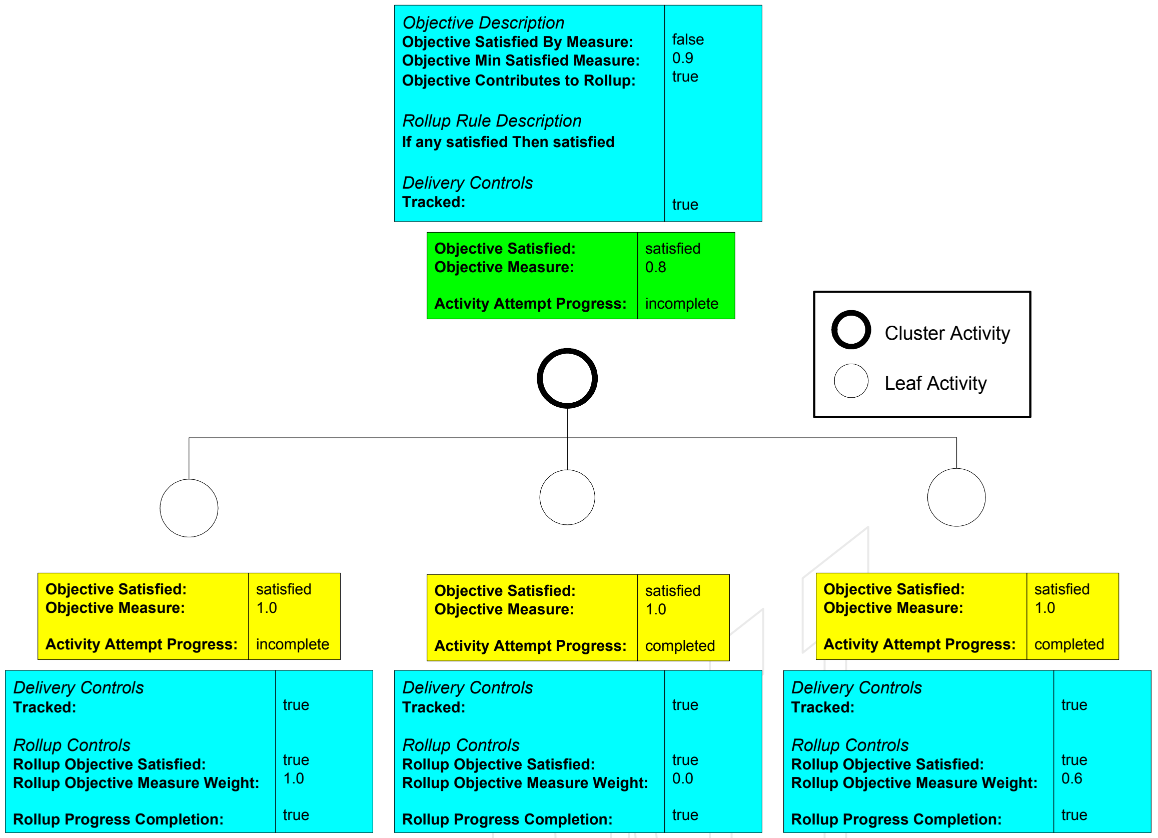

Objective Description
Objective Satisfied By Measure: false
Objective Min Satisfied Measure: 0.9
Objective Contributes to Rollup: true
Rollup Rule Description
If any satisfied Then satisfied
Delivery Controls
Tracked: true
Objective Satisfied: satisfied
Objective Measure: 0.8
Activity Attempt Progress: incomplete
Cluster Activity
Leaf Activity
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: incomplete
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 1.0
Rollup Progress Completion: true
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: completed
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.0
Rollup Progress Completion: true
Objective Satisfied: satisfied
Objective Measure: 1.0
Activity Attempt Progress: completed
Delivery Controls
Tracked: true
Rollup Controls
Rollup Objective Satisfied: true
Rollup Objective Measure Weight: 0.6
Rollup Progress Completion: true

Figure 7.1.5.5b: Activity Progress Rollup Using Default Rule

In addition to setting the cluster's activity attempt progress status, the Activity Progress Rollup Progress will also set the cluster's activity attempt completion amount – to 1.0 , if the cluster is complete; to 0.0, if the cluster is incomplete. The value of activity attempt completion amount is not used by the sequencer: currently, it is only maintained for use by the LMS.

7.1.6. Selection and Randomization Behavior

To Be Supplied

7.1.7. Sequencing Behavior

To Be Supplied

7.1.8. Delivery Behavior

To Be Supplied

SECTION 8
Navigation User Interface Managment

This page intentionally left blank.

8.1. User Interface Management Background

SCORM content is used to provide a learner experience. A Run Time Service (RTS) delivers the content and manages the experience. The RTS may be part of a LMS, or it may be a service –online or offline—that delivers the content on behalf of a LMS.

The RTS manages two types of user interface events that fall into the scope of SCORM 1.3: Some navigation events, and other events that can broadly be categorized as "auxiliary service" events directly related to the SCORM content.

Navigation and auxiliary service events assume the existence of user interface devices to trigger those events. When the learner triggers such a device, the RTS processes the event as a request, processes the request, determines whether the request is valid and what action to take, and then carries out that action. For example, if a package contains an auxiliary resource identified as "Glossary", the RTS would launch the glossary when the user clicks the corresponding button in a toolbar.

The SCORM Version 1.3 imposes no requirements on the type or style of the user interface, including any user interface devices for navigation or auxiliary services. The nature of the user interface and the mechanisms for interactions between the learner and the RTS are intentionally not specified. Issues such as look and feel, presentation style, and placement of user interface devices or controls are outside the scope of the SCORM Version 1.3.

8.2. Navigation Background

In the context of the SCORM Version 1.3, the learning experience provided to the user is supported by an aggregation of learning activities. Learning activities correspond to *Items* in a content packaging manifest and can be nested to any arbitrary depth, based on the desired learning taxonomy, or on the complexity of decomposable activities. Learning activities at the lowest level of nesting may specify an associated learning resource. Each learning resource is represented in content packaging by a *Resource.* A *Resource* typically contains various assets identified by *Files* in the content package manifest. Alternatively, a *Resource* may just reference an asset that is not included in the package, such as a simulation that is delivered by an external service. An activity may specify only one launchable resource to launch in order to experience the activity; however, different activities may specify the same resource.

The SCORM Version 1.3 recognizes two types of learning resources that can be delivered to the learner to perform a learning activity. These are:

- Sharable Content Objects (SCO) that consist of one or more assets packaged as a single launchable resource that does communicate with the LMS via the SCORM Run-Time Environment API.

- Sharable Content Assets (SCA) that consist of one or more assets or services packaged as a single launchable resource that does not communicate with the LMS via the SCORM Run-Time Environment API. These are also referred to as Non-Communicative content objects.

A SCO or SCA must conform to the SCORM Runtime Environment Launch requirements. In other words, a SCO or SCA that was launched by the RTS may not launch another SCO or SCA that will replace it in its own window.

8.2.1. Navigation Overview

In the context of the SCORM, navigation is the process by which a learner and an RTS cooperate to define a specific path through learning activities. To the extent allowed by the author of the aggregation, the learner may trigger user interface devices that indicate a navigation request. These user interface devices may be provided by the RTS, by the content object used in the current activity or by both.

When the learner triggers a navigation request through a navigation user interface device, the RTS processes the request and decides what to do next. This could be one of several things:

- If the effect of the navigation request is to exit the current attempt on the content, the RTS ends the attempt and returns control to the LMS.
- Based on the navigation request and the current tracking status, and after evaluation of any sequencing rule that may apply, the RTS decides on a specific activity, and launches the corresponding SCO or SCA.
- After evaluating the current tracking status and the applicable rules, the RTS decides that the navigation request cannot be honored. In that case, the RTS denies it, and the request is ignored. The RTS takes no sequencing action until the learner triggers another navigation request.

The scope of the SCORM sequencing applies only to navigation between activities. Activities are represented by *Items* in the manifest and may or may not be supported by a launchable resource, which may be a SCO or SCA. At this time, the SCORM does not support the ability to define sequencing or navigation inside a SCO or a SCA that corresponding to a single Activity (*Item*). For example, SCORM navigation does not apply to navigation between individual pages of a multi-page SCO, only to the navigation from one SCO to another SCO.

The IMS Simple Sequencing Specification defines a set of navigation events that can be triggered by a learner through user interface devices. These events can either be triggered by user interface device provided by the LMS or the SCO. How such an event is triggered inside a SCO is not defined by SCORM. For example, whether the triggering event inside the SCO is a user action or the result of an internal scripting event is outside the scope of SCORM.

Navigation events defined by SCORM allow the learner to navigate through a set of learning activities by specifying a desired action, such as to choose a particular learning activity, continue to the next activity, or go back to a previous activity. Additional navigation events allow the learner to exit from a learning activity or to abandon the attempt on a learning activity.

8.2.2. Navigation Events

In the context of the SCORM, all navigation events are processed by an RTS that may be part of a LMS, or that may be a service invoked by a LMS to deliver the content. Based on a navigation event, the RTS determines the appropriate corresponding action and carries out that action. For example, the learner might use a menu or tree user interface control to choose a particular learning activity. The RTS processes the corresponding navigation event, and then, based on the content aggregation, tracking status model and corresponding sequencing rules, launches the content object that is associated with the chosen activity. Likewise, the learner may click a "next" button provided by a SCO. The SCO, must then communicate the appropriate corresponding navigation event to the RTS, so that the RTS can determine, based on the content aggregation and corresponding sequencing rules, which content object to launch next.

The table below defines a set of navigation events and defines how these navigation events map to sequencing requests. In addition, the table defines the source from which each of the navigation events can be triggered. Some events can be triggered by a SCO, while all events can be triggered by the RTS.

The SCORM Version 1.3 imposes no requirements on the type or style of the user interface, including any user interface devices for navigation. The nature of the user interface and the mechanisms for interactions between the learner and the RTS are intentionally not specified. Issues such as look and feel, presentation style, and placement of navigation controls are outside the scope of the SCORM Version 1.3.

In order for a RTS to be conformant, it must manage navigation events and exhibit the behavior defined below in response to each navigation event.

Navigation Event	Behavior Description	Source
Start	This navigation event triggers the *Start* sequencing request. This event is typically generated automatically by the RTS when the learner launches the content aggregation.	RTS Only
Continue	This navigation event triggers the *Continue* sequencing request.	RTS or SCO
Previous	This navigation event triggers the *Previous* sequencing request.	RTS or SCO
Choose	This navigation event triggers the *Choose* sequencing request and is accompanied by the identifier of the chosen activity. If no identifier is provided, or if the identifier is not the identifier of an allowable activity in the current aggregation, the behavior is undefined.	RTS Only
Abandon	This event implies a desire to prematurely or abnormally terminate the	RTS or SCO

	current attempt on the currently delivered activity and corresponding content object or SCO, with no intent to resume at a later time. If the activity is a sub-activity, only this sub-activity is abandoned. Abandon has no effect on currently delivered parent activities. An abandoned attempt is counted as an attempt. In no case does Abandon mean that tracking information that has already been recorded should be "rolled back". For example, if the activity was already recorded as passed or completed, then it does not become failed or incomplete. Any other tracking information recorded for this activity during the current attempt may be maintained or discarded, according to the implementation and policies of the LMS. This event results in an *Exit* sequencing request.	
Abandon All	This event implies a desire to prematurely, abnormally terminate the current attempt on the entire content aggregation. This includes the currently delivered activity, as well as all of its parent activities. The attempt is over and there is no implied intent to resume the current activity. An abandoned attempt is counted as an attempt. In no case does Abandon mean that tracking information that has already been recorded should be "rolled back". For example, if the activity was already recorded as passed or completed, then it does not become failed or incomplete. Any other tracking information recorded for this activity during the current attempt may be maintained or discarded, according to the implementation and policies of the learning management system. This event results in an *ExitAll* sequencing request.	LMS Only
Suspend	This event implies a desire to terminate the current session within the current attempt on the current activity, potentially with the intent to resume at a later time. This event indicates that the learner or the LMS has decided to terminate the currently delivered activity. Normal termination is implied. The activity is being stopped prior to the activity being completed. There is an implied intent to resume the activity at a later time. The LMS must also record state information that is communicated by the resource and/or captured by the LMS for the current activity and any other status information, including rolled up status information, necessary to resume the attempt on the activity at a later time. The LMS must record a value of "suspended" for the cmi.core.exit data model element on behalf of the SCO. This event results in an *Exit* sequencing request.	LMS Only
Suspend All	This event implies a desire to terminate the current session in the current attempt on the entire aggregation. When suspending an attempt on a whole aggregation of activities, the	LMS Only

	status and other control information required to resume the state of the whole aggregation must be stored. In addition, the information required to resume must include which activity, if any was active when the SuspendAll request was made. The LMS must set a value of "suspended" on the cmi.core.exit value for the current SCO. When resuming an attempt on a whole aggregation, that was previously suspended, the activity that was active at the time of suspension must be resumed. If an activity is currently delivered when the SuspendAll event occurs, the behavior for that activity is the same as it would be for a Suspend event for that activity. This event results in an *ExitAll* sequencing request.	
Unqualified Exit	This event implies that the current attempt on the currently delivered activity has finished normally, and that exit of the currently delivered resource was not triggered by another navigation event, such as *Continue*, *Previous*, or *Choose*. This navigation event triggers an Exit sequencing request.	LMS or SCO
Exit All	This navigation event triggers the *ExitAll* sequencing request	LMS Only

Table 5.1.2a: Navigation Events and Descriptions

Navigation events are enabled, and occur in the context of a sequencing control mode. The control mode defined for the parent of a cluster of learning activities defines the navigation events that are able to apply to the children of the cluster. If the choice control mode is enabled, then the choose navigation event can be applied to the child activities in the cluster. Likewise, if the flow control mode is enabled, then both the previous and continue navigation events are allowed for the child activities in the cluster.

8.3. Auxiliary Services Background

In the context of the SCORM Version 1.3, the learning experience provided to the user is supported by an aggregation of learning activities. As described above, learning activities correspond to *Items* in a content packaging manifest and can be nested to any arbitrary depth, based on the desired learning taxonomy, or on the complexity of decomposable activities. Learning activities at the lowest level of nesting specify an associated learning resource. Learning activities, at any level of nesting, can also specify associated auxiliary resources. Such resources are not delivered automatically when the activity is experienced. Auxiliary resources are also not sequenced through navigation events. In other words, only the learning resources defined for leaf nodes in the activity tree are controlled by navigation events. Auxiliary resources are invoked only through Auxiliary Service Events that do not affect navigation.

The SCORM Version 1.3 recognizes two types of learning resources that can be used as an Auxiliary Service Event. These are:

- SCOs that consist of one or more assets packaged as a single launchable resource that does communicate with the LMS via the SCORM Run-Time Environment API.
- SCAs that consist of one or more assets or services packaged as a single launchable resource that does not communicate with the LMS via the SCORM Run-Time Environment API.

Auxiliary services and resources include glossaries, reference manuals, and so on. Unlike the learning resources used for navigation, auxiliary services may be specified for any activity regardless of the level of nesting of an activity.

8.3.1. Auxiliary Services Overview

In the context of the SCORM, auxiliary services are services that are delivered to the learner while the learner is navigating through the learning activities defined in a SCORM package. To the extent specified by the author of the aggregation for each activity, the learner can trigger user interface devices that invoke auxiliary services that are available within the current scope of activity. For example, a glossary specified for the top level activity in an activity hierarchy is available whenever the learner engages in any sub-activity that is included in that top-level activity.

When an auxiliary service comes into scope as the learner navigates through the activity tree, the RTS makes it available but does not launch it automatically. The learner may trigger an auxiliary service event to launch the service, through a corresponding user interface device provided by the RTS. Any SCO that is launched when an auxiliary service is available may also trigger the corresponding auxiliary service event. For example, something in a SCO may ask the RTS to launch a glossary, if one is available.

Whether the learner triggers an auxiliary service event through a user interface device, or the event is triggered by a SCO, the RTS processes the request and decides whether or how to launch the service. The actual appearance or behaviors of auxiliary services and any related user interface devices are outside the scope of SCORM 1.3. In particular, SCORM 1.3 does not define any communication or synchronization for auxiliary services.

8.4. User Interface Device Management

8.4.1. Management of User Interface Devices for Navigation

The RTS must provide the ability to trigger all of the defined navigation events via user interface devices. How those devices are implemented, and in particular how and whether they include a particular prompt or enabled status indicator, is beyond the scope of SCORM 1.3.

As a best practice, the RTS should only enable a user interface device if the corresponding event will be allowed. However, this is not always possible. For example, the Continue user interface device may be enabled on the basis of the current state of the aggregation, but sequencing rules that are evaluated while processing a Continue navigation request may result in denying the request. This could occur if tracking data is not reported by the current SCO until it is unloaded by the RTS as part of processing the sequencing request triggered by the navigation request.

The sequencing control mode should guide the corresponding behaviors of the user interface. For example, a RTS should enable the "choose" navigation user interface device only for those activity clusters that have the "choice" sequencing control mode enabled. Similarly, the RTS should enable the "previous" and "continue" navigation user interface devices only for those activity clusters that have the "flow" sequencing control mode enabled. If the RTS does not selectively enable or disable user interface devices based on the sequencing control mode, then only navigation events and corresponding sequencing requests that are valid for the defined sequencing control modes shall be processed by the RTS. In any case, evaluation of sequencing rules that may take place during the processing of the sequencing request will take precedence, and as a result the sequencing request may be denied. In other words, the RTS must discard or ignore navigation events and corresponding sequencing requests that are not valid based on the defined sequencing control mode of the cluster and the current status of the content aggregation .

The SCO may optionally implement user interface devices for navigation events and auxiliary service events. SCORM specifies how a SCO can query the RTS to determine whether a particular user interface device should be enabled, and how a SCO can signal the triggered event to the RTS.

The content designer can choose to indicate, on an activity-by-activity basis, that the user interface devices provided by default by the RTS should not be available (e.g. visible or enabled) when the content object that corresponds to the activity is launched. One purpose of this indication is to avoid confusing the learner with redundant user interface devices, such as redundant "previous" and "continue" buttons that would appear in the SCO as well as in the user interface provided by the RTS. Other uses of this feature are outside the scope of SCORM 1.3.

8.4.2. Management of User Interface Devices for Auxiliary Services

The RTS must provide the ability to trigger all of the auxiliary service events defined in the content via the RTS user interface.

As a best practice, RTS should enable or show user interface devices only for those auxiliary services that are currently in scope. How those devices are implemented, and in particular how and whether they include a particular prompt or enabled status indicator, is beyond the scope of SCORM 1.3.

At this time a SCO cannot implement user interface devices for auxiliary service events. The SCORM specifies how a SCO can query the RTS to determine whether a particular user interface device for continue, previous and exit events should be enabled, and how a SCO can signal the triggered event to the RTS.

8.4.2.1 Presentation Information Model

The SCORM Version 1.3 Application Profile defines a Presentation model that allows content developers to describe presentation characteristics of a given learning activity. The current driving use case for this presentation model is to allow the content developer a mechanism to indicate whether or not the RTS should provide certain navigation user interface controls. The content developer has the option of defining, on an activity-by-activity basis, whether or not the RTS should provide the user interface controls. The Presentation model can also be used to hold other learning activity presentation characteristics in the future.

Nr	Name	Explanation	Value Space	Data Type	Default
1	Presentation	Information regarding presentation.	-	-	
1.1	NavigationInterface	Characteristics about the user interface controls.	-	-	
1.1.1	HideRTSUI	Indicates that the RTS should not provide user interface devices that enable the learner to trigger specific events.	One or more vocabulary tokens	Open , extensible vocabulary, with specific defined tokens (see next table).	(empty)

Table 8.4.2.1a: SCORM Version 1.3 Presentation Information Model

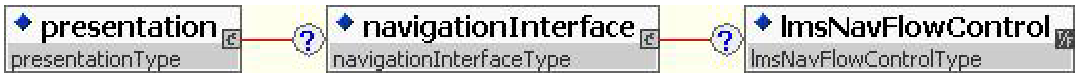
Token	Definition	Explanation
Previous	Previous navigation device	If this token is specified, RTS should not display a “Previous” navigation device when a child of this activity cluster is the current activity.
Continue	Continue navigation device	If this token is specified, RTS should not display a “Continue” navigation device when a child of this activity cluster is the current activity.
Exit	Exit navigation device	If this token is specified, RTS should not display an “Exit SCO” or navigation device when a child of this activity cluster is the current activity.
Abandon	Abandon navigation device	If this token is specified, RTS should not display an "Abandon SCO" navigation device when a child of this activity cluster is the current activity.

Table 8.4.2.1b: SCORM Version 1.3 Runtime user interface device class vocabulary

8.4.2.2 Presentation XML Binding

The XML Binding of the presentation information is handled through an extension to the Content Packaging Manifest XML Schema. A new element called <presentation> has been specified. The <presentation> element contains a single sub-element called <navigationInterface>. The <navigationInterface> element has a single sub-element called <lmsNavFlowControl>.

8.4.2.3 <adlcp:presentation> Element



Description: The <adlcp:presentation> element is a container element that encapsulates presentation information for a given learning activity.

Multiplicity: The <adlcp:presentation> element shall occur 0 or 1 time within the <item> element.

Attribute:

- None

Elements:

- navigationInterface

8.4.2.3.1 <adlcp:navigationInterface> Element

Description: The <adlcp:navigationInterface> element is a container element that encapsulates navigation interface presentation requirements for a given learning activity.

Multiplicity: The <adlcp:navigationInterface> element shall occur 0 or 1 time within the <adlcp:presentation> element.

Attribute:

- None

Elements:

- lmsNavFlowControl

8.4.2.3.1.1 <adlcp:hideRTSUI> Element

Description: The <adlcp:hideRTSUI> element indicates to the that the RTS should not provide user interface devices that enable the learner to trigger specific events. The value of this element is one of the following restricted vocabularies:

- "previous" – If specified, the RTS shall not display a "Previous" navigation device when a child of this activity cluster is the current activity.
- "continue" – If specified, the RTS shall not display a "Continue" navigation device when a child of this activity cluster is the current activity.
- "exit" – If specified, the RTS shall not display a "Exit" navigation device when a child of this activity cluster is the current activity.
- "abandon" – If specified, the RTS shall not display a "Abandon" navigation device when a child of this activity cluster is the current activity.

Multiplicity: The <adlcp:hideRTSUI> element shall occur 0 or More time within the <adlcp:navigationInterface> element.

Attribute:

- None

Elements:

- None

8.4.3. Communication of User Interface status or Events

8.4.3.1 Querying of user interface device status by a SCO

When a SCO includes a user interface device, it desirable for the SCO to know whether it should enable or disable the device depending on whether the RTS determines it should be enabled or disabled. For example, a content designer may choose to develop SCOs in such a way that they display a "Continue" or "Next" button only if there is a next SCO in the sequence. The SCO itself may not make that determination, but the RTS does know and the SCO needs to be able to query it. Similarly, a SCO may need to know under which sequencing control mode it is running, so that it can conditionally render user interface navigation devices.

8.4.3.2 The SCORM User Interface Event Communication Data Model

A SCO may or may not contain user interface devices that allow the learner to trigger a navigation event. Regardless of whether a SCO provides such user interface devices, a SCO can communicate a navigation event to the RTS. For example, a SCO can communicate events such as "Previous", "Continue" and/or "Exit" to the LMS.

Communication takes place using the SCORM API. The information models used in this communication are however not related to the CMI data model used to communicate tracking and learner information. This communication between the SCO and RTS about user interface devices and events use the Navigation Data Model

Nr	Name	Explanation	Value Space	Data Type
1	Navigation	Information regarding navigation.	-	-
1.1	Event	Information regarding the navigation event that the SCO would like to signal to the LMS.	"previous" "continue" "exit"	Vocabulary (Restricted)
1.2	ControlModeEnabled	Information indicating whether or not certain control modes are enabled or not.	-	-
1.2.1	Choice	This element is used to determine if the Choice sequencing control mode is enabled for the cluster containing the SCO.	"true"/"false"	Boolean
1.2.2	Flow	This element is used to determine if the Flow sequencing control mode is enabled for the cluster containing the SCO.	"true"/"false"	Boolean
1.3	EventPermitted	Information indicating whether or not certain navigation events are permitted or not.	-	-
1.3.1	Continue	This element is used to determine if a continue navigation event will result in the	"true"/"false"	Boolean

		delivery of learning resource, or if no learning resource will be delivered as a result of the continue navigation event. **Note:** This element can be used a SCO to determine if it should provided a mechanism such as a user interface navigation control that allows the learner to trigger a continue navigation event.		
1.3.2	Previous	This element is used to determine if a previous navigation event will result in the delivery of learning resource, or if no learning resource will be delivered as a result of the previous navigation event. **Note:** This element can be used a SCO to determine if it should provided a mechanism such as a user interface navigation control that allows the learner to trigger a previous navigation event.	“true”/”false”	Boolean

Table 8.4.3.2a: Navigation Communication Data Model

8.4.3.3 Event Restricted Vocabulary

The values that the Event data model element is comprised of are considered a restricted set. Only values defined in the Data Model shall be used. The following table (Table 8.4.3.3a) describes these values and the intended behaviors.

NavEvent Vocabulary	Behavior
“previous”	When the SCO signals the end of the communication session by calling LMSFinish , the RTS triggers the *Previous* Navigation Event. **Usage:** api.LMSSetValue(“nav.event”, “previous”)
“continue”	When the SCO signals the end of the communication session by calling LMSFinish, the RTS triggers the *Continue* Navigation Event. **Usage:** api.LMSSetValue(“nav.event”, “continue”)
“exit”	When the SCO signals the end of the communication session by calling LMSFinish, the RTS triggers the *Unqualified Exit* Navigation Event. **Usage:** api.LMSSetValue(“nav.event”, “exit”)

Table 8.4.3.3a: Event Data Model Element Fixed Vocabulary

The “previous” and “continue” vocabulary values correspond only to the “Flow” sequencing control mode. Other navigation events are excluded and may not be communicated from the SCO at this time. The LMS must provide the ability to trigger all of the defined Navigation Events.

8.4.3.4 RTS behavior for user interface events communicated by a SCO

When the learner triggers a navigation request event in the RTS user interface, the RTS acts on it immediately. However, when a SCO communicates a navigation request event to the RTS, the RTS buffers this request and does not act on it until the SCO terminates the current communication session by calling LMSFinish. In other words, triggering a communication event by a SCO requires two-step: Communicate the desired event to the RTS, using the API function LMSSetValue, and then trigger the event by calling LMSFinish.

Note: This two-step approach allows the SCO and the RTS to shut down the SCO's communication session in an orderly manner; it also allows the SCO to communicate the request even though LMSFinish does not allow a parameter.

Note: Until it calls LMSFinish, a SCO can set as many navigation request events as it wants, but any previous navigation request is discarded when a new one is set. Only the last recorded navigation request will be acted upon after the RTS finishes processing the LMSFinish from the SCO.

8.4.3.5 Navigation Data Model Binding

The Navigation Communication Data Model is an extension to the SCORM Run-Time Environment communication data model. It is not considered part of the CMI data model and therefore is not bound by using the "cmi" prefix.

Dot-Notation Binding	Details
nav.event	Information regarding the navigation event that the SCO would like to signal to the LMS. LMS Behavior: • **Initialization:** The LMS is not responsible for initializing this value. This data element is used by the SCO to indicate a navigation event. • **Conformance:** This element is mandatory and shall be implemented by an LMS as write-only. SCO Behavior: • **Conformance:** This element is write-only. The SCO is not permitted to invoke an LMSGetValue() request for this data model element. API Requests: • **LMSGetValue():** The LMS shall set the API Error Code to "404" – Element is write only and return "false". • **LMSSetValue():** When a SCO communicates a navigation event to the LMS, the LMS does not process the navigation event immediately. The navigation event is processed upon communication termination – LMSFinish() - of the SCO. Example API Call: • LMSSetValue("nav.event","continue")
nav.control_mode_enabled.choice	This element is used to determine if the Choice sequencing control mode is enabled for the cluster containing the SCO making the request. LMS Behavior: • **Initialization:** The LMS is responsible for initializing this value based on the sequencing rules defined for the parent cluster. If the learning activities parent has the choice mode enabled, the LMS shall initialize this value to "true", otherwise "false". • **Conformance:** This element is mandatory and shall be implemented by an LMS as read-only. SCO Behavior: • **Conformance:** This element is read-only. The SCO is not permitted to invoke an LMSSetValue() request for this data model element. API Requests: • **LMSGetValue():** The LMS shall return the associated Boolean value based on the Control Mode associated with the cluster containing the SCOM making the request. • **LMSSetValue():** The LMS shall set the API Error Code to "403"

	– Element is read only and return "false". The LMS shall not alter the state of the element based on the request. Example API Call: • LMSGetValue("nav.control_mode_enabled.choice")
nav.control_mode_enabled.flow	This element is used to determine if the Flow sequencing control mode is enabled for the cluster containing the SCO making the request. LMS Behavior: • **Initialization:** The LMS is responsible for initializing this value based on the sequencing rules defined for the parent cluster. If the learning activities parent has the flow mode enabled, the LMS shall initialize this value to "true", otherwise "false". • **Conformance:** This element is mandatory and shall be implemented by an LMS as read-only. SCO Behavior: • **Conformance:** This element is read-only. The SCO is not permitted to invoke an LMSSetValue() request for this data model element. API Requests: • **LMSGetValue():** The LMS shall return the associated Boolean value based on the Control Mode associated with the cluster containing the SCO making the request. • **LMSSetValue():** The LMS shall set the API Error Code to "403" – Element is read only and return "false". The LMS shall not alter the state of the element based on the request. Example API Call: • LMSGetValue("nav.control_mode_enabled.flow")
nav.event_permitted.continue	This element is used to determine if a Continue navigation event will result in the delivery of a learning resource, or if no learning resource will be delivered as a result of the Continue navigation event. Note: This element can be used by a SCO to determine if it should provide a mechanism such as a user interface control that allows the learner to trigger a Continue navigation event. LMS Behavior: • **Initialization:** Due to the state of a conceptual "activity tree", no initialization value can be determined. • **Conformance:** This element is mandatory and shall be implemented by an LMS as read-only. SCO Behavior: • **Conformance:** This element is read-only. The SCO is not permitted to invoke an LMSSetValue() request for this data model element. API Requests: • **LMSGetValue():** At the time of the LMSGetValue() request, the LMS shall determine, based on the current delivered SCO, the applicable sequencing behaviors, modes, rules and conditions as defined in the SCORM, if a Continue navigation event will result in the LMS being able to launch a "continue" learning resource. Note that because sequencing behaviors can be based on time-based limit conditions, it is necessary that the LMS re-evaluate the return value of this element each time an LMSGetValue() request is made by the SCO for this element. • **LMSSetValue():** The LMS shall set the API Error Code to "403" – Element is read only and return "false". The LMS shall not alter the state of the element based on the request. Example API Call: • LMSGetValue("nav.event_permitted.continue")
nav.event_permitted.previous	This element is used to determine if a Previous navigation event will result

	in the delivery of a learning resource, or if no learning resource will be delivered as a result of the Previous navigation event. Note: This element can be used by a SCO to determine if it should provide a mechanism such as a user interface control that allows the learner to trigger a Previous navigation event. LMS Behavior: • **Initialization:** Due to the state of a conceptual "activity tree", no initialization value can be determined. • **Conformance:** This element is mandatory and shall be implemented by an LMS as read-only. SCO Behavior: • **Conformance:** This element is read-only. The SCO is not permitted to invoke an LMSSetValue() request for this data model element. API Requests: • **LMSGetValue():** At the time of the LMSGetValue() request, the LMS shall determine, based on the current delivered SCO, the applicable sequencing behaviors, modes, rules and conditions as defined in the SCORM, if a Previous navigation event will result in the LMS being able to launch a "previous" learning resource. Note that because sequencing behaviors can be based on time-based limit conditions, it is necessary that the LMS re-evaluate the return value of this element each time an LMSGetValue() request is made by the SCO for this element. • **LMSSetValue():** The LMS shall set the API Error Code to "403" – Element is read only and return "false". The LMS shall not alter the state of the element based on the request. Example API Call: • LMSGetValue("nav.event_permitted.previous")

Table 5.1.4.3a: Dot-notation Binding of Navigation Communication Data Model

The SCORM Navigation Data Model is only reliable during the attempt on a SCO. For instance, assume the following scenario (Attempt 1):

- SCO sets nav.event element to "continue"
- LMSFinish() is called resulting in termination of the communication between the SCO and the Communications API Adapter

The result of the above is that a "Continue" navigation request is issued.

Later during the learning experience, the user is presented with the same activity again (Attempt 2):

- During this attempt, the SCO does not set the nav.event element
- LMSFinish() is called resulting in termination of the communication between the SCO and the API Adapter

In this case, the nav.event data model element was not persisted from Attempt 1, and does not contain the previously set value of "continue". LMSFinish() in this case would not trigger a navigation request, and the RTS would wait for an event after the activity is terminated.

8.4.4. Termination of a Learning Resource by Navigation

8.4.4.1 Termination of a SCO

The LMSFinish() API Function indicates that the SCO has completed communication with the LMS. Once the LMSFinish() request has been processed, the LMS shall process the last navigation request event communicated by the SCO, if any. Prior to the termination of communications with the LMS, as indicated by a successful call to LMSFinish(), the navigation event communicated to the LMS via the nav.event element has no effect. It is buffered by the LMS until such time as the SCO terminates. In addition to a successful call to LMSFinish(), the attempt on the SCO is also considered to be terminated if the SCO is forcefully terminated prior to the SCO executing a successful call to LMSFinish().

Consider the scenario where a SCO (A) is launched and then calls LMSInitialize(). While SCO A is running, the learner chooses another activity using a navigation user interface control provided by the LMS, for which the corresponding SCO, SCO B, is then launched in the same browser window, thereby forcing the unloading of the previous SCO, SCO A. In this case, SCO A must call LMSFinish() when it detects that it is being forcefully unloaded. The RTS implementation must terminate the communication session it had opened for SCO A, even if SCO A failed to call LMSFinish().

SCO A must implement a handler for the onUnload event that performs any needed communication with the LMS and then calls LMSFinish(). The SCORM may, in the future, introduce a communication mechanism that allows the LMS to notify the SCO prior to forcefully terminating the SCO.

Navigation request events triggered by the RTS always take precedence over the navigation request events communicated by a SCO. For example, while it is possible in the scenario described above that SCO A communicated a navigation request event to the RTS prior to being terminated, the RTS discarded that navigation request as soon as another navigation request was triggered by a learner action in the RTS user interface. In the above scenario, if SCO A communicates a “continue” navigation event to the RTS and then is forced to terminate due to the “choose” navigation event, the “choose” event, because it was triggered by the RTS, always takes precedence.

SECTION 9
The Run-Time Environment Data Model

This page intentionally left blank.

9.1.1. Introduction

With the introduction of the IMS Simple Sequencing Specification within the SCORM, a variety of changes are needed to the SCORM Run-Time Environment Data Model. The changes that are being introduced at this time, have also been suggested to the IEEE CMI Working Group for inclusion in the P1484.11.1 Data Model for Content to Learning Management System Communication. It is envisioned that once the IEEE CMI draft standard is approved more changes will be forthcoming to the SCORM. The following section outlines those changes needed to support the inclusion of the IMS Simple Sequencing Specification, deprecation of elements based on these changes and indications of which elements impact the Tracking Status Model discussed in Section 7.1.1.

9.1.2. SCORM Run-Time Environment Data Model

This section outlines those changes to the SCORM Run-Time Environment Data Model. Some of these changes are to support the inclusion of the IMS Simple Sequencing Specification and other changes are being made for clarification of ambiguities with the data model.

There are some key relationships between certain SCORM Run-Time Environment Data Model elements to the Tracking Status Model for the corresponding activities. The following section also details the relationships between the various SCORM Run-Time Environment Data Model elements and their impacts/effects on the Tracking Status Model. These relationships are important to understand due to the affect of a setting a value has on the conditions defined in the sequencing rules by the content developer.

To increase interoperability, the SCORM Version 1.3 will make all SCORM Run-Time Environment Data Model elements mandatory. The concept of optional data model elements from an LMS perspective has been removed. If a specific element is not included in this section, this indicates that there is no change (other than all elements being mandatory) to the current definition and behavior from that of the SCORM Version 1.2.

9.1.2.1 cmi.core.lesson_status

The cmi.core.lesson_status element has in the past held a dual meaning, success status (pass/fail) and completion status (complete/incomplete/not attempted/browsed). Simple Sequencing requires a distinction between the two meanings, consequently, the cmi.core.lesson_status element is being deprecated and replaced with two new elements:

- cmi.core.success_status
- cmi.core.completion_status

9.1.2.1.1 cmi.core.success_status

The cmi.core.success_status data model element is a status indicator that shall be used by the SCO to indicate mastery or success.

cmi.core.success_status	Indicates whether or not the learner has mastered (passed/failed) the associated SCO. **Data Element Specifics:** • **Data Type:** Vocabulary • **Value Space:** Shall be a member of the following fixed vocabulary: ○ "unknown" – The SCO or LMS cannot assert or has not asserted whether the learner has passed or failed the SCO. ○ "passed" – The SCO's objective(s) is considered passed. ○ "failed" – The SCO's objective(s) is considered failed. **LMS Behavior:** • **Usage:** Normally the SCO will report its own success_status to the LMS: (1) If there is mastery information provided by the content developer (imsss:minNormalizedMeasure), the LMS shall change the status to either passed or failed depending on the learner's score (cmi.core.score.normalized) compared to the mastery score (imsss:minNormalizedMeasure. (2) If there is not mastery information supplied by the content developer, the LMS shall rely on any reported success_status reported by the SCO. • **Initialization:** The LMS is responsible for initializing the success_status to "unknown". • **Conformance:** This element is mandatory and shall be implemented by an LMS as read/write. **SCO Behavior:** • **Conformance:** The element is optionally used by a SCO. The SCO can read/write this data model element if required. **API Requests:** • **LMSGetValue():** The LMS shall return the associated success_status currently maintained by the LMS for the learner. • **LMSSetValue():** The LMS shall set the success_status data model element to the value passed by the SCO If the SCO invokes a request to set the success_status and the value is not of the correct data type, the LMS shall set the API Error Code to "405" – Incorrect Data Type and return "false". The LMS shall not alter the state of the element based on the invalid request. **Example:** LMSGetValue("cmi.core.success_status")

Table 9.1.2.1.1a: cmi.core.success_status Data Model Element Requirements

Tracking Status Model Relationship: The cmi.core.success_status data model element shall be mapped to the *Objective Satisfied Status* element defined in the Tracking Status Model. Due to the nature of the Tracking Status Model, the *Objective Progress* Status element is also affected. The cmi.core.success_status element directly impacts the specific objective of the activity that contributes to rollup. The mapping from the cmi.core.success_status to the *Objective Satisfied Status* and *Objective Progress Status* is dictated in the following table:

cmi.core.success_status	Objective Satisfied Status
"passed"	True
"failed"	False
"unknown"	False

Table 9.1.2.1.1b: Mapping Success Status to Objective Satisfied Status

9.1.2.1.2 cmi.core.completion_status

The cmi.core.completion_status data model element is a status indicator that shall be used by the SCO to indicate completion.

cmi.core.completion_status	Indicates whether or not the learner has completed the associated SCO. **Data Element Specifics:** • **Data Type:** Vocabulary • **Value Space:** Shall be a member of the following fixed vocabulary: o "unknown" – The SCO or LMS cannot assert or has not asserted one of the other three states. o "completed" – The learner has experienced enough of the SCO for the content developer to considered it completed. o "incomplete" – The SCO was begun but not finished. o "not attempted" – The learner has not accessed the SCO, or the learner previously has accessed the SCO but has experienced so little of it that the content developer considers it to be not attempted. **LMS Behavior:** • **Usage:** The SCO should report its own completion_status to the LMS. The LMS does not have a mechanism to be able to determine completion status. • **Initialization:** The LMS is responsible for initializing the completion_status to "unknown". • **Conformance:** This element is mandatory and shall be implemented by an LMS as read/write. **SCO Behavior:** • **Conformance:** The element is optionally used by a SCO. The SCO can read/write this data model element if needed. **API Requests:** • **LMSGetValue():** The LMS shall return the associated completion_status currently maintained by the LMS for the learner. • **LMSSetValue():** The LMS shall set the completion_status data model element to the value passed by the SCO. If the SCO invokes a request to set the resume_location and the value is not of the correct data type, the LMS shall set the API Error Code to "405" – Incorrect Data Type and return "false". The LMS shall not alter the state of the element based on the invalid request. **Example:** LMSGetValue("cmi.core.completion_status")

Table 9.1.2.1.2a: cmi.core.completion_status Data Model Element Requirements

Tracking Status Model Relationship: The cmi.core.completion_status data model element shall be mapped to the *Activity Attempt Completion Status* element defined in the Tracking Status Model. Due to the nature of the Tracking Status Model, the *Activity Attempt Progress Status* element is also affected. The mapping from the cmi.core.completion_status to the *Activity Attempt Completion Status* and *Activity Attempt Progress Status* is dictated in the following table:

cmi.core.completion_status	Activity Attempt Completion Status
"not attempted"	False
"passed"	True
"failed"	False
"unknown"	False

Table 9.1.2.1.2b: Mapping Completion Status to Activity Attempt Completion Status

9.1.2.2 cmi.core.entry

The cmi.core.entry SCORM Run-Time Data Model indicates whether or not the learner has been in the SCO before.

cmi.core.entry

Supported API calls
LMSGetValue()

Data Type:
CMIVocabulary (Entry)

"ab-initio"

"resume"

"" - *empty string*

SCO Accessibility:
Read Only

Definition: Indication of whether the student has been in the SCO before.

Usage: When a student enters the SCO for the first time the cmi.core.entry element should be set to ab-initio by the LMS. If the student re-enters a suspended SCO then the entry flag should be set to resume by the LMS.

Format: A set vocabulary phrase. Three possible vocabulary values:

- **"ab-initio":** This indicates it is the first time the student is entering the SCO. Because the student may have passed all of the objectives in a SCO by completing a pre-test, the completion_status of not attempted is not a reliable indicator. That is, a SCO may be passed without the student having ever seen it.
- **"resume":** This indicates that the student was in the SCO earlier. The student is resuming a suspended SCO.
- **"":** The empty string should be used to represent an entry into the SCO that is neither an initial (ab-initio) nor a continuation from a suspended state (resume). A scenario that this might be used is if the SCO was already completed and then later it was loaded for review purposes. In this case it was neither an initial launch (ab-initio) nor a continuation from a suspended state (resume).

LMS Behavior:

- **Initialization:** Upon initial launch of the SCO, the LMS should initialize the data model value to "ab-inito".
  - Additional Behavior: Upon receiving an LMSFinish() or the user navigates away, the LMS should set the cmi.core.entry to either “” – (empty) or “resume”. This is determined by the LMS looking at the value that the SCO had set for the cmi.core.exit. If the SCO set the value of cmi.core.exit to “suspend”, then the LMS will set cmi.core.entry to “resume” upon the next launch of the SCO. If the SCO set cmi.core.exit to a value other than “resume” or did not set the value at all, the LMS will set cmi.core.entry to “” (empty).
- **LMSGetValue():** Returns the value stored in the data model. The return must be one of the set vocabularies for the cmi.core.entry data element
  - **Example Return Values:**
    "ab-initio"
    "resume"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **403 - Element is read only.** If a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 204.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported..

SCO Usage Example:

```
var entryStatus = LMSGetValue("cmi.core.entry")
if (LMSGetLastError() == "0")
{
   if (entryStatus == "resume")
```

	{
	  // Student is resuming SCO
	}
	else
	{
	  // This is the first time the student has entered the SCO
	}
	}
	else
	{
	  // Error condition, handle appropriately
	}

Table 9.1.2.2a: cmi.core.entry Data Model Element Requirements

9.1.2.3 cmi.core.exit

The cmi.core.exit SCORM Run-Time Data Model indicates how or why the learner exited the SCO.

cmi.core.exit	
Supported API calls: LMSSetValue() **Data Type:** CMIVocabulary (Exit) **"time-out"** **"suspend"** **"logout"** **""** - ***empty string*** **SCO Accessibility:** Write Only	**Definition:** An indication of how or why the student left the SCO. **Usage:** Used to indicate the reason that the SCO was last exited. **Format:** A set vocabulary phrase. Three possible vocabulary values: • **"time-out":** This indicates the SCO ended because the SCO has determined an excessive amount of time has elapsed, or the max_time_allowed has been exceeded. The max_time_allowed can be found in the manifest (imsss:attemptExperiencedDruationLimit for the item) • **"suspend":** This indicates the student leaves the SCO with the intent of returning to it later at the point where he/she left. • **"logout":** This indicates that the student logged out from within the SCO instead of returning to the LMS system to log out. This implies that the SCO passed control to the LMS system, and the LMS system automatically logged the student out of the course -- after updating the appropriate data model elements. • **"" :** The empty string vocabulary should be used to represent a normal exit state. **LMS Behavior:** ▪ **Initialization:** Element does not need initialized. There is never a LMSGetValue() done on this element. Element is controlled by the SCO. o **Additional behavior:** A SCO has the option of setting the cmi.core.exit to one of four values. Based on the value, the LMS behavior should be as follows: ▪ If the SCO set the cmi.core.exit to "time-out" the LMS should set cmi.core.entry to "" (empty) upon the next launching of the SCO. ▪ If the SCO set the cmi.core.exit to "suspend" the LMS should set the cmi.core.entry to "resume" upon the next launching of the SCO. ▪ If the SCO set the cmi.core.exit to "logout" the LMS should set the cmi.core.entry to "" (empty) upon the next launching of the SCO. In addition, the LMS should issue a *Exit All* Navigation Request that overrides any user initiated Navigation Event. ▪ If the SCO set the cmi.core.exit to "" (empty) the LMS should set the cmi.core.entry to "" (empty) upon the next launching of the SCO. ▪ If the SCO did not set the cmi.core.exit to any value that LMS should set cmi.core.entry to "" (empty) upon

	the next launching of the SCO. ▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string (""). o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string (""). ▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element. o **Example API call:** LMSSetValue("cmi.core.exit","logout") o **Example Set Values:** "time-out" "suspend" "logout" o **Error Code:** ▪ **405 – Incorrect Data Type:** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.* **SCO Usage Example:** LMSSetValue("cmi.core.exit","time-out")

Table 9.1.2.3a: cmi.core.exit Data Model Element Requirements

<u>Activity State Information Model Relationship</u>: The cmi.core.exit data model element shall be mapped to the *Activity is Suspended* element defined in the Activity State Information Model. If a SCO sets the cmi.core.exit data model element to "suspend", the LMS shall set the *Activity is Suspended* flag to true. If the SCO sets the cmi.core.exit data model element to "time-out" or "logout", the LMS shall set the *Activity is Suspended* flag to false. *Note: See additional LMS behavior for "logout" in the table above.*

9.1.2.4 cmi.core.score.normalized

The SCORM Version 1.3 will introduce the cmi.core.score.normalized element. This element represents a normalized score between -1..1 (inclusive).

cmi.core.score.normalized	
Supported API calls: LMSGetValue() LMSSetValue() **Data Type:** Normalized real number between –1 and 1. (inclusive) **SCO Accessibility:** Read / Write	**Definition:** Indication of the measure of performance of the student during his last attempt on the SCO. This score may be determined and calculated in any manner that makes sense to the SCO designer as long as the result is normalized. The cmi.core.score.normalized must be a value between -1 and 1 (inclusive). **Usage:** When the student is in their first attempt at a SCO the cmi.core.score.normalized should be set to "" (empty string). For additional attempts the cmi.core.score.normalized reflects what measure was recorded on the student's last previous attempt. If no cmi.core.score.normalized was set in a SCO and a request for the cmi.core.score.normalized was made by the SCO, then an empty string should be returned (""). **Format:** Decimal number between –1 and 1 (inclusive) or an empty string.

LMS Behavior:

- **Initialization:** LMS should initialize this to an empty string ("") upon initial launch of a SCO. The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (“”)
- **LMSGetValue():** Returns the value stored in the data model. The value returned must be a real number between –1 and 1.
    - **Example API call:** LMSGetValue("cmi.core.score.normalized")
    - **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
    - **Example API call:** LMSSetValue("cmi.core.score.normalized","0.95")
    - **Error Code:**
        - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  “0.90"
  "-0.85"
  ""

SCO Usage Example:
The SCO could use this to keep track of the normalized score of the student in the SCO.

LMSSetValue("cmi.core.score.normalized","0.85");

Table 9.1.2.4a: cmi.core.score.normalized Data Model Element Requirements

Tracking Status Model Relationship: The cmi.core.score.normalized data model element shall be mapped to the *Objective Normalized Measure* element defined in the Tracking Status Model. Due to the nature of the Tracking Status Model, the *Objective Measure Status* element is also affected. The cmi.core.score.normalized element directly impacts the measure of the specific objective of the activity that contributes to rollup. The mapping from the cmi.core.score.normalized to the *Objective Normalized Measure* and *Objective Measure Status* is dictated in the following table:

cmi.core.score.normalized Value	Objective Measure Status	Objective Normalized Measure
“”	False	False
Score between –1 and 1	True	Score between –1 and 1

Table 9.1.2.4b: Mapping Normalize Score to Objective Normalized Measure

9.1.2.5 Raw (cmi.core.score.raw)

The *cmi.core.score.raw* SCORM Run-Time Environment Data Model element indicates the measure of performance of the learner during his last attempt on the SCO. Due to the introduction of the IMS Simple Sequencing specification, the SCORM is changing the accepted values for this data model element.

This is changed from the SCORM Version 1.2. The SCORM version 1.2 required this element to be normalized between 0 and 100. This restriction is lifted due to the introduction of the cmi.core.score.normalized element. Content developers can now use the raw score in any way that that suits their specific use case.

9.1.2.6 Max (cmi.core.score.max)

The *cmi.core.score.max* SCORM Run-Time Environment Data Model element indicates maximum score that the learner could have achieved. Due to the introduction of the IMS Simple Sequencing specification, the SCORM is changing the accepted values for this data model element.

This is changed from the SCORM Version 1.2. The SCORM version 1.2 required this element to be normalized between 0 and 100. This restriction is lifted due to the introduction of the cmi.core.score.normalized element. Content developers can now use the max score in any way that that suits their specific needs.

9.1.2.7 Min (cmi.core.min)

The *cmi.core.score.min* SCORM Run-Time Environment Data Model element indicates minimum score that the learner could have achieved. Due to the introduction of the IMS Simple Sequencing specification, the SCORM is changing the accepted values for this data model element.

This is changed from the SCORM Version 1.2. The SCORM version 1.2 required this element to be normalized between 0 and 100. This restriction is lifted due to the introduction of the cmi.core.score.normalized element. Content developers can now use the min score in any way that that suits their specific needs.

9.1.2.8 SessionTime (cmi.core.session_time)

To Be Supplied

9.1.2.9 cmi.objectives.n.score.normalized

The SCORM Version 1.3 will introduce the cmi.objectives.n.score.normalized element. This element represents a normalized score between -1..1 (inclusive).

cmi.objectives.n.score.normalized	
Supported API calls: LMSGetValue() LMSSetValue() **Data Type:** Normalized real number between –1 and 1. (inclusive) **SCO Accessibility:** Read / Write	**Definition:** Numerical representation of student measure after each attempt on the objective. The cmi.objectives.n.score.normalized must be a normalized value between -1 and 1 (inclusive). **Usage:** Normal score after each attempt for an objective. **Format:** Real number between –1 and 1 (inclusive). **LMS Behavior:** ▪ **Initialization:** Element should be initialized to an empty string (""). The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string

	(“”).
	▪ **LMSGetValue():** The LMS should return the objectives normalized score identified by the request. The returned value must match the associated data type.
	o **Example API call:** LMSGetValue("cmi.objectives.0.score.normailized")
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
	o **Example API call:** LMSSetValue("cmi.objectives.0.score.normailized","0.5")
	o **Error Code:**
	▪ **405 – Incorrect Data Type** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	▪ **Example Return/Set Values:**
	"0.96"
	"-0.5"
	""
	
	SCO Usage Example:
	SCO could use this to set a normalized score associated with an objective.
	
	var objScoreChildren = LMSGetValue("cmi.objectives.0.score")
	if (objScoreChildren.indexOf(“normalized”) != -1)
	{
	   LMSSetValue("cmi.objectives.0.score.normalized","0.85")
	}

Table 9.1.2.9a: cmi.objectives.n.score.normalized Data Model Element Requirements

__**Tracking Status Model Relationship**__: The cmi.objectives.n.score.normalized data model element shall be mapped to the *Objective Normalized Measure* element defined in the Tracking Status Model. Due to the nature of the Tracking Status Model, the *Objective Measure Status* element is also affected. The cmi.objectives.n.score.normalized element directly impacts the measure of the specific objective of the activity that has a matching identifier to the cmi.objectives.n.id. The mapping from the cmi.core.score.normalized to the *Objective Normalized Measure* and *Objective Measure Status* is dictated in the following table:

cmi.objectives.n.score.normalized Value	Objective Measure Status	Objective Normalized Measure
“”	False	False
Score between –1 and 1	True	Score between –1 and 1

Table 9.1.2.9b: Determining Status Base on Normalized Score

The following pseudo code illustrates the process for mapping an objectives normalized score to the appropriate *Objective Normalized Measure* and *Objective Measure Status*.

```
Loop over the objective array cmi.objectives._count times
   Read the cmi.objectives.<loopIndex>.id.
   If the activity has an objective with a corresponding ID
         Set the associated Objective Measure Status.
         Set the associated Objective Normalized Measure.
```

```
      Else
            Ignore the SCO set normalized score.
      End If
   End Loop
```

9.1.2.10 cmi.objectives.n.score.raw

The *cmi.objectives.n.score.raw* SCORM Run-Time Environment Data Model element indicates the measure of performance of the learner during his last attempt on the objective. Due to the introduction of the IMS Simple Sequencing specification, the SCORM is changing the accepted values for this data model element.

This is changed from the SCORM Version 1.2. The SCORM version 1.2 required this element to be normalized between 0 and 100. This restriction is lifted due to the introduction of the cmi.objectives.n.score.normalized element. Content developers can now use the raw score in any way that that suits their specific needs.

9.1.2.11 cmi.objectives.n.score.max

The *cmi.objectives.n.score.max* SCORM Run-Time Environment Data Model element indicates maximum score that the learner could have achieved. Due to the introduction of the IMS Simple Sequencing specification, the SCORM is changing the accepted values for this data model element.

This is changed from the SCORM Version 1.2. The SCORM version 1.2 required this element to be normalized between 0 and 100. This restriction is lifted due to the introduction of the cmi.objectives.n.score.normalized element. Content developers can now use the max score in any way that that suits their specific needs.

9.1.2.12 cmi.objectives.n.score.min

The *cmi.objectives.n.score.min* SCORM Run-Time Environment Data Model element indicates minimum score that the learner could have achieved. Due to the introduction of the IMS Simple Sequencing specification, the SCORM is changing the accepted values for this data model element.

This is changed from the SCORM Version 1.2. The SCORM version 1.2 required this element to be normalized between 0 and 100. This restriction is lifted due to the introduction of the cmi.objectives.n.score.normalized element. Content developers can now use the min score in any way that that suits their specific needs.

9.1.2.13 cmi.objectives.n.status

The cmi.core.lesson_status element has in the past held a dual meaning, success status (pass/fail) and completion status (complete/incomplete/not attempted/browsed). Simple Sequencing requires a distinction between the two meanings, consequently, the cmi.core.lesson_status element is being deprecated and replaced with two new elements:

- cmi.core.success_status
- cmi.core.completion_status

9.1.2.13.1 cmi.objectives.n.success_status

The cmi.objectives.n.success_status data model element is a status indicator that shall be used by the SCO to indicate mastery or success or an objective.

cmi.objectives.n..success_status	Indicates whether or not the learner has mastered (passed/failed) the associated objective. **Data Element Specifics:** • **Data Type:** Vocabulary • **Value Space:** Shall be a member of the following fixed vocabulary: o "unknown" – The SCO or LMS cannot assert or has not asserted whether the learner has passed or failed the objective. o "passed" – The objective(s) is considered passed. o "failed" – The objective(s) is considered failed. **LMS Behavior:** • **Usage:** Normally the SCO will report its own objectives' success_status to the LMS: (3) If there is mastery information provided by the content developer (imsss:minNormalizedMeasure), the LMS shall change the status to either passed or failed depending on the learner's score (cmi.objectives.n.score.normalized) compared to the mastery score (imsss:minNormalizedMeasure). (4) If there is not mastery information supplied by the content developer, the LMS shall rely on any reported success_status reported by the SCO. • **Initialization:** The LMS is responsible for initializing the success_status to "unknown". • **Conformance:** This element is mandatory and shall be implemented by an LMS as read/write. **SCO Behavior:** • **Conformance:** The element is optionally used by a SCO. The SCO can read/write this data model element if required. **API Requests:** • **LMSGetValue():** The LMS shall return the associated success_status currently maintained by the LMS for the learner. • **LMSSetValue():** The LMS shall set the success_status data model element to the value passed by the SCO If the SCO invokes a request to set the success_status and the value is not of the correct data type, the LMS shall set the API Error Code to "405" – Incorrect Data Type and return "false". The LMS shall not alter the state of the element based on the invalid request. **Example:** LMSGetValue("cmi.objectives.0.success_status")

Table 9.1.2.13.1a: cmi.objectives.n.success_status Data Model Element Requirements

Tracking Status Model Relationship: The cmi.objectives.n.success_status data model element shall be mapped to the *Objective Satisfied Status* element defined in the Tracking Status Model. Due to the nature of the Tracking Status Model, the *Objective Progress* Status element is also affected. The cmi.objectives.n.success_status element directly impacts the specific objective of the activity that has a matching identifier to the cmi.objectives.n.id. The mapping from the cmi.objectives.n.success_status to the *Objective Satisfied Status* and *Objective Progress Status* is dictated in the following table:

cmi.objectives.n.success_status value	Objective Satisfied Status
"passed"	True
"failed"	False
"unknown"	False

Table 9.1.2.13.1b: Mapping cmi.objectives.n.success_status to Objective Satisfied Status

The following pseudo code illustrates the process for mapping an objectives normalized score to the appropriate *Objective Normalized Measure* and *Objective Measure Status*.

```
Loop over the objective array cmi.objectives._count times
   Read the cmi.objectives.<loopIndex>.id.
   If the activity has an objective with a corresponding ID
         Set the associated Objective Progress Status.
         Set the associated Objective Satisfied Status.
   Else
         Ignore the SCO set success status.
   End If
End Loop
```

9.1.2.13.2 cmi.objectives.n.completion_status

The cmi.objectives.n.completion_status data model element is a status indicator that shall be used by the SCO to indicate an objectives completion.

cmi.objectives.n.completion_status	Indicates whether or not the learner has completed the associated objective. **Data Element Specifics:** • **Data Type:** Vocabulary • **Value Space:** Shall be a member of the following fixed vocabulary: o "unknown" – The SCO or LMS cannot assert or has not asserted one of the other three states. o "completed" – The learner has experienced enough of the objective for the content developer to considered it completed. o "incomplete" – The objective was begun but not finished. o "not attempted" – The learner has not accessed the objective, or the learner previously has accessed the objective but has experienced so little of it that the content developer considers it to be not attempted. **LMS Behavior:** • **Usage:** The SCO should report its objectives' completion_status to the LMS. The LMS does not have a mechanism to be able to determine completion status. • **Initialization:** The LMS is responsible for initializing the completion_status to "unknown". • **Conformance:** This element is mandatory and shall be implemented by an LMS as read/write. **SCO Behavior:** • **Conformance:** The element is optionally used by a SCO. The SCO can read/write this data model element if needed. **API Requests:** • **LMSGetValue():** The LMS shall return the associated completion_status currently maintained by the LMS for the learner. • **LMSSetValue():** The LMS shall set the completion_status data model element to the value passed by the SCO. If the SCO invokes a request to set the resume_location and the value is not of the correct data type, the LMS shall set the API Error Code to "405" – Incorrect Data Type and return "false". The LMS shall not alter the state of the element based on the invalid request. **Example:** LMSGetValue("cmi.objectives.n.completion_status")

Table 9.1.2.13.2a: cmi.objectives.n.completion_status Data Model Requirements

9.1.2.14 cmi.student_data.mastery_score

The cmi.student_data.mastery_score element is a indicator of a passing normalized score for that SCO as determined outside the SCO.

cmi.student_data.mastery_score	
Supported API calls: LMSGetValue() **Data Type:** Normalized real number between –1 and 1. (inclusive **SCO Accessibility:** Read Only	**Definition:** The passing score, as determined outside the SCO. When the SCO score is greater than or equal to the mastery score, the student is considered to have passed, or mastered the content. In some cases, the SCO does not know what this passing score is, because it is determined by the LMS system. **Usage:** For an LMS system to support mastery_score, it must be able to change the success_status based on the score passed to if from the SCO. Just passing a mastery_score to a SCO does not constitute full support for this feature. **Format:** Decimal number between –1 and 1 (inclusive). **LMS Behavior:** ▪ **Initialization:** LMS is responsible – value obtained from manifest. An item's primary objective's <imsss:minNormalizedMeasure> is used for initialization. ▪ **LMSGetValue():** Returns the current value stored by the LMS. o **Example Return Values:** "0.75" "-0.23" ".5" o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **LMSSetValue():** LMS should set an error code according to the following: o **Error Code:** ▪ **403 - Element is read only.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. **SCO Usage Example:** var masteryScoreValue = LMSGetValue("cmi.student_data.mastery_score");

Table 9.1.2.14a: cmi.student_data.mastery_score Data Model Requirements

9.1.2.15 cmi.student_data.max_time_allowed

The cmi.student_data.max_time_allowed element is a indicator of the maximum time that is permissible on any attempt of a SCO.

cmi.student_data.max_time_allowed	
Supported API calls: LMSGetValue() **LMS Mandatory:** No **Data Type:** CMITimespan **SCO Accessibility:** Read Only	**Definition:** The amount of time the student is allowed to have in the current attempt on the SCO. See time_limit_action for the SCO's expected response to exceeding the limit. **Usage:** Used to present the SCO with the maximum amount of time that the student is allowed to be in the SCO. **Format:** Hours, minutes and seconds separated by a colon. HHHH:MM:SS.SS Hours has a minimum of 2 digits and a maximum of 4 digits. Minutes shall consist of exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits. (i.e. 34.45). **LMS Behavior:** ▪ **Initialization:** LMS is responsible – value obtained from manifest (*ActivityExperiencedDurationLimit*).

	▪ **LMSGetValue():** Returns the current value stored by the LMS. o **Example Return Values:** "00:14:30" "02:03:00" "01:09:00" o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **LMSSetValue():** LMS should set an error code according to the following: o **Error Code:** ▪ **403 - Element is read only.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. **SCO Usage Example:** var maxTimeAllowedValue = LMSGetValue("cmi.student_data.max_time_allowed");

Table 9.1.2.15a: cmi.student_data.max_time_allowed Data Model Requirements

This page intentionally left blank.

APPENDIX A
Acronym List

This page intentionally left blank.

Acronym Listing

ADL	Advanced Distributed Learning
API	Application Program Interface
CMI	Computer Managed Instructions
CP	Content Packaging
DC	Dublin Core
DoD	Department of Defense
DOM	Document Object Model
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
LMS	Learning Management System
LOM	Learning Objects Metadata
LTSC	Learning Technology Standards Committee
MD	Meta-data
MIME	Multipurpose Internet Mail Extensions
PIF	Package Interchange Format
RTE	Run-Time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
URI	Universal Resource Identifier
URL	Universal Resource Locator
XSD	XML Schema Definition
XML	eXtensible Markup Language

This page intentionally left blank.

APPENDIX B
References

This page intentionally left blank.

References

1. Final 1484.12.1-2002 Learning Object Metadata draft standard
   Available at: http://ltsc.ieee.org/

2. IMS Simple Sequencing Specification Version 1.0 Public Draft
   Includes:
      IMS Simple Sequencing Information and Behavior Model
      IMS Simple Sequencing Best Practice and Implementation Guide
      IMS Simple Sequencing XML Binding
   Available at http://www.imsglobal.org/

This page intentionally left blank.

APPENDIX C
Revision History

This page intentionally left blank.

Revision History

Version	Comment
Draft 0.9	Initial released draft version